Edgar Soldin – Stadtweg 119 – 39116 Magdeburg; Telefon 0171 278 2880; Emailadresse edgar@soldin.de; Web http://www.soldin.de

Mittwoch, 14. April 2004

Lizenzbedingungen zur Nutzung der Diplomarbeit

Sehr geehrter Leser, verehrte Leserin,

die nachfolgende Arbeit von Edgar Soldin zum Thema „Mobile Bearbeitung und Erstellung von Geodaten durch satellitengestützte Positionsbestimmung“ erstellt am 30.1.2004 wird Ihnen zu den folgenden Bedingungen zur kostenlosen Nutzung überlassen.

1. Es steht Ihnen frei, die Arbeit in ihrer ursprünglichen Form (pdf-format) und mit unverändertem Inhalt (diese Lizenzbedingungen, Verzeichnisse, kompletter Inhalt, Anhänge) zu vervielfältigen und kostenlos an Dritte weiterzugeben.

2. Die auszugsweise Verwendung der Arbeit (z.B. für die Erstellung einer anderen nicht-kommerziellen Arbeit) ist nur erlaubt, wenn Autor, Bezeichnung der Publikation und Erscheinungsdatum* explizit an geeigneter Stelle (Literaturliste, Credits, Abspann, Danksagung) erwähnt werden und das resultierende Werk ebenfalls kostenlos durch Dritte bezogen werden kann.
   * z.B. Edgar Soldin, „Mobile Bearbeitung und Erstellung von Geodaten durch satellitengestützte Positionsbestimmung“, 30.1.2004

3. Jede Verwendung dieser Arbeit (komplett oder in Teilen) für einen kommerziellen Zweck (z.B. kostenpflichtige Weitergabe oder für die Erstellung kostenpflichtiger Werke) oder zu anderen als den hier genannten Bedingungen ist nicht erlaubt.

Viel Spaß bei der Lektüre
Edgar Soldin

D I P L O M A R B E I T

Mobile Bearbeitung und Erstellung von Geodaten durch satellitengestützte Positionsbestimmung

unter der Leitung von
Prof. Dr. Bernd Müller
Dr. Holger Hinrichs

eingereicht an der Hochschule Harz
Fachbereich Wirtschaftswissenschaften
Studiengang Medieninformatik

von
Edgar Soldin
Stadtweg 119
39116 Magdeburg

Magdeburg, 30. Januar 2004

Inhaltsverzeichnis

Kapitel 1

Einleitung

Bis in das Jahr 3800 v.Chr. reichen die Wurzeln der wissenschaftlichen Grundlage dieser Diplomarbeit. Die erste überlieferte Landkarte der Welt, eine in die Tontafel von Nuzi (Ga-Sur) geritzte Karte von Nord-Mesopotamien[1] wird auf diese Zeit datiert und markiert somit die Entstehung der Kartographie. Es folgt eine parallele Entwicklung der Kartographie im griechischen und römischen Teil der Welt. Während in Griechenland die Bestimmung der astronomischen Lage der Erde und ihrer Gestalt im Vordergrund steht, werden Karten im Römischen Reich für praktischere Ziele benötigt. Griechisches Wissen wird für die Darstellung des Weltreiches und der visuellen Struktur für dessen Verwaltung verwendet. Doch die Abwendung von der Wissenschaft bleibt nicht ohne Folgen. Nach dem Zerfall der beiden Imperien im 4. Jahrhundert n.Chr. werden bis ins 15. Jahrhundert Karten anhand von griechischen Erkenntnissen aus dem ersten Jahrhundert n.Chr. gefertigt[2]. Erst im 16. Jahrhundert wird die Kartographie wieder zu einer Wissenschaft und vor allem in Europa weiterentwickelt. Die neuen Handwerke Landkartenstich und Buchdruck fördern die allgemeine Verbreitung von kartographischen Darstellungen. Mitte des 18. Jahrhunderts entstehen die ersten topographischen[3] Karten durch die große Triangulation[4] Frankreichs der Gebrüder Cassini. Diesem Beispiel wird weltweit Folge geleistet, so dass im 19. Jahrhundert große Teile Europas, Amerikas und Asiens kartographiert werden. Mit der aktiven Erkundung von ”weißen Flecken”, den unbekannten Regionen auf Landkarten, entsteht aus der Kartographie die neue Wissenschaft Geographie. Die Gründung privatwirtschaft-

[1] (Vgl. Wikipedia 2004, Kartographie)
[2] (Vgl. Wikipedia 2004, Kartographie, Frühgeschichte, Karte des Ptolemäus)
[3] ”Die Topographie befasst sich mit der Lagebeschreibung und Vermessung von Orten und Landschaften durch Topographen (Vermessungsingenieure). Das Ergebnis der topographischen Arbeit stellen die Landkarten dar.”(Wikipedia 2004, Topographie)
[4] ”Ein Meßverfahren, mit dem ein Dreiecksnetz, abgesehen von der Bestimmung des Maßstabs, nur durch Winkelmessung geschaffen wird.”(Institut für Geodäsie und Geoinformatik Universität Rostock 2003, Triangulation)

licher Institute und Gesellschaften, sowie militärischer Abteilungen hat die Erstellung und Verbreitung vielfältiger Spezialkarten zur Folge. Bis zum Ende des 19. Jahrhunderts kann Geographie als eine rein deskriptive Wissenschaft beschrieben werden. Es wurden Tatsachen gesammelt und in Bildern, Karten oder Berichten beschrieben. Mit der Eroberung des Luftraumes und des Weltalls Mitte des 20. Jahrhunderts werden zusätzlich zur klassischen Geodäsie[5] die Methoden der Fernerkundung und Photogrammetrie[6] für die Vermessung nutzbar. Die Qualität und Vollständigkeit der Karten erreicht ein bis dahin unbekanntes Niveau. Mit der Erfindung von Rechenmaschinen in der Mitte und der Nutzung von Computern Ende des 20. Jahrhunderts verändern sich schließlich Tätigkeitsbild und Funktion der Geographie.

Im Jahr 2004 ist Geographie eine komplexe Wissenschaft und untersucht Zusammenhänge zwischen Mensch und Erdoberfläche[7]. Anstelle von Leuchttisch, Feder und Griffel arbeiten Kartographen mit Maus, Graphiktablett und Geoinformationssystem (kurz GIS). Landvermesser werden Geodät genannt und Fachmessen für Geoinformationssysteme sind Publikumsmagneten[8] nicht nur für Fachleute. Geodaten werden von Brokern gehandelt und der Weg von A nach B per Routenplaner im Internet ermittelt. Geoinformationssysteme sind allgegenwärtig[9]. In der Wirtschaft werten GIS geographische Daten aus. Die Populationsdichte und die Lage der Mitbewerber haben Einfluss auf den Standort von Einkaufszentren. Die Lager von Großhändlern entstehen logistisch vorteilhaft an Orten mit den kürzesten und schnellsten Lieferwegen. Ver- und Entsorger verwalten und planen ihre Leitungen mit GIS. Telekommunikationskonzerne überprüfen die Netzabdeckung und Landwirte die Ertragsdichte und Bodenzusammensetzung ihrer Felder durch Auswertung der Messungen in einem GIS. Großstädte planen den Verkehrsfluss und den Bau öffentlicher Einrichtungen mithilfe von GIS. Für jeden möglichen Standort kann hiermit die Erreichbarkeit öffentlicher Verkehrsmittel oder Versorgungspunkte ermittelt werden. Verkehrsbetriebe optimieren mit GIS ihre Routen. Von der Erschließung der Ölfelder bis zum Transport über kürzeste Wege profitiert die Ölindustrie von GIS. Doch auch Anwendungsgebiete wie Geomarketing entstehen. In einer Zeit zurückgehender Werbeausgaben wird versucht, die teure Botschaft effizienter an den potentiellen Kunden zu bringen. Doch wo befindet

[5] "Geodäsie [...] - (auch Vermessungskunde, Vermessungswesen) ist eine Geowissenschaft, die sich mit der Vermessung der Erdoberfläche und ihrer Objekte sowie dem Erdschwerefeld beschäftigt." (Wikipedia 2004, Geodäsie)

[6] "In der Photogrammetrie rekonstruiert man aus Abbildungen (z. B. einer perspektivischen Abbildung der Fotografie) eines Gegenstandes seine dreidimensionale Form. Man unterscheidet zwischen Luftbildphotogrammetrie und Nahbereichsphotogrammetrie."(Wikipedia 2004, Photogrammetrie)

[7] (Vgl. Wikipedia 2004, Geografie)

[8] (Vgl. Telepolis 2004, Geoinformationssysteme werden IT-Mainstream, Krystian Woznicki, 2000)

[9] (Vgl. Geoscience Online 2004, Geodaten überall...)

sich dieser? Wie viele Kunden befinden sich auf welcher Fläche? Fragen, die ein GIS, anhand georeferenzierter Marketing Rohdaten, beantworten kann.
Seit den 90er Jahren des letzten Jahrhunderts ist das Global Positioning System (GPS) der USA voll funktionsfähig und zivil nutzbar. Mit dieser Technologie ist es möglich im Außeneinsatz digital und unter Umständen zentimetergenau die globale Position zu bestimmen. Früher mussten die Daten umständlich und fehlerbehaftet von Hand erfasst und konnten erst im Büro mit anderen Referenzdaten verglichen, korrigiert und dann in Karten übertragen werden. Heutzutage kann unter Zuhilfenahme (noch teurer) mobiler GIS-Technik komplett auf den Medienbruch zwischen Papier und Informationstechnologie verzichtet werden. Doch nicht nur für Geodäten ist GPS von Vorteil: Navigationssysteme, im Marinekreuzer oder kleinen Stadtautomobil, nutzen GPS-Signale zur Positionsbestimmung und in Kombination mit GIS-Funktionalität zur Ermittlung von Routen oder automatischen Navigation. Die Daten weiterer Informationsquellen (Rundfunk, Satellit o.ä.), die über Stau oder Unwetter informieren, können in Betracht gezogen und die ermittelten Wege und Geschwindigkeiten angepasst werden[10]. Auch in die Unterhaltung hat GPS Einzug gefunden. Im Spielkonzept ”Can you see me now?” wird GPS benutzt, um virtuelle Personen in realen Umgebungen ausfindig zu machen[11]. Zur vorbereitenden Erfassung des realen Spielfeldes werden GIS und GPS gleichermaßen verwendet. GPS ist nicht die einzige Lösung. Positionsbestimmung ist auch (bei eingeschränkter Genauigkeit) mit Mobiltelefonen möglich. Szenarien für Nutzungskonzepte z.B. zur Lebensrettung oder in Verbindung mit GPRS/UMTS für Infotainment-Zwecke werden entwickelt. Konkurrierend zu GPS sind weitere satellitengestützte Positionssysteme vorhanden oder in Planung (Abschnitt 2.3 auf Seite 18).
Die Bestimmung der globalen Position in Echtzeit ermöglicht vielfältige neue standortbezogene Anwendungen. In Verbindung mit Informationen und Diensten in Abhängigkeit vom Standort des Nutzers können standortbezogene Dienste (Location Based Services) angeboten werden. GIS-Funktionalität kombiniert mit Echtzeitpositionsbestimmung ermöglicht die Erstellung hochspezialisierter kompakter Anwendungen z.B. für Wissenschaft, Unterhaltung oder Handwerk.
Welche Geschäftsmodelle und Angebote sich zukünftig durchsetzen werden, hängt vor allem von der Akzeptanz des Nutzers ab. Faktoren wie Bedarf an Mobilität, technische Anforderungen und tatsächlicher wirtschaftlicher Nutzen sind wichtige Kriterien[12]. Die technische Machbarkeit allein wird keine Nachfrage verursachen.

[10] (Vgl. Golem News 2003)
[11] Oldenburg, Blast Theory
[12] (Vgl. Ranzinger: Einsatz mobiler GIS bei Energieversorgungsunternehmen in den Bereichen Instandhaltung und Störungsmanagement, in: Zipf, Strobl 2002, S.204-209)

Das Ziel dieser Diplomarbeit ist die Verschmelzung der Technologien GIS und satellitengestützter Positionsbestimmung zu einem mobilen Werkzeug. Eine funktionstüchtige *mobile GIS-Anwendung*, welche grundsätzliche Merkmale eines Geoinformationssystems (Abschnitt 2.2.1 auf Seite 10) mit der Funktion satellitengestützter Positionsbestimmung verbindet, soll das Ergebnis dieser Diplomarbeit sein. Die Bearbeitung und Erstellung von Geodaten soll nicht nur mit, sondern durch die Positionsbestimmung geschehen. Geodaten sollen mit dieser Anwendung schon im Feld komplett digital erhoben und auch modifiziert werden können.

Die Struktur dieser Diplomarbeit wurde so entworfen, dass die einzelnen Schritte der Bearbeitung des Themas nachvollzogen werden können. Das nachfolgendende Kapitel 2 auf der nächsten Seite ist das Ergebnis der Einarbeitung in das Thema. Grundlegendes Wissen aus den Bereichen Geografie, Geoinformatik, Satellitengestützte Positionsbestimmung und Softwarelizenzierung wurde erarbeitet. Das sich daran anschließende Kapitel 3 auf Seite 33 dient der Analyse der Zielstellung. Die Vision des mobilen GIS wird in ihren Einzelheiten erläutert und Möglichkeiten für ihre Nutzung vorgestellt. In Kapitel 4 auf Seite 40 wird die tatsächliche Umsetzung der Anwendung beschrieben. Nach der Formulierung eines abstrakten Konzeptes[13] werden bereits existierende Software-Komponenten mit benötigten Funktionen recherchiert und auf Verwendbarkeit geprüft. Ergebnis dieser Evaluation ist ein konkretisierter Konzeptentwurf, der das abstrakte Konzept auf die geeigneten Komponenten abbildet. Mit diesem konkreten Konzept werden fehlende Funktionen und Schnittstellen offensichtlich, deren Umsetzung im letzten Abschnitt thematisiert wird. Das abschließende Kapitel 5 vergleicht kritisch die Zielsetzung des Anfangs mit dem erreichten Produkt und beurteilt den Erfolg der Arbeit.

Hinweis: Im Anhang C der Diplomarbeit befindet sich ein Datenträger, der neben Quellen und Software zur Diplomarbeit auch das Dokument selbst als Datei im PDF-Format (mobigis.pdf) enthält. Als zusätzlicher Mehrwert sind in dieser Datei *alle* Quellverweise auf Internet-Seiten mit Verknüpfungen versehen. Quellcode-Listings sind zum besseren Verständnis farbig ausgezeichnet.

[13] beschreibt die geplante Funktionalität der Anwendung unter Beachtung von Beschränkungen der realen Welt (das allgemein technisch Machbare) aber losgelöst von spezifischen Implementierungsmerkmalen (z.B. Programmiersprache, Prozessortyp ...)

Kapitel 2

Grundlagen

Dieses Kapitel erarbeitet Grundbegriffe und Technologien aus den Bereichen Geographie, Geoinformatik, Satellitengestützte Positionsbestimmung und Softwarelizenzen. Diese Kenntnisse bilden die Basis des Diplomthemas und sind somit Vorraussetzung für die Umsetzung im praktischen Teil dieser Arbeit. Dabei muss sich auf ausschnittsweise Erläuterungen beschränkt werden, um den Rahmen nicht zu sprengen.

2.1 Räumliche Objekte und Bezugssysteme

Räumliche Objekte und die Bezugssysteme, in denen Objekte Gültigkeit besitzen, werden in diesem Abschnitt thematisiert. Es wird gezeigt, wie Phänomene der realen Welt abstrahiert und auf Karten oder in dreidimensionalen Modellen abgebildet werden können. Zuletzt wird auf das geodätische Datum eingegangen, das als Methode zur Beschreibung eines geozentrischen Bezugssystems die Grundlage für Transformationen bildet.

2.1.1 Geoobjekte

> ”Geoobjekte sind räumliche Elemente, die zusätzlich zu Sachinformationen geometrische und topologische Eigenschaften besitzen und zeitlichen Veränderungen unterliegen können. Kennzeichnend für Geoobjekte sind somit Geometrie, Topologie[1], Thematik und Dynamik.”[2]

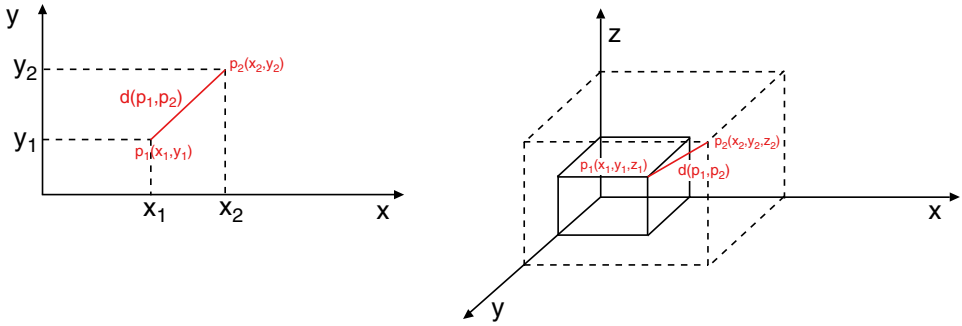
[1] ”(gr. Lehre von den Örtern) ist die Lehre von allgemeinen räumlichen Beziehungen und Eigenschaften des Raumes. Der Begriff wird spezifisch in verschiedenen Wissenschaften gebraucht...” (Vgl. Wikipedia 2004, Topologie, 4.1.2001)

[2] (Lange 2002, S.157)

Somit ist ein Geoobjekt ein Raumelement mit Informationen zur Geometrie (Form, Fläche, Volumen, Lage, etc.), Topologie (relative Lage zu anderen Objekten, Überschneidungen), Thematik (Art des Objektes, Bezeichnung) und Dynamik (Veränderungen erstgenannter Kriterien über die Zeit).

2.1.2 Koordinatensysteme

Mithilfe von Koordinatensystemen lassen sich Punkte in n-dimensionalen Räumen beschreiben. Um *einen* Punkt in einem n-dimensionalen Koordinatensystem zu beschreiben wird mindestens *ein* n-stelliges Tupel[3] benötigt, wobei jedes Element als eine Koordinate bezeichnet wird[4].



$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Abbildung 2.1: Kartesische Räume mit euklidischer Metrik

Kartesische Koordinatensysteme

Eine zentrale Bedeutung besitzen rechtwinklige Koordinatensysteme (kartesische Koordinatensysteme), indem sie die Grundlage für die Darstellung und Positionierung von Geoobjekten im Raum bilden. So wird normalerweise für geometrische Berechnungen oder die digitale Bearbeitung von Geodaten ein kartesisches Koordinatensystem vorausgesetzt. Verwendet wird die euklidische Metrik, welche in der natürlichen Luftlinienentfernung eine Entsprechung findet[5]. Durch senkrech-

[3] "Ein Tupel, häufig auch N-Tupel, ist ein Begriff aus der Mathematik. Er bezeichnet eine geordnete Zusammenstellung von Objekten, die als Elemente, Komponenten oder Einträge des Tupels bezeichnet werden. N bezeichnet hierbei die Anzahl der Elemente des Tupels. Diese Anzahl muss abzählbar sein. Üblicherweise werden die Elemente eines Tupels mit Hilfe der natürlichen Zahlen indiziert." (Wikipedia 2004, Tupel, 4.1.2004)

[4] (Vgl. Wikipedia 2004, Koordinatensystem, 4.1.2004)

[5] (Vgl. Lange 2002, S.165)

tes (orthogonales) Aufeinanderstellen kartesischer Koordinatensysteme mit gleicher Achseneinteilung und identischem Ursprung werden n-dimensionale Koordinatensysteme geschaffen (Abbildung 2.1 auf der vorherigen Seite). Mit diesen können Punkte im n-dimensionalen Raum durch Angabe der Achsenabschnitte (Koordinaten) eindeutig dargestellt werden.

Polarkoordinatensystem und geographisches Koordinatensystem

Um die dreidimensionale, geozentrische[6] Realität mit einem entsprechenden dreidimensionalen Koordinatensystem zu beschreiben, werden Polarkoordinatensysteme verwendet[7]. Diese entsprechen zwei kartesischen Koordinatensystemen, welche wie oben beschrieben gestapelt sind. Zu beachten ist, dass durch einfachen Austausch der Achsenabschnitte gegen Winkelangaben in der entsprechenden Dimension eine anschauliche Ortsangabe eines Punktes auf einer Kugel ermöglicht wird (Abbildung 2.2).

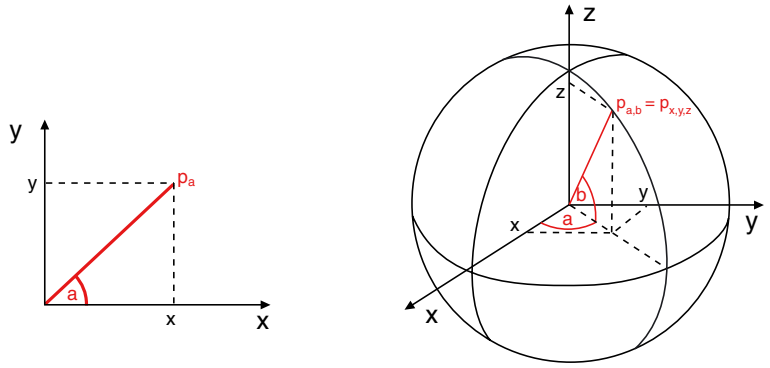


Abbildung 2.2: kartesische Koordinate und polare Koordinate

Wenn nun der Abstand zum Kugelmittelpunkt dem mittleren Erdradius (6371km) gleichgesetzt wird, ist durch alleinige Angabe der Winkel, genannt geographische Länge (a) und Breite (b), ein Punkt auf der Erdoberfläche definierbar. Dieses spezielle Koordinatensystem wird geographisches Koordinatensystem genannt und ist der de facto Standard der globalen Ortsbestimmung. Die Winkel können und werden zumeist auch in 60er Schritte unterteilt, wobei ein Grad 60 Minuten entspricht. Z.B. 52,276388 Grad entsprechen dann 52 Grad 16’ 35” (’Minuten, ”Sekunden).

[6] Erdmittelpunkt als Ursprung
[7] (Vgl. Lange 2002, S.166ff)

2.1.3 Karten und Kartennetzentwürfe

Kartographische Darstellungen (Karte oder auch Projektion genannt) auf Bildschirmen oder Ausdrucken unterliegen einer Beschränkung. Durch die fehlende dritte Dimension muss die dreidimensionale, gekrümmte Erdoberfläche in der flachen Ebene abgebildet werden[8]. Prinzipiell ist es nicht möglich größere Ausschnitte der Kugeloberfläche in der Fläche verzerrungsfrei darzustellen. Insbesondere zum Kartenrand hin ergeben sich Ungenauigkeiten in Bezug auf Längen-, Flächen- und Winkeltreue. Es wird versucht, dieses Problem zu umgehen, indem so genannte Kartennetzentwürfe entweder winkel- oder flächengenau hergestellt werden. Es gilt die Faustregel: Je größer der Ausschnitt desto ungenauer die Darstellung[9]. So werden die kartographischen Darstellungen anhand des Maßstabes (des Grades der Verzerrung) in zwei Kategorien eingeteilt[10].

Maßstab 1:500.000–∞ Gradnetzkarten[11], z.B. kleinmaßstäbige Karten von Kontinenten. Die Verzerrung ist bei dieser Art Karte sehr gut an dem krummlinienförmigen Gradnetz zu erkennen.

Maßstab 1:0–500.000 Gitternetzkarten, z.B. großmaßstäbige Stadtpläne, sind sehr kleine Ausschnitte, die mit kleineren Verlusten in die Ebene (ein kartesisches Koordinatensystem) übertragen wurden. Namensgebend sind die, wie bei einem Gitter, senkrecht aufeinander stehenden geraden Längen- und Breitengrade.

Die Lage der Abbildungsfläche zur abzubildenden Kugeloberfläche hat Auswirkungen auf die Qualität der Projektion. Die Auswahl einer Projektionsart ist vom angestrebten Verwendungszweck abhängig. Hauptsächlich werden folgende Arten unterschieden[12]: Azimutal-, Zylinder- und Kegelprojektion (Abbildung 2.3).

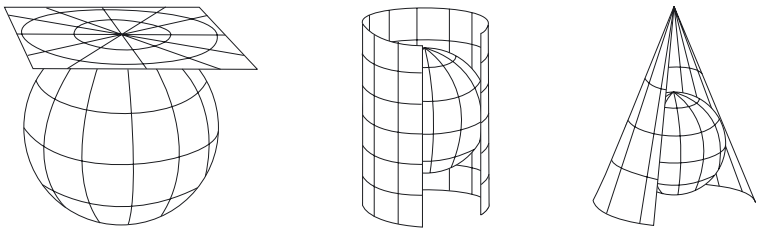


Abbildung 2.3: Azimutalprojektion, Zylinderprojektion, Kegelprojektion

[8] (Vgl. Stahl, R.; Greve, K. (Hrsg.) 1998, Koordinatensysteme, Geodätische Bezugssysteme, Kartenprojektionen)
[9] (Vgl. Lange 2002, S.174ff); (Vgl. Olbrich, Quick, Schweikart 2002, S.29ff)
[10] (Vgl. Lange 2002, S.175)
[11] 1:1.000.000 laut Olbrich et al. (2002, S.29)
[12] (Vgl. Lange 2002, S.176ff)

Eine Verschiebung des Auflagepunktes und damit der akkuraten Mitte der Abbildungsfläche wird durch die Bezeichnungen äquatorständig, polständig oder schiefständig wiedergespiegelt. Bekanntes Beispiel ist die äquatorständige Zylinderprojektion, die alle Kontinente und Meere in einer Übersichtskarte darstellt (Abbildung 2.4). Auch als Seekarte ist diese Darstellung besonders geeignet, da trotz der starken Proportionsverzerrungen am oberen und unteren Ende der Karte, z.B. Grönland größer als Südamerika, diese Projektion winkelgenau ist und damit eine Navigation per Kompass visualisieren kann.

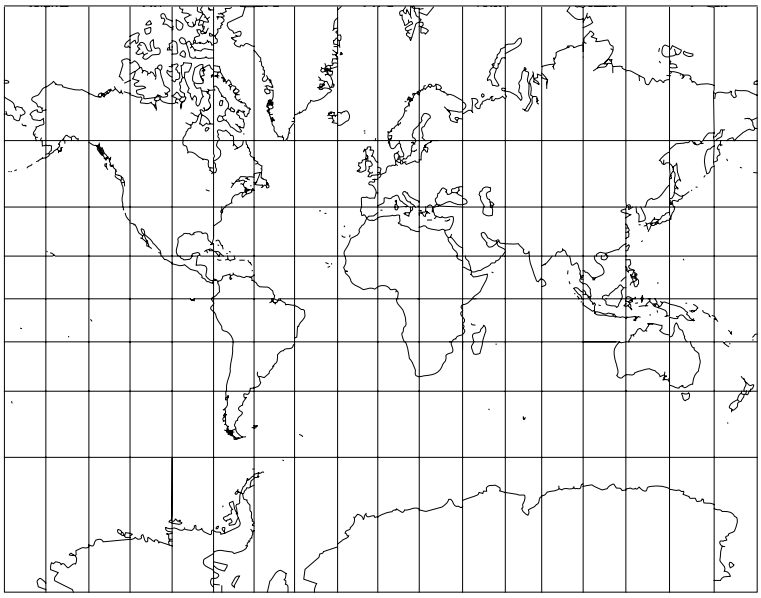


Abbildung 2.4: Zylinderprojektion mit Verzerrung am oberen und unteren Rand nach Lange (2002, S.179)

Winkeltreue Zylinderprojektionen werden auch Mercatorprojektion genannt. Die Universale Transversale Mercatorprojektion (UTM) und die deutsche Gauß-Krüger-Projektion sind transversale, winkeltreue Zylinderprojektionen.

2.1.4 Ellipsoide

Die Erde ist nur vereinfacht als Kugel zu betrachten. Genauere Messungen beweisen Abflachungen an den Polen und eine Ausbuchtung am Äquator, sowie weitere Abweichungen von der modellhaften Kugelform. Durch die Verwendung von Rotationsellipsoiden, um eine ihrer zwei Achsen rotierende Ellipsen, kann die Ungenauigkeit der Kugel für die Ortsfestlegung von Punkten an der Erdoberfläche teilweise ausgeglichen werden. So wird eine um die kürzere Achse, die Polarachse, rotierende Ellipse verwendet, um die Abflachung der Pole auszugleichen. WGS84

und WGS72 sind solche weltweiten Referenzellipsoide, die aber für lokale Vermessungen zu ungenau sind. Aus diesem Grunde werden verschiedene Referenzellipsoide verwendet, welche ausschnittsweise genau für verschiedene Regionen auf der Erdoberfläche sind[13].

2.1.5 Geodätisches Datum und Transformation

Das geodätische Datum (GD) ist ein Festpunktfeld, also ein Satz von Parametern, der Ursprung, Orientierung und Maßstab eines Bezugssystems im Verhältnis zu einem absoluten Bezugssystem enthält, sowie die Spezifikation des Referenzellipsoiden dieses absoluten Bezugssystems[14]. Durch die Spezifikation des geodätischen Datums für völlig unterschiedliche Kartennetze und Projektionen ist es möglich, diese miteinander zu vergleichen beziehungsweise deren Daten ineinander umrechenbar zu machen. Diese Umrechnung wird Datumstransformation genannt und ist nur dann direkt möglich, wenn beide Daten sich auf ein und dasselbe absolute Bezugssystem beziehen. Falls dies nicht der Fall ist, muss ein Umweg über (evtl. mehrere) andere Bezugssysteme gesucht werden[15].

2.2 Grundlagen der Geoinformatik

Geoinformatik (selten auch Geomatic genannt) ist ein neues interdisziplinäres Fachgebiet. Es schlägt eine Brücke zwischen den traditionellen Fachgebieten Informatik, Geographie und geographische Informationstechnologie[16].
Typisch für die Entstehung eines Wissenschaftszweiges, fehlen aktuell noch umfassende wissenschaftstheoretische Diskussionen um dieses Thema. Allerdings sind seit den 90er Jahren internationale Bestrebungen erkennbar, Geographic Information Science (Geoinformationswissenschaften) zu etablieren[17]. Die Einforderung der Abkürzung GIS für Geographic Information Science erscheint aufgrund der Popularität dieses Begriffs für einen anderen Aspekt der Geoinformatik problematisch.

2.2.1 GIS

Der eingängige Begriff GIS wird normalerweise für geografisches Informationssystem verwendet. Da ein GIS allerdings nur die technologischen Aspekte der

[13] (Vgl. Lange 2002, S.179ff)
[14] (Vgl. Lange 2002, S.181)
[15] (Vgl. Lange 2002, S.181ff)
[16] (Vgl. Lange 2002, S.1ff)
[17] (Vgl. Goodchild 1990)

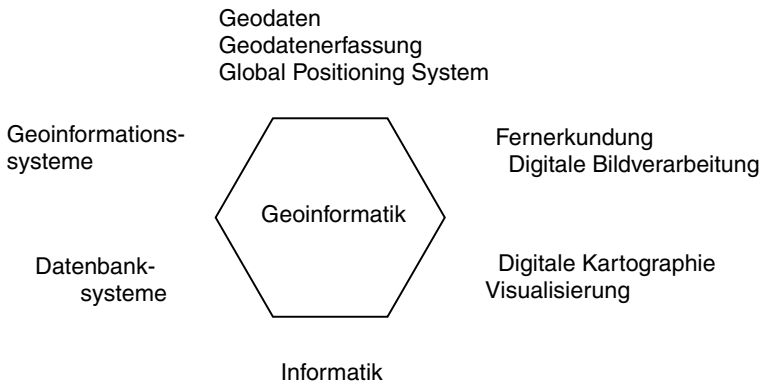


Abbildung 2.5: Teilbereiche der Geoinformatik
nach Lange (2002, S.4)

Geoinformatik beschreibt, sollte bei einer Verwendung dieses Begriffes für Geographic Information Science oder Geoinformatik im Allgemeinen explizit darauf hingewiesen werden[18].

Seit etwa 1975, durch die Verfügbarkeit leistungsfähiger Computersysteme ausgelöst, werden anwendbare Geoinformationssysteme entwickelt. Vor allem proprietäre kommerzielle Lösungen, welche die aufwändige manuelle Verarbeitung von Geodaten (Vermessung, Erstellung von Karten etc.) ersetzen und somit Geld- und Zeiteinsparungen versprechen, können sich durchsetzen. Erste Kunden sind Behörden mit großen Geodatenbeständen. Die Entwicklung verlagert sich zusehends von Universitäten zu Privatfirmen. Die Firma ESRI, marktdominanter Hersteller für professionelle GIS-Lösungen, ist aus einem Projekt der Harvard Universität hervorgegangen.

Als in den 1990er Jahren kostengünstigere leistungsfähige Personal Computer (PC) eingeführt und Speichermedien, Monitore, graphisches Zubehör (z.B. Graphiktabletts) immer preiswerter werden, erlebt die Verbreitung von GIS einen neuen Schub[19]. Von Großrechnern verlagern sich GIS-Anwendungen auf die Arbeitsstationen und der Begriff Desktop-GIS wird geprägt. Seit 2001 sind GIS-Hersteller mit einer eigenen Halle bei der weltgrößten Computermesse, der Cebit, vertreten[20].

Eine Definition für den Begriff GIS ist in Lange (2002, S.310) zu finden:

> ”Ein Geoinformationssystem ist ein rechnergestütztes System, das aus Hardware, Software, Daten und Anwendungen besteht. Mit diesem

[18] (Vgl. Lange 2002, S.309ff)
[19] (Vgl. Stahl, R.: GIS Historie, in: Stahl et al. 1998)
[20] (Vgl. Lange 2002, S.316)

können raumbezogene Daten digital erfasst, gespeichert, verwaltet, aktualisiert, analysiert und modelliert, sowie alphanumerisch und graphisch präsentiert werden. [...] Die zentralen Gegenstände dieser Informationssysteme sind Informationen über Geoobjekte."

Datenstrukturen

Im GIS werden raumbezogene Daten als Rasterdaten oder als Vektordaten (Geometrien) erfasst beziehungsweise verarbeitet und ausgewertet[21].

Rasterdaten entstehen durch das Scannen von Bildern oder durch die direkte Aufnahme per Digitalkamera. Die Aufnahme wird dabei als ein Raster von Bildpunkten gespeichert. Das digitale Bild (Matrix von Farb- oder Grauwerten) wird nun mit einem Referenzkoordinatensystem in Bezug gesetzt, z.B. durch Zuweisung von Bildpunkten zu bekannten Koordinaten. Dieser Vorgang wird Georeferenzierung oder Geokodierung genannt. Nun können mit Methoden der digitalen Bildbearbeitung die Farb- und Helligkeitsinformationen der Pixel ausgewertet werden (Klassifizierung). Pixel gleicher Klassen bilden virtuellen Flächen (z.B. Waldflächen, Gebäudedächer), die für Auswertungen zur Verfügung stehen. Rasterdaten enthalten in der Regel keine Sachdaten und werden deshalb oft nach der Geokodierung zur Digitalisierung von Vektordaten verwendet. So macht es wenig Sinn, alle Pixel einer Seefläche mit Eigenschaften des Sees zu verknüpfen. Allerdings kommt es vor, dass Rasterdaten eine Legende mit bestimmten Farb- oder Grauwerten und Sachdaten zu diesen beigefügt wird, um Analysen reproduzierbar zu machen.

Vektordaten sind durch Linien oder Kurven verbundene Punkte. Während die Verbindung durch mathematische Definitionen beschrieben wird, sind Start- und Endpunkt Koordinaten in einem geographischen Referenzsystem. Mit Vektordaten lassen sich komplexe Vektorgrafiken (Geometrien genannt) erzeugen. Linien oder Polygone[22] können z.B. Strassen, Grenzverläufe oder Gewässerflächen darstellen.

Moderne GIS bieten die gleichzeitige (hybride) Verarbeitung von Raster- und Vektordaten an. Ein aktueller Trend ist die Visualisierung und Bearbeitung in drei Dimensionen[23].

[21] (Vgl. Stahl, R.: Datenstrukturen oder what's so special about spatial?, in: Stahl et al. 1998)
[22] dt. Vieleck
[23] (Vgl. o.V.: GRASS 3D/4D raster volume Interface (voxel support), in: GRASS-Projekt 2004)

Objektorientierung

Länger schon werden objektorientierte[24] Strukturen zur Modellierung von GIS verwendet. Die reale Welt besteht aus Objekten mit Eigenschaften, die durch geometrische Beschreibung in Form von Flächen und Körpern nur unvollständig dargestellt wird. Das derzeit noch dominierende planorientierte Denken wird durch objektorientiertes Denken ersetzt, in dem Lage und Form von Geoobjekten nur weitere Attribute sind[25]. Tatsächlich ist schon 1988 mit Smallworld das erste objektorientierte GIS entwickelt worden[26].

2.2.2 Standards

Im Laufe der Zeit entstanden GIS-Lösungen von verschiedenen Herstellern für verschiedene Zwecke. Diese basieren auf unterschiedlichen Datenstrukturen und Speicherformaten. Durch die Nutzung dieser Systeme wurden inzwischen weltweit große Geodatenbestände angesammelt. Das ansteigende (kommerzielle) Interesse an der Verwendung dieser Geodaten[27], besonders in den Teilmärkten klassische GIS-Anwendungen, Navigation und Geomarketing[28], wird durch diese Inkonsistenzen stark gebremst. ”Eine breite interoperable Nutzung von Geodaten ist noch nicht existent.”, resümiert Winfried Kopperschmidt (Vorstandsmitglied Runder Tisch GIS e.V.) in einem Interview mit Schaeff (2003). Die von der Wirtschaft geforderte Interoperabilität von Geodaten herzustellen, ist das Ziel mehrerer nationaler und internationaler Organisationen[29]. Zwei Organisationen tragen momentan maßgeblich zur Entwicklung offener Standards in der Geoinformatik bei: Open GIS Consortium und ISO TC 211.

Das Open GIS Consortium (OGC) ist eine internationale Non-Profit-Initiative von Datenbank- und GIS-Herstellern. Das OGC hat sich zum Ziel gesetzt, einheitliche und offene Spezifikationen (u.a. für Datenformate, Datenmodelle, Koordinaten-Referenz-Systeme und Geometrien) zu erarbeiten, welche die Grundlage für zukünftige kompatible GIS-Anwendungen bilden sollen.

ISO TC 211 , bezeichnet das Technical Committee - Geographic Information/-Geomatics (TC 211) der ISO (International Standards Organisation), das als zweites wichtiges Normgremium damit beschäftigt ist, die Normfamilie ISO 19101-19135

[24] Jedes Objekt besteht aus Daten und zugehörigen Zugriffsmethoden. Alle weiteren Funktionalitäten bleiben der Außenwelt verborgen. Man spricht von Kapselung. Objekte können zu Klassen zusammengefasst werden und erben gegebenenfalls die Klasseneigenschaften einer übergeordneten Klasse (Vererbung).
[25] (Vgl. Lange 2002, S.106, 303)
[26] (Vgl. Walter 1998, Lecture 1)
[27] (Vgl. Fornefeld, Oefinger 2001, Fornefeld, Oefinger, Rausch 2001)
[28] (Vgl. Fornefeld, Oefinger 2001, S.1)
[29] weitere Gremien in Lange (2002, S.210ff) und World Standard Services Network (2004)

Eine Zusammenarbeit von OGC und TC 211 existiert. Einige OGC Standards bilden die Grundlage für den praktischen Teil dieser Arbeit. Für das Verständnis dieser theoretischen Papiere ist ein Studium des Standardisierungsprozesses hilfreich. Der Aufbau und das Zustandekommen von OGC-Spezifikationen soll im nächsten Abschnitt beschrieben werden.

Verfahrensweise

Das OGC beschreibt den Prozess ausführlich in ”Abstract Specification Overview”[30]. Zu bestimmten Kernthemen werden Special Interest Groups oder Working Groups gebildet, in denen abstrakte Spezifikationen erarbeitet werden. Die abstrakten Spezifikationen beschreiben zwei konzeptionelle Modelle, die die Grundlage für Implementierungsspezifikationen bilden.

Das essentielle Modell ist eine Beschreibung, wie die reale Welt funktioniert oder funktionieren sollte. Sinn und Zweck ist es, einen Bezug zwischen Spezifikation und realer Welt herzustellen.

Das abstrakte Modell , der eigentliche Inhalt der Spezifikation, ist eine abstrahierte Beschreibung des letztendlichen Systems in einer implementierungsneutralen Art und Weise. Es ist somit ein Kompromiss zwischen essentiellem Modell (realer Welt) und den Gemeinsamkeiten verschiedener Softwarelösungen für ähnliche Aufgaben (z.B. Datenspeicherung - Datenbanken, Dateien).

Wenn die abstrakte Spezifikation fertiggestellt ist, wird ein Request for Proposals (RFP - Anfrage von Vorschlägen) herausgegeben. Dieser Prozess ist öffentlich, so dass eine Vielzahl von interessierten Herstellern, und *nicht* nur Gremiumsmitglieder, Vorschläge einreichen können. Die Antworten fließen in einer Implementierungsspezifikation zusammen, die aufgabenspezifisch aber plattformneutral (Betriebsystem, Hardware) ist. Nach Bewilligung durch das Gremium wird die Implementierungsspezifikation in die offizielle Sammlung integriert.
Spezifikationen von besonderer Wichtigkeit für den praktischen Teil sind Simple Features Specification und Coordinate Transformation Services Specification, beide vom OGC, auf die nun näher eingegangen werden soll.

2.2.3 Simple Feature Specification

Die Simple Feature Specification Spezifikation basiert auf dem OGC Reference Modell[31], ist aber in der Funktionalität eingeschränkt. Simple Features können nur eine Geometrie und keine weiteren Features enthalten.

[30] (Vgl. OGC 2004, The OpenGIS Abstract Specification Overview Version 4 1999)
[31] (Vgl. OGC 2004, OpenGIS Reference Model Version 0.1.2, 2003, S.8ff)

Feature, Geometrie, Featurecollection

Der Ansatz zur Darstellung von Geodaten ist das Geographic Feature (engl. wichtiger Teil, charakteristische Eigenschaft). Ein Feature ist die Abstraktion eines Phänomens[32] in der realen Welt. Ein Geographisches Feature ist demnach eine wahrnehmbare Erscheinung mit einer Position relativ zur Erde. Eine Sammlung von geographischen Features kann als Abstraktion der Welt angesehen werden. Geographische Features werden zweiteilig abgebildet, als *Feature Instance (dt. Instanz)* und *Feature Type (dt. Typ)* (Abbildung 2.6).

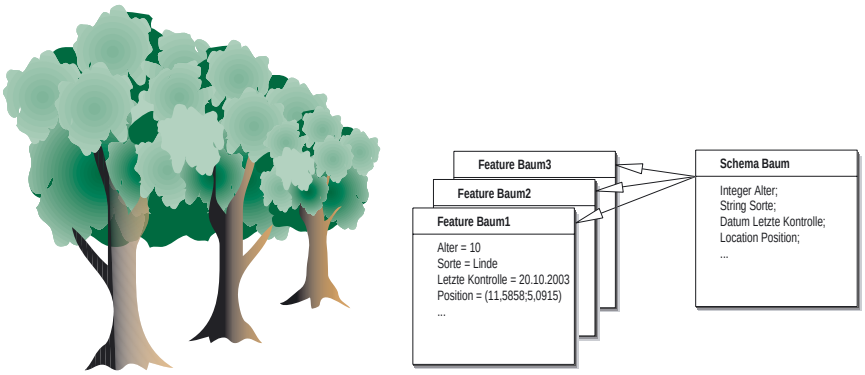


Abbildung 2.6: Featureinstanzen und das dazugehörige Schema

Eine Geofeature-Instanz repräsentiert das einzelne Phänomen mit seinen Attributen, wie Ort und Zeit. Individuelle Instanzen sind als Klassen mit gemeinsamen Attributen gruppierbar - die Feature Types, auch Feature Schema genannt. Feature Types bilden die Vorlage zur Erzeugung von Instanzen (Abbildung 2.6). Die Beschreibung des Aufbaus (Metamodell) von Feature Types wird *General Feature Modell* genannt. Obwohl das OGC noch kein General Feature Modell beschlossen hat, werden für die bisherigen Spezifikationen Anregungen aus der ISO-Entwurf 19109 (Geographic Information) und dem Datenmodell der Geographic Markup Language (GML) verwendet.

Event Ein Event (dt. Ereignis) ist ein spezielles Geofeature in einem gegebenen zeitlichen Gültigkeitsbereich.

Attribute Jedes Geofeature kann mehrere Eigenschaften haben, die Operationen, Attribute oder Verknüpfungen sein können. Es soll möglich sein, einem Geofeature mehrere verschiedene Geometrien oder räumliche Daten zuzuweisen.

[32] mit den Sinnen wahrnehmbare Erscheinung

Das Geofeature sollte nicht als grafische Abbildung eines Phänomens verstanden werden. Viel eher ist es ein Objekt mit Gültigkeit in einem bestimmten Bereich und Eigenschaften. Geometrien sind Eigenschaften.

Geometrie Eine Geometrie (im Sinne dieser Spezifikation) ist eine geometrische Figur, deren Koordinaten in einem geographischen Bezugssystem (Referenzkoordinatensystem) definiert sind. Auch geographische Geometrien werden in einem Metamodell spezifiziert. In dem Modell sind grundsätzliche Klassen definiert (z.B. Punkt, Linie, Polygon). Das OGC hat dieses Metamodell von der ISO 19107 (Geographic Information - Spatial Schema) Spezifikation übernommen.

Feature Collection Komplexe Geofeatures können als Collection (Sammlung) existieren. Das heißt, einzelne Attribute können wieder Geofeatures sein und sind über Beziehungen mit dem *elterlichen* Geofeature verknüpft. Außerdem können Geofeatures mit gemeinsamen Attributen eine Feature Collection bilden, wobei die Referenz hier ein wenig ungenau ist.

Continuous und Discrete Features, Coverage

Geographische Phänomene lassen sich grob in zwei Kategorien unterteilen, *discrete* (dt. diskret, separat) und *continuous* (dt. andauernd / fortlaufend).

Diskrete Phänomene haben definierbare Grenzen oder festlegbare räumliche Ausbreitung (z.B. Gebäude, Felsen etc.).

Andauernde Phänomene variieren in ihrer Grenze und haben keine festlegbare Ausbreitung. Beispiel dafür wären Temperaturen und Bodenzusammensetzung. Andauernde Phänomene sind somit nur für bestimmte Orte oder Zeitangaben gültig.

Diese beiden Arten schließen sich nicht gegenseitig aus. So kann ein Fluss für einen gegebenen Zeitpunkt hinsichtlich seines Verlaufes als diskretes Phänomen betrachtet werden, aber genauso gut als andauerndes, da sich Wasserqualität und Flussrate von Punkt zu Punkt unterscheiden.

Vektordaten repräsentieren diskrete Features. Die räumlichen Eigenschaften werden durch eine oder mehrere geometrische Formen wiedergespiegelt. Andere Eigenschaften sind durch weitere Attribute angegeben. Normalerweise hat ein Geofeature nur einen Satz von Attributen.

Rasterdaten geben komplexe oder mehrere Phänomene zugleich wieder, die sich über einen Raum erstrecken. Je nach Auswertungsmethode (z.B. von

Farbtönen oder Mustern) sind andere Interpretationen möglich. Rasterdaten sind andauernde Features über den Raum.

Coverage Das OGC benutzt den Begriff *Coverage* (dt. Abdeckung), um die Zuweisung von gleichen Attributen zu einzelnen Positionen zu einem bestimmten Zeitpunkt zu beschreiben. Kennzeichnend ist die Eigenschaft, auf Anforderung Attribute für eine bestimmte Position innerhalb der Domain (dt. Gebiet) zu erhalten. Eine Abdeckung können mehrere Geofeatures sein, die ein Gebiet komplett abdecken, also für jede Position der Fläche Attribute zurückgeben. Desweiteren können ein Netz von Werten und mathematische Funktionen Abdeckungen sein. Rasterdaten, in denen z.B. mathematische Funktionen einen See erkennen und je nach Anfrage Attribute des Sees oder des umliegenden Waldes zurückgegeben, sind normalerweise Coverages.

2.2.4 Coordinate Transformation Services (CTS) Specification

Die *CTS-Spezifikation*[33] beschreibt *Koordinatenumrechnung* (Coordinate Conversion) und *Koordinatentransformation* (Coordinate Transformation) zwischen verschiedenen Bezugssystemen. Dadurch ist es möglich Geodaten aus verschiedenen Quellen vergleichbar zu machen oder für die weitere Bearbeitung und Auswertung miteinander zu verbinden.

Die Koordinatenumrechnung ist eine mathematische Operation auf den Koordinaten, welche das Datum (Bezugssystemskennzeichnung) nicht verändert. Bekanntes Beispiel einer Koordinatenumrechnung ist die Umrechnung von Koordinaten aus einem geodätischen Koordinatensystem in ein planares zweidimensionales Bezugssystem. Dabei werden Länge und Breite konvertiert und in eine projizierte Karte (engl. Map-Projection) übernommen. Die Parameter für die Umrechnungen sind definiert.

Eine Koordinatentransformation hingegen verändert das Datum der Geodaten. Die Parameter dafür werden empirisch aus einer vergleichenden Betrachtung der beiden Bezugssysteme gewonnen. Über spezielle Algorithmen werden Angaben zur Ungenauigkeit dieser Transformation errechnet und sollten vor einer Transformation für den Verwendungszweck in Betracht gezogen werden. Durch die stochastische[34] Natur der Transformationsbestimmung kann es passieren, dass unterschiedliche Ergebnisse für dieselbe Transformation existieren.

[33] (Vgl. OGC 2004, OpenGIS Implementation Specification: Coordinate Transformation Services v1.0 2001)

[34] zufallsabhängig, von Stochastik - aus dem Griechischen, statistische Untersuchung zufallsabhängiger Erscheinungen

Das OGC hat inzwischen eine CTS-Implementierungs-Spezifikation beschlossen, welche in wesentlichen Punkten von dem alten Spezifikationsentwurf vom Dezember 2001 abweicht. Es werden Anstrengungen unternommen die Abstrakte Spezifikation anzupassen[35]. Dementsprechend sind die folgenden Informationen aus der CTS-Implementierungs-Spezifikation entnommen.
Die CTS Implementierungs-Spezifikation enthält detaillierte Schnittstellenbeschreibungen für die Koordinatentransformation und zusätzlich für allgemeine Positionierung und (Referenz-)Koordinatensysteme. Um die Beschränkung der Anzahl der möglichen Dimensionen von Koordinatensystemen aufzuheben und Referenzkoordinatensysteme frei definierbar zu machen, sind diese schon in der Simple Features Specification definierten Teile neu spezifiziert worden. Die neue Spezifikation ist größtenteils rückwärts kompatibel, so dass die Definitionen auch mit den alten Koordinatensystemen funktionieren sollten. Zukünftig sollen die neuen Schnittstellen die alten ersetzen. Implementierungen der Interfaces für Java, Com und Corba sind verfügbar.
Viele Objekte der CTS-Implementierung können per WKT oder XML repräsentiert werden. Die (Zwischen-)Speicherung in Datenbanken und Dateien wird dadurch erheblich erleichtert. Ausführliche Formatierungsvorgaben für Koordinatensysteme (Datumsangaben) und mathematische Transformationen im WKT- oder XML-Format sind in der Spezifikation enthalten.
Transformationsobjekte werden über ein Factory-Objekt erzeugt. Nach Angabe von Quell- und Zielbezugssystem wird ein Transformationsobjekt erzeugt. Falls keine direkte Transformation möglich ist (Abschnitt 2.1.5 auf Seite 10), kann dieser Prozess fehlschlagen. Dem Transformationsobjekt kann nun eine Menge von Koordinaten zur Transformation übergeben werden. Ein Transformationsobjekt kann Auskunft über die Anzahl der Dimensionen von Quell- und Zielkoordinatensystem geben und eine inverse Transformation von sich selbst erzeugen. Quell- und Zielkoordinatensystem kennt die Transformation nicht mehr.

2.3 Satellitengestützte Positionsbestimmung

In der Gegenwart sind drei verschiedene Systeme zu Positionsbestimmung und damit auch zu Navigation auf der Erde im Gespräch. Real existierend und funktionsfähig sind davon das amerikanische GPS und das russische GLONASS. In der Planung befindet sich das europäische Galileo. Im Folgenden soll auf jedes dieser Systeme eingegangen werden, wobei mit dem für die Realisierung der Anwendung benutzten und weltweit (kommerziell) etablierten GPS begonnen wird.

[35] (Vgl. OGC 2004, OGC Reference, S.19)

Es wird versucht eine Einführung in die Konzeption und Technologie der Systeme zu geben. Gegebenenfalls wird auf weiterführende Information verwiesen.

2.3.1 GPS - Global Positioning System

Wirtschaftlich interessant wurde GPS circa 1984, als die ersten Pläne zur Freigabe für zivile Zwecke bekannt wurden. Für die Vermessung von Landflächen konnte mit der begrenzten Anzahl an bis dahin vorhandenen Satelliten gearbeitet werden und die Einsparungen waren enorm[36]. Es folgten Anwendungen im Luft- und Straßenverkehr. Die frühe Nutzung im kommerziellen Bereich (z.B. professionelle Flottennavigation und -überwachung) dringt jetzt auch in private Bereiche vor. Eine aktuelle Entwicklung sind Fahrzeugnavigationssysteme die TMC (Traffic Message Channels - Verkehrsnachrichten) der lokalen Rundfunkanstalten auswerten. So geht die Gesellschaft für Unterhaltungs- und Kommunikationselektronik (gfu) davon aus, dass im Jahr 2003 mehr als 900.000 Navigationsanlagen verkauft werden[37].

Geschichte

Auf Initiative des Departement of Defense (DoD, Verteidigungsministerium) der USA wurde Anfang der 60er Jahre nach einem präzisen Waffenleitsystem und eine Alternative zur stattfindenden parallelen Entwicklung von verschiedenen Navigationssystemen in der Armee gesucht. Die Idee war ein globales, allwettertaugliches, dauerverfügbares und sehr genaues System zur Positionsbestimmung und Navigation zu schaffen. Durch die Eignung des Systems für ein breites Spektrum an Einsatzmöglichkeiten plante das DoD die Einsparung von Finanzen, die momentan und zukünftig für Speziallösungen einzelner Abteilungen und Heere gebunden waren.
Zwei Programme der U.S. Navy (dt. Marine), namentlich Transit und Timation, und das U.S. Airforce (dt. Luftwaffe) Programm 621B ebneten den Weg für das heutige NAVSTAR (Navigation Satellite Timing and Ranging - die militärinterne Bezeichnung für das GPS)[38].

Transit war das erste funktionierende satellitenbasierte Navigationssystem der Welt. Sieben Satelliten sendeten Radiosignale aus einem Polarorbit in relativ geringer Höhe. Mehrere Bodenstationen empfingen diese Signale und

[36] (Vgl. Page, Frost, Lachow, Frelinger, Fossum, Wassem, Pinto 1995, S.248)
[37] (Vgl. Golem News 2003)
[38] (Vgl. Page et al. 1995, S.238)

berechneten die Bahnen der Satelliten, indem die Dopplerverschiebung[39] ausgewertet wurde. Interessanterweise stammte diese Idee von Wissenschaftlern, die den russischen Satelliten Sputnik beobachteten. Diese stellten fest, dass durch Messung der Dopplerverschiebung an verschiedenen Punkten, der Orbit des Satelliten bestimmt werden konnte. Folgerichtig wurde angenommen, dass eine Positionsbestimmung auf der Erdoberfläche anhand mehrerer Satelliten möglich sei.

Timation , der zweite Vorfahr von GPS, basierte auf Satelliten mit Quarz- und Atomuhren an Bord. Während die Quarzuhren noch zu instabil waren, erzielte man mit den Atomuhren (Rubidium und Cäsium) wesentliche Erfolge. Mit synchron laufenden Uhren in Monitorstationen und Satelliten ließen sich die Positionen der Satelliten sehr genau bestimmen. Tatsächlich wurden die letzten beiden Timation-Satelliten als Prototypen für GPS genutzt.

621B basierte auch auf präzisen Uhren. Für Tests wurden Sender auf dem Boden und in Ballons eingesetzt. Die Benutzung von Pseudorandom-Noise (PRN) codierten Signalen stellte sich als besonders stabil gegen Störversuche heraus und ermöglichte die Nutzung einer einzigen Frequenz für mehrere Sender ohne gegenseitige Störung und war in der Lage benötigte Daten wie Uhrzeit und Position des Satelliten zu transportieren.

Eine 1973 neu gegründete Abteilung des DOD, Defense Navigation Satellite System (DNSS), sollte die weitere Entwicklung leiten. In Zusammenarbeit mit allen militärischen Abteilungen sollten die vorhandenen Systeme evaluiert und zu einer Lösung gebündelt werden. 1995 wurde GPS offiziell vom US Air Force Space Command (Luftwaffen-Raumfahrt-Kommando) für voll funktionstüchtig erklärt, nachdem es schon 1990 im Golfkrieg erfolgreich eingesetzt wurde.

Aufbau des GPS

Das Global Positioning System ist in drei Segmente unterteilt:

Das Raumsegment besteht aus mindestens 24 Satelliten in einer Höhe von ca. 20.200 Kilometern, auf sechs Bahnen mit einer Umlaufzeit von 12 Stunden. Die Bahnen sind um 55° gegen die Äquatorebene geneigt, so dass für jeden Punkt auf der Erde, inklusive der Pole, drei bis maximal fünf Satelliten empfangbar sind[40]. Jeder Satellit trägt zwei Cäsium- und zwei Rubidium-Atomuhren.

[39] Folge des Dopplereffekts - Dopplereffekt, ”bezeichnet die Veränderung der Frequenz von Wellen jeder Art, wenn sich die Quelle und der Beobachter einander nähern oder voneinander entfernen.” Wikipedia (2004, Dopplereffekt)

[40] Ab etwa 5° über dem Horizont ist ein GPS Satellit empfangbar.

Das Kontrollsegment sind fünf Stationen (Pazifik - Hawaii, Kwajalein; Indischer Ozean - Diego Garcia; Atlantik - Ascencion; USA - Colorado Springs) rund um den Erdball, welche beständig die GPS-Signale aller Satelliten sammeln. Die Leitstation in den USA wertet die Daten aus und stellt Navigationsnachrichten an die Satelliten zusammen. Diese werden wiederum von den Stationen (außer Hawaii) an die Satelliten übermittelt.

Das Nutzersegment entspricht den Anwendungen des globalen Positionsbestimmungssystems. Herauszustellen ist die passive Natur dieses Segmentes. GPS-Empfänger sind reine Empfänger und senden keine Daten an die Satelliten. Grundsätzlich sind per GPS Positions-, Geschwindigkeit- und Zeitbestimmungen möglich.

Dienste

GPS-Satelliten senden dauerhaft auf zwei Frequenzen im Mikrowellenbereich. Insgesamt existieren fünf Frequenzen (L1-5[41]), von denen nur L1 und L2 GPS-Dienste zur Verfügung stellen[42]. L1 liefert sowohl Standard Positioning Service (SPS), als auch Precise Positioning Service (PPS), L2 nur den PPS. Diese zweite Frequenz wird gemeinhin als militärische Frequenz bezeichnet, was irreführend ist, da grundsätzlich PPS der militärisch genutzte Service ist und dieser beide Frequenzen benötigt.

SPS ist kostenfrei weltweit nutzbar und eignet sich insbesondere für Positionsmessungen und Zeitbestimmung. Es wird bei Positionsbestimmungen eine Genauigkeit von bis zu fünf Metern, durchschnittlich zehn Meter, erreicht. Diese Qualität kann jederzeit durch Aktivierung der Selective Availability (S/A - Ausgewählte Erreichbarkeit) drastisch reduziert werden. Tatsächlich war S/A bis zum 1.5.2000 aktiv und minimierte die Genauigkeit auf circa 100 Meter[43] (Abbildung 2.7 auf der nächsten Seite). Die USA behalten sich dieses Mittel für militärische *Notfälle* vor.

PPS ist zusätzlich zur Positions- und Zeitbestimmung für Geschwindigkeits- und Richtungsmessungen geeignet und erreicht eine absolute Mindestgenauigkeit von 20 Metern. Diese Genauigkeit wird über das Fehlen jeder S/A und die gemeinsame Auswertung von PPS auf L1 und L2 erreicht, um atmosphärische Störungen auszugleichen. Seit dem 31.1.1994 wird der PPS zu-

[41] namensgebend ist das L-Frequenzband
[42] GPS Satelliten führen Sensoren mit, die nukleare Explosionen erkennen und über eine gesonderte Frequenz melden.
[43] Selective Availability ist der Name einer Möglichkeit des US amerikanischen DOD die Genauigkeit des SPS zu minimieren. Dazu werden manipulierte, sprich verfälschte Positionsdaten und/oder Zeitangaben von den Satelliten per SPS gesendet.

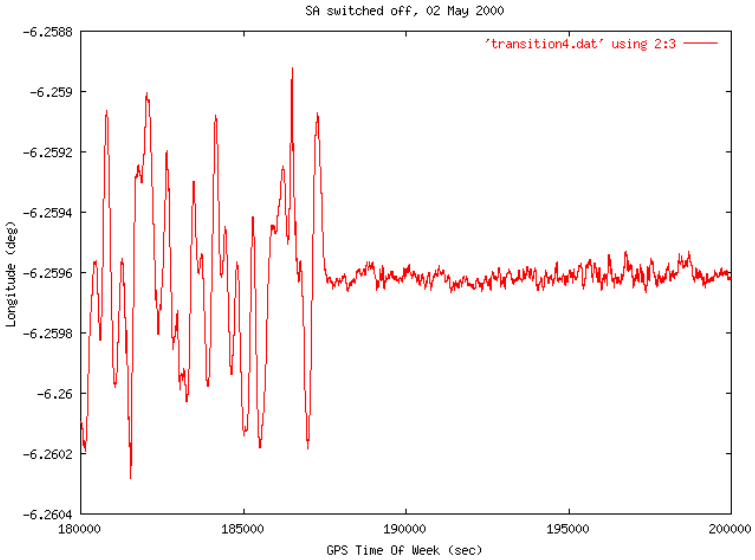


Abbildung 2.7: Deaktivierung der Selective Availability am 1.5.2000 von Desbonnet (2000)

grunde liegende P-Code mit einem unbekannten Schlüssel (W-Code) zusätzlich verschlüsselt. Dieses Vorgehen heißt Anti Spoofing (AS) und soll verhindern, dass gefälschte Satellitendaten in das System eingeschleust werden. Zusätzlich sorgt es dafür, dass nur durch die USA autorisierte Nutzer PPS nutzen können. Für die Nutzung wird ein spezielles AS-Modul benötigt.

Zur Zusammensetzung der Signale, deren Inhalte sowie weitere technische Details verweist der Autor auf U.S. Naval Observatory (2004), kowoma.de (2003) und Lange (2002, S.213).

Funktionsweise

Jeder GPS Satellit sendet dauerhaft folgende Daten:

- Almanach
    - Zusammenfassung aller verfügbaren Satelliten und deren Bahnen
    - Uhrenkorrekturfaktoren
    - atmosphärische Verzögerungsparameter
- Navigationsdaten
- Identifikation
- aktuelle Position
- aktuelle Zeit

Anhand dieser Daten berechnet der GPS-Empfänger die Zeitdifferenz zwischen dem Senden der Satellitennachricht und deren Empfang (aus Position und Zeitangabe). Da elektromagnetische Wellen sich mit Lichtgeschwindigkeit ausbreiten, kann durch Multiplikation eine Entfernung zum Satelliten berechnet werden.
Durch die Berechnung von drei Entfernungen zu drei verschiedenen Satelliten (unterscheidbar anhand der Identifikation) und unter Zuhilfenahme der Annahme, dass der GPS-Empfänger sich in der Nähe der Erdoberfläche befindet (also durch die Verwendung des Ellipsoiden), kann nun eine Position trianguliert werden. Dies wird *2D position fix* (zweidimensionale Positionsbestimmung) genannt. Wenn weitere Satelliten empfangbar sind, wird auf den Ellipsoiden verzichtet und stattdessen jede weitere Entfernungsmessung verrechnet. Dies wird *3D position fix* genannt, da zusätzlich die Höhe bestimmt werden kann.
Es existieren komplexe Mechanismen z.B. für das Initialisieren des Empfängers (Kalt- oder Warmstart), das Stellen der Uhr oder die Verwaltung des Almanachs. Ausführliche Informationen darüber sind auf der Internet-Seite der U.S. Coast Guard (2004, GPS-related Technical Documents) zu finden.

Fehler und DGPS

Als komplexes System beinhaltet GPS, abgesehen vom vorsätzlichen S/A, diverse Fehlerquellen. Dazu gehören (in Klammern die Auswirkung auf die Messgenauigkeit)[44]:

- Ungenauigkeiten der Satellitenuhren (1m)
- Schwankungen der Satellitenbahn (1m)
- Troposphärische Signalverzerrungen (1m)
- Ionosphärische Signalverzerrungen (bis zu 10m)
- Reflexionen der Signale (0,5m)

Der Anwender kann davon ausgehen, dass der GPS-Empfänger einige Probleme selbstständig erkennt und ausgleicht. So wird zum Beispiel durch *intelligente* Verarbeitung der Signale die Ungenauigkeit der empfängerinternen Uhr ausgeglichen (Stichwort - Pseudobereiche). Signalverzerrungen durch die Atmosphäre können je nach Signalstärke durch Kreuzkorrelation[45] behoben werden.
Doch insbesondere bei Aktivierung von S/A oder der Notwendigkeit exakter Messungen im Zentimeterbereich hat sich Differentielles GPS (DGPS) bewährt.

[44] nach Lange (2002, S.217)
[45] Kreuzzusammenhang - Signalpaare oder allgemeine Wertepaare, z.B. Meßreihen, werden durch Berechnung gegeneinander verschoben und miteinander verglichen. Ergebnis sind Korrelationskoeffizienten, die die Ähnlichkeit der Werteketten für vorgegebene Verschiebungen repräsentieren.

DGPS ist ein Fehlerkorrekturverfahren, dass auf dem Vergleich von Messergebnissen basiert. Benötigt werden eine oder mehrere Referenzmessstationen, deren Koordinaten exakt bekannt sind. Diese bestimmen dauerhaft die eigene Position per GPS und berechnen anhand der bekannten eigenen Position einen Messfehler. Bei Verrechnung dieses Messfehlers mit der mobilen Messung werden in der Regel absolute Genauigkeiten von zwei bis fünf Metern erreicht.
Zu unterscheiden sind bei diesem Differentialausgleich zwei Verfahren. Zum einen werden seit 1997 auf Langwelle Korrekturdaten von der Deutschen Telekom (ALF - Accurate Positioning by Low Frequency) zur Verfügung gestellt, die eine Korrektur in Echtzeit ermöglichen. Zum anderen gibt es auch die Möglichkeit, Messergebnisse in der Nachbearbeitung (z.B. von SAPOS[46]) bis zur Millimetergenauigkeit korrigieren zu lassen. Die Korrektur in Echtzeit wird Real-Time-Kinematik (RTK) GPS genannt, während die Nachbearbeitung als Kinematik-GPS bezeichnet wird [47].
Des weiteren können noch satellitenbasiertes Wide Area (WA)-DGPS und erdstationsbasiertes Local Area (LA)-DGPS unterschieden werden. So ist ALF ein LA-DGPS, da die Genauigkeit auf eine Region um die Referenzstation begrenzt ist. Die im nächsten Absatz erwähnten Satellite Based Augmentation Systems sind WA-DGPS.

Zukunft des zivilen GPS

Es ist geplant, zukünftig zwei weitere Dienste zur zivilen Nutzung zur Verfügung zu stellen. Diese sollen auf L2 und L5 platziert werden. Die Qualität des zivilen GPS soll damit ab 2005 erheblich verbessert werden. Für 2010 wird mit dem Start der dritten GPS-Generation gerechnet[48,49].
International werden momentan drei satellitengestützte WA-DGPS-Systeme entwickelt: *WAAS* (Wide Area Augmentation System - dt. Erweiterungssystem für einen großen Bereich) von den USA mit kanadischer Unterstützung, *EGNOS* (European Geostationary Navigation Overlay System) von der Europäischen Union, Europäischen Raumfahrt Behörde und EUROCONTROL und *MSAS* (Multi-Functional Satellite Augmentation System) von Japan und anderen asiatischen Ländern. Nähere Angaben zu diesen SBAS (Satellite Based Augmentation Systems[50]) sind auf der Webseite der SBAS Technical Interoperability Working Group (2002) (IWG) erhältlich.

[46] (Satellitenpositionierungsdienst der deutschen Landesvermessung 2004)
[47] (Vgl. Kaiser o.J.)
[48] (Vgl. Ashley 2003)
[49] (Vgl. NAVSTAR GPS Joint Office Program 2004)
[50] dt. Satellitengestützte Erweiterungssysteme

Antrieb für diese Initiativen, und insbesondere für die Zusammenarbeit (in der IWG) und letztendliche Kompatibilität der Systeme untereinander, ist der geplante Einsatz für die globale Flugsicherung. So sollen bei voller Funktionsfähigkeit von SBAS-CAT1-Anflüge (mind. 500m Sicht) sicher durchführbar sein.
SBAS wird die Genauigkeit und Zuverlässigkeit von GPS, im Fall von EGNOS auch GLONASS (siehe nächstes Kapitel), verbessern. Dafür werden zahlreiche Referenzstationen weltweit errichtet und in Betrieb genommen. Geplant sind 25 Stationen in den USA und zu Beginn 10, im Endausbau 34 in Europa und im Pazifikraum. Einige dieser Stationen werden Hauptstationen sein, welche Langzeitfehler der Satellitenpositionen, Kurz- und Langzeitfehler der Satellitenuhren, Ionosphären-Korrekturgitter[51] und Integritätsinformationen über die Satelliten aus den gemessenen Daten der Referenzstationen berechnen und innerhalb von sechs Sekunden über geostationäre Satelliten zur Verfügung stellen. SBAS-Satelliten senden auf derselben Frequenz wie ”echte” GPS-Satelliten ein ähnliches Signal. Damit können die SBAS-Satelliten zur Positionsbestimmung verwendet werden und verbessern gleichzeitig die Genauigkeit der Positionsbestimmung. Problematisch wird die Benutzung der geostationären Satelliten für bodengebundene Nutzer. Diese werden die dicht über dem Horizont (zwischen 15 und 35°) stehenden Satelliten schlecht empfangen können, wenn sie sich im Funkschatten von Gebäuden, Bäumen usw. befinden[52].
WAAS ist seit 2000 einsatzbereit, wenn auch noch nicht als vollständig einsatzbereit deklariert. Die Beschränkungen betreffen aber allein den Einsatz im Flugverkehr [53]. EGNOS ist in der Testphase und soll Mitte 2004 einsatzbereit sein [54]. MSAS soll Ende 2004 mit einem und Ende 2006 mit zwei Satelliten zur Verfügung stehen [55].

2.3.2 GLONASS

Wie GPS besteht GLONASS (Global Navigation Satellite System) aus 24 Satelliten, welche aber in 19.100 km Höhe positioniert sind. Die Satelliten sind auf drei Bahnen verteilt, welche um 120° zueinander versetzt sind. Für eine Erdumrundung braucht ein GLONASS-Satellit 11,25 Stunden. Jeder Satellit trägt drei Cäsium-Atomuhren und ist mit 64,8° zur Äquatorebene geneigt (inkliniert). Der

[51] Grundlage um für jedes Signal eines GPS-Satelliten, das zur Positionsberechnung verwendet wird, den Durchtrittspunkt (Pierce Point) durch die Ionosphäre zu bestimmen und die Signalverzögerung zu berechnen.
[52] (kowoma.de 2003, Was ist WAAS/EGNOS?)
[53] (Vgl. Trimble 2004, WAAS-FAQ)
[54] (Vgl. European Space Agency 2004, EGNOS-FAQ)
[55] (Vgl. International Civil Aviation Organisation 2004, Fifth Meeting of CNS/MET Sub-Group of APANPIRG, 2001)

erste Satellit wurde 1982 gestartet und seit 1993 ist das System offiziell in Betrieb.
Es werden ein Standard Precision (SP) und ein High Precision Signal (HP - Hochpräzision) gesendet, wobei der SP-Dienst seit 1988 (seinerzeit 10 Satelliten verfügbar) für zivile Nutzung freigegeben ist. Die Genauigkeit von SP liegt bei circa 60 Metern, ist aber schon mit 26 Metern gemessen worden [56]. Bei Benutzung nördlich von 50° nördlicher Breite ist GLONASS deutlich besser verfügbar als GPS. Laut GLONASS (2004) sind jederzeit fünf Satelliten verfügbar, was bedeutet, dass Höhenmessungen zu jeder Zeit möglich sein müssten.
Problematisch könnte die Benutzung eines Bezugsystems sein, das außerhalb Russlands kaum eine Rolle spielt und dementsprechend transformiert werden muss.
Ausführliche Informationen sind auf der Webseite für GLONASS vom russischen Verteidigungsministerium zu finden[57].

2.3.3 Galileo

Galileo soll das erste satellitengestützte Positionsbestimmungs- und Navigationssystem speziell für zivile Zwecke werden. Galileo soll in allen Bereichen des Transportwesens für Navigation, Verkehrs- und Flottenmanagement, Tracking-, Überwachungs- und Notfallsysteme zum Einsatz kommen. Die Souveränität Europas beim zukünftigen Verkehrsmanagement und der Telematikinfrastruktur ist die politische Triebfeder für ein drittes GPS, das im Nutzersegment kompatibel zu GPS und GLONASS werden soll.
Anfänglich durch europäische Infrastruktur- und Entwicklungsfonds sowie über das GalileoSat-Entwicklungs- und Validierungsprogramm der ESA finanziert, sind später zusätzliche Mittel aus dem privaten Sektor nötig. Diese sollen über ein Public-Private-Partnership-Modell beschafft werden. Für die Umsetzung werden Kosten von 3,2 bis 3,4 Milliarden Euro veranschlagt. 90 Milliarden Euro makroökonomischer Nutzen für Europa wird versprochen, welche über zusätzliche Ausrüstung, Verkäufe und Dienstleistungen während der Einführung des Service und in den ersten 15 Betriebsjahren erwartet werden.
Galileo ist momentan in der Entwicklung und soll 2005 in den Testbetrieb gehen und spätestens 2008 voll funktionsfähig sein. Mehr Informationen sind auf der Webseite der Galileo-Projektes[58] zu finden.

[56] (Vgl. National Oceanic and Athmospheric Administration 2004, Steve Dye, Navigation Satellites)
[57] (Vgl. GLONASS 2004)
[58] (Europäische Union 2004, Galileo Webseite)

2.4 Software - Kategorien und Lizenzen

Im praktischen Teil werden mehrere Softwarebibliotheken und ein Framework fremder Urheber verwendet. Da für die Nutzung normalerweise eine Lizenzvereinbarung[59] Vorraussetzung ist, beschäftigt sich dieser Abschnitt mit Softwarekategorien und deren spezifischen Lizenzmerkmalen. Besonderes Augenmerk wird dabei auf die Verfügbarkeit des Quellcodes und die Erlaubnis zur Modifikation und Weiterverbreitung gelegt. Die Aufzählung der Lizenzierungsformen erhebt keinen Anspruch auf Vollständigkeit, deckt aber alle wichtigen, am Markt vorhandenen Lizenzen ab. Manche Lizenzarten erscheinen sehr ähnlich, sind aber Abwandlungen mit Unterschieden in feinen aber *wichtigen* Details. Die Kombination von Lizenzmerkmalen z.B. kommerzielle freie Software oder proprietäre Shareware ist möglich, da sich deren Merkmale nicht gegenseitig ausschließen. Einen Überblick über die Verwandtschaft der Kategorien gibt Abbildung 2.8. Nach Klarstellung der Unterschiede wird kurz auf die Begriffe Copyleft, Non-Copyleft und Copyright eingegangen.

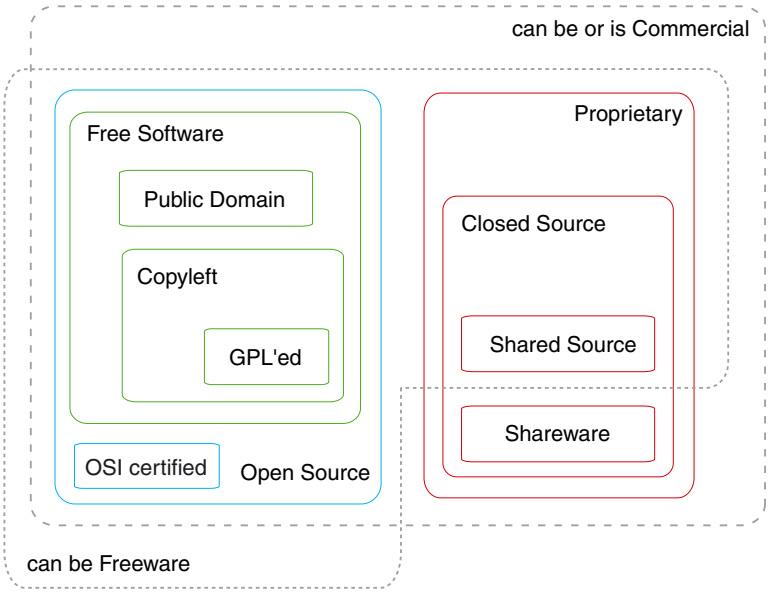


Abbildung 2.8: Softwarekategorien im Überblick

[59] Eine Lizenzvereinbarung legt die Nutzungsrechte des Anwenders und Verpflichtungen des Urhebers fest.

2.4.1 Public Domain

Übersetzbar als öffentliches Eigentum, bezeichnet Public Domain Software, deren Autor alle Rechte und Ansprüche an die Allgemeinheit abgetreten hat. Sie unterliegt also keiner Lizenz im ursprünglichen Sinne mehr (lizenzfrei) und kann dementsprechend beliebig verändert, verbreitet, verkauft oder zum Bestandteil einer neuen Lizenz erklärt werden. Übrigens ist es laut deutschem Urheberrecht, nicht möglich das Urheberrecht abzutreten[60]. So können einzig und allein exklusive Verwertungsrechte verkauft werden. Public Domain ist in den USA ein offizieller Rechtsbegriff des Urheberrechts und bedeutet exakt ”ohne Copyright”.

2.4.2 Freie Software

Die Erlaubnis der Nutzung, Vervielfältigung, Verteilung, Veränderung (unwichtig ob gratis oder gegen Gebühr) zeichnet diese Software aus. Das bedeutet insbesondere, dass der Quellcode zur Verfügung stehen muss. Das bedeutet auch, dass ein kommerzielles freies Softwareprodukt vom Käufer (un-)verändert weiterverkauft werden kann. Die Geschäftsmodelle mit freier Software basieren deshalb meist auf Mehrwertverkäufen. So verkaufen Linux-Distributoren die freie Software zusammen mit erweiterter Dokumentation, Service, anderer (auch proprietärer) Software u.a. rund um das freie Produkt.
Das Wort *frei* bezieht sich hier also auf die tatsächliche Freiheit mit der Software zu verfahren. So heißt es treffend auf der Webseite des GNU-Projekts:

> ”Free software is a matter of freedom, not price.”[61]

Die 1985 gegründete Free Software Foundation (FSF)[62] unterhält das GNU-Projekt, dessen Ziel es ist ein freies Unix-ähnliches Betriebssystem zu entwickeln. GNU/Linux ist z.B. aus diesem Engagement entstanden. Desweiteren wurden drei Lizenzen (GPL - General Public Lizenz, LGPL - Lesser General Public License, FDL - Free Documentation License) entwickelt, die von Urhebern kostenlos verwendet werden können, um ihr Produkt dauerhaft zu ’befreien’. FDL und GPL sind Copyleft Lizenzen. LGPL ist Non-Copyleft. Die Unterschiede werden im Abschnitt 2.4.9 auf Seite 31 näher betrachtet.
Wichtig am Konzept freier Software ist noch ein anderer Fakt: Dem Urheber werden keinerlei Auflagen gemacht (keine Garantieleistungen, keine exklusiven Nutzungsrechte etc.). Effektiv behält er weiterhin alle Urheberrechte am Produkt und

[60] ”§29 *Abs.1* Das Urheberrecht kann in Erfüllung einer Verfügung von Todes wegen oder an Miterben im Wege der Erbauseinandersetzung übertragen werden. *Abs.2* Im übrigen ist es nicht übertragbar.”(Urheber- und Verlagsrecht 2003)
[61] (GNU-Project 2004, Categories of Free and Non-Free Software)
[62] Webseite FSF

kann parallel das Produkt auch anders lizenziert vertreiben (z.B falls ein Kunde das Produkt weiterentwickeln, aber nicht frei machen möchte). Schwierig bis unmöglich ist dies bei Gruppenprojekten, da alle Urheber einer zweiten oder anderen Lizenzvereinbarung zustimmen müssen[63].

2.4.3 Open Source Software (OSS)

Oft verwendet als Synonym für freie Software bedeutet Open Source Software rechtlich doch nicht zwingend dasselbe. Einschränkungen im Detail für Urheber oder Nutzer in der Lizenzvereinbarung machen den Unterschied zur freien Software aus. Freie Software ist immer Open Source, aber nicht umgekehrt[64]. Ein guter Überblick zur Unterscheidung wird in O'Reilly (1999, Die Philosophie des GNU und die Pragmatik des Open Source) gegeben. Im allgemeinen Sprachgebrauch hat sich der Begriff OSS als Nachfolger für freie Software durchgesetzt. Dies sollte beachtet werden um Mißverständnisse zu vermeiden.
Die Führung der Open-Source-Bewegung hat die Open Source Initiative (OSI) übernommen. Die OSI, eine Non-Profit-Organisation, gegründet 1998 aufgrund der Veröffentlichung des Netscape-Quellcodes[65], versteht sich als Marketingwerkzeug für Open Source. Der Begriff freie Software erscheint der OSI zu vage um kommerziell erfolgreich zu sein[66]. Die Lizenzen und Einstellungen der FSF werden als zu engstirnig und hinderlich für die Verbreitung der Open-Source-Idee gesehen[67]. Es wird eine pragmatischere Einstellung zu Open Source propagiert. Potentielle Verwender sowie Hersteller sollen von den Vorteilen eines reinen Open-Source-Konzeptes überzeugt werden. Die Open Source Definition (OSD) und das OSI-Zertifikat für OSD-kompatible Lizenzen wurden als öffentlichkeitswirksame Maßnahmen entwickelt, um die Marke Open Source[68] zu stärken. Dementsprechend ist die OSD[69], zu der alle Open Source Lizenzen kompatibel sein müssen, weniger einschränkend als die GNU Lizenzen. Allerdings sind GNU GPL und GNU LPL auch OSI zertifiziert, da sie *unter Umständen* Non-Copyleft und damit OSD kompatibel sind. Eine weitere bekannte OSS-Lizenz ist die Common Public License (CPL) für IBMs Eclipse-Projekt[70].

[63] (Vgl. Korrespondenz 1, 2, 3, 4 auf Seite XVI ff)
[64] (Vgl. Free Software Foundation 2004, Relationship between the Free Software movement and Open Source movement)
[65] (Vgl. Open Source Initiative 2004, History, 15.1.2004)
[66] (Vgl. Open Source Initiative 2004, Why "Free" Software is too Ambiguous, 15.1.2004)
[67] (Vgl. Open Source Initiative 2004, How is "open source" related to "free software"?, 15.1.2004)
[68] Mit der Gründung der OSI wurde *Open Source* als Markenzeichen registriert.
[69] (Vgl. Open Source Initiative 2004, The Open Source Definition, 15.1.2004)
[70] (Vgl. IBM 2004, CPL FAQ, 15.1.2004)

2.4.4 Freeware

Nicht zu verwechseln mit freier Software, bezeichnet Freeware kostenlose proprietäre Software-Produkte. So bezieht sich das Wort *free* (engl. kostenlos, frei) auf den Preis. Veränderungen an der Software sind normalerweise nicht erlaubt oder schwierig, da der Quellcode fehlt. Bekannte Beispiele für Freeware sind der Microsoft Internet Explorer oder der Adobe Acrobat Reader. Zu beachten ist, dass Freeware heutzutage keine klare Definition mehr besitzt, da unter dem absatzkräftigenden Schlagwort Freeware vertriebene Produkte völlig unterschiedliche Lizenzvereinbarungen mit sich bringen können[71].

2.4.5 Shareware

Shareware leitet sich aus dem Wort share ab (engl. für teilen) und weist damit auf das wichtigste Kriterium dieser Lizenzart hin. Die freie (gemeint ist kostenlose) Verteilbarkeit, welche eine hohe Verbreitung generieren soll, zeichnet diese Software aus. Allerdings handelt es sich hier um proprietäre, kommerzielle Software[72]. Nach einem in der Lizenz festgehaltenen Testzeitraum (Evaluation Period) ist der Anwender verpflichtet, eine uneingeschränkte Lizenz zu erwerben. Durch die freie Verbreitung ist für die Urheberseite die Nutzung nicht nachvollziehbar. Darum wird der Zahlungsmoral oft durch kleinere Einschränkungen der Software nachgeholfen. Typisch sind störende Einblendungen, Funktionsverweigerung nach Ablauf der Testzeit oder Einschränkung von Speicherfunktionen. Winzip und Paintshop Pro sind bekannte Shareware-Produkte.

2.4.6 Kommerzielle Software

Die Erwirtschaftung von Gewinn, z.B. durch den Verkauf von Software, ist das Merkmal dieser Softwarekategorie. Kommerzielle Software muss keine proprietäre Software sein, ist es aber meist. Ausnahmen sind zum Beispiel verschiedene kommerzielle Distributionen freier Software (z.B. des Linux-Betriebssystems), bei denen das freie Produkt in Kombination mit ausführlicher Dokumentation, technischer Hilfe oder auch proprietärer Software (z.B. Yast) vertrieben wird. Für die FSF ist eine Trennung von kommerzieller und proprietärer Software von essentieller Bedeutung, da eine Begriffstrennung erst die Möglichkeit der kommerziellen Nutzung von copylefted Software in das gesellschaftliche Bewusstsein trägt[73].

[71] (Vgl. GNU-Project 2004, Categories of Free and Non-Free Software, Freeware)
[72] (Vgl. GNU-Project 2004, Categories of Free and Non-Free Software, Shareware)
[73] (Vgl. GNU-Project 2004, Categories of Free and Non-Free Software, Commercial Software)

2.4.7 Shared Source Software

Shared Source Software ist eine relativ junge Art der Lizenzierung. Der Quellcode wird nur für private oder firmeninterne, genau definierte (Test-)Zwecke offengelegt. Ein solcher Zweck ist z.B. die Implementierung von (Fremd-)Produkten in das dem Quellcode zugrundeliegende Produkt. Microsofts Windows CE Quellcode ist unter einer solchen Lizenz erhältlich. Nebenbei führt diese Lizenz zu einer, für den Lizenzgeber kostengünstigen, zweiten Qualitätskontrolle des Codes durch den Lizenznehmer[74].

2.4.8 Proprietäre Software

Vom englischen Adjektiv proprietary (dt. geschützt - im Sinne des Marken-/Eigentumsrechts) abgeleitet und schützt proprietäre Software explizit alle Rechte des Urhebers und auch sein Eigentum an dem Softwareprodukt. Dem Anwender wird gegen Zahlung der Lizenzgebühren lediglich ein eingeschränktes Nutzungsrecht (Lizenzierung) eingeräumt. Einschränkungen können zum Beispiel die Beschränkung der Anzahl von (Neu-)Installationen, der Nutzeranzahl, der Funktionen und vieles mehr sein. So ist zum Beispiel in der EULA (End User License Agreement - Endnutzer Lizenzvereinbarung) vom Microsoft Betriebssystem Windows XP geregelt, dass die Installation nur auf *einem* Computer zulässig ist. Zusätzlich ist der Käufer verpflichtet, das Produkt beim Hersteller zu registrieren, damit es vollständig funktioniert. Diese sogenannte Aktivierung ist beim Austausch von Hardwarekomponenten gegebenenfalls zu wiederholen.

2.4.9 Copyleft, Non-Copyleft, Copyright

Copyleft

Angeregt durch den asozialen Charakter des restriktiven Urheberrechts (Copyrights) prägte Richard Stallman (Gründer der Free Software Foundation) den Begriff Copyleft[75], als bewusst gegenteilig klingenden Begriff. Copyleft bedeutet aber keineswegs die Aufgabe aller Rechte à la Public Domain. Vielmehr handelt es sich um eine spezielle Form der Freien-Software-Lizenz, die keine weiteren Restriktionen zulässt. Software unter Copyleft bleibt somit immer freie Software, auch wenn sie modifiziert wurde. Die GNU GPL ist eine solche Lizenz.

[74] (Vgl. Open Source Initiative 2004, Tiemann, Michael: Speech at the O'Reilly Open Source Convention 2001, Esse Quam Videri, 15.1.2004)

[75] (Behlendorf, Bradner, Hamerly, McKusick, O'Reilly, Paquin, Perens, Raymond, Stallman, Tiemann, Torvalds, Vixie, Wall, Young 1999, Stallman, The GNU Operating System and the Free Software Movement, S.53ff)

Non-Copyleft

Non-Copyleft weist schlicht auf die Tatsache hin, dass eine Lizenz nicht den Copyleft-Schutzbedingungen unterliegt. Die Gnu Lesser Public Lizenz (LGPL) ist eine solche Lizenz. Non-Copyleft Software kann also jederzeit in proprietäre Software verwandelt werden.

Copyrighted Software

Copyrighted ist jede Software, deren Urheber sich Rechte nach dem Urheberrechtsgesetz (engl. Copyright Act) vorbehält, also nicht alle Nutzungsrechte erteilt (z.B. Weitergabe, Modifikation). Microsofts EULA's sind zum Beispiel solche Lizenzen.

2.4.10 Überblick

Abschliessend werden zur besseren Übersicht die Merkmale der unterschiedlichen Softwarekategorien einander direkt gegenübergestellt. Als Merkmale wurden *kostenlos* (kostenfreie Erhältlichkeit), *verteilbar* (Vervielfältigung und Weitergabe erlaubt), *offen* (Quellcode erhältlich) und *veränderbar* (Modifikation der Software erlaubt) gewählt.

Softwarekategorie	kostenlos	verteilbar	offen	veränderbar
Public Domain	x	x	teilweise	x
Freie Software	teilweise	x	x	x
Open Source S.	teilweise	teilweise	x	x
Freeware	x	teilweise		
Shareware		x		
Kommerzielle S.		selten	selten	selten
Shared Source			zweck-gebunden	
Proprietäre S.	teilweise			

Tabelle 2.1: Softwarekategorien und Merkmale

Die Tabelle 2.1 zeigt, daß insbesondere Public Domain, Freie Software und Open Source Software für die Wiederverwertung als Teil einer anderen Anwendung geeignet sind. Diese Kategorien können (müssen aber nicht) alle dafür nötigen Merkmale besitzen.

Kapitel 3

Analyse der Problemstellung

In diesem Kapitel sollen das Ziel der Arbeit und die daraus resultierende Problemstellung näher analysiert werden. Nach einer Erläuterung der Motivation für die Themenwahl wird der Begriff Mobiles GIS erklärt und dessen charakteristische Merkmale erarbeitet. Abschließend unterstreichen mögliche Einsatzszenarien die Vielseitigkeit der umzusetzenden Anwendung.

3.1 Vision der Anwendung ”Mobiles GIS”

Ziel der Arbeit ist die Entwicklung einer Anwendung für die ”Mobile Bearbeitung und Erstellung von Geodaten durch satellitengestützte Positionsbestimmung”. Die eingängige Bezeichnung hierfür lautet gemeinhin mobiles GIS und wird in Abschnitt 3.1.3 auf der nächsten Seite ausführlich definiert. Hauptmotivation für die Themenwahl ist das Fehlen einer solchen Lösung als freie Software oder besser ausgedrückt, die bisherige ausschließliche Existenz teurer proprietärer Speziallösungen[1]. Trotz umfangreicher Entwicklungen in einzelnen Bereichen der Geoinformatik fehlt eine mobile GIS Lösung, die auf das Konzept freie Software und offene, zukunftssichere Industriestandards zugleich setzt.

3.1.1 Warum Freie Software?

Für die Realisierung im Kapitel 4 sollen vorhandene Software-Komponenten wiederverwendet werden. Freie Software bietet die nötigen Eignungsmerkmale für die Wiederverwertung fremden Quellcodes. Die ausführliche Diskussion zu diesem Thema ist im Abschnitt 2.4.10 zusammengefasst. Zudem kommt die zumeist kos-

[1] z.B. ESRI’s ArcPad, FieldWorker.

tenfreie Erhältlichkeit dem Etat dieses Diplom-Projektes entgegen. Nachfolgend werden die Vorteile noch einmal ausformuliert:[2].

Freie Software ist wiederverwendbar und stabil. Die Verfügbarkeit des Quellcodes erlaubt einen synergetischer Effekt zwischen Nutzern und anderen Entwicklern. ”Das Rad muss nicht neu erfunden werden” - es können Funktionalitäten anderer freier Software direkt integriert werden. Noch besser ist die Anbindung freier Software, welche sich in der aktiven Entwicklung befindet. In der Folge wächst die Anwendung selbstständig, funktioniert aber auch zuverlässiger und effektiver.

Freie Software ist flexibel. Durch die Zugriffsmöglichkeit auf den Quellcode können Anwender Fehlerkorrekturen selbst vornehmen oder selbstständig in Auftrag geben. Das trifft auch auf die Entwicklung von Anpassungen und Erweiterungen zu.

Freie Software ist kostengünstig. Quellcode ist in der Regel kostenfrei erhältlich. Für Nutzer und (Weiter-)Entwickler entfällt somit gleichermaßen der Erwerb teurer Lizenzen. Dieser Vorteil relativiert sich etwas durch die laufenden Kosten freier Software. Eingeschränkte Dokumentation und Konfigurationsanleitungen können Schulungen und einen höheren Lernaufwand im Vergleich zum proprietären Produkt nach sich ziehen.

3.1.2 Warum Industriestandards?

Industriestandards sind in der Praxis bewährte, von mehreren etablierten Produkten am Markt eingesetzte Konzepte. Normalerweise schließen sich Hersteller zu Initiativen zusammen, um die gemeinsame Marktnische zu verbreitern (vgl. Abschnitt 2.2.2 auf Seite 13). Vorher inkompatible Produkte werden für den Nutzer leicht(er) kombinierbar. Daraus ergibt sich eine neue und größere Vielfalt an Einsatzmöglichkeiten. Der Wegfall von Konvertierungen und Anpassungen verschiedener Produkte entfällt. Der Einsatz der Produkte ist kostengünstiger und effizienter. Durch die breite Anwendung der Standards in aktuellen, aber auch zukünftigen Produkten, können standardisierte Produkte länger produktiv eingesetzt werden. Standardisierte Software ist zukunftssicher.

3.1.3 Mobiles GIS und Location Based Services

Anspruchsvolle GIS-Funktionalitäten, die über das reine Betrachten hinausgehen, sind durch die Entwicklung immer leistungsfähigerer Endgeräte inzwischen auch

[2] (Vgl. Hang, Hohenson 2003, S.34ff)

mobil realisierbar. Durch die Integration von Echtzeit-Positionsbestimmung (zumeist per GPS) können diese zu ortsbezogenen Diensten (Location Based Services) erweitert werden[3]. Eine Begriffsdefinition ist auf der Internetseite Trimble (2004)[4] zu finden.

> ”Mobile GIS is the use of geographic data in the field on mobile devices. It’s an evolution of how the enterprise database is used and managed within an organization. Mobile GIS integrates three essential components; global positioning system (GPS), rugged handheld computers, and GIS software. Bringing these technologies together makes the enterprise database directly accessible to field based personnel - whenever and wherever it is required.”

Mit Betonung auf ”rugged handheld computers” (dt. robuste tragbare Computer) und GPS-Empfängern (beides Trimble-Produkte), werden die drei essentiellen Bestandteile eines mobilen GIS aufgezählt:

1. GIS-Software
2. Positionsbestimmung
3. mobiles Endgerät

Somit kann ein mobiles GIS theoretisch alles, was ein statisches GIS auch kann. Praktisch ist der Funktionsumfang des GIS (Vgl. Abschnitt 2.2.1 auf Seite 10) nurmehr durch die Eigenschaften des Endgerätes eingeschränkt und natürlich von aktuellen Umgebungszuständen (z.B. sind GPS oder Internetzugang nicht überall verfügbar) abhängig. Gerätespezifische Hard- und Software setzen hier sehr unterschiedliche Grenzen, die für den spezifischen Einsatzzweck zu berücksichtigen sind. So unterscheiden sich Betriebssysteme oder Bildschirmgrößen erheblich zwischen mobilen Computermodellen. Wichtig und problematisch sind auch die (teilweise zu) kurzen Laufzeiten der Geräte[5].

3.1.4 Grundkonzept

Wie im vorangegangenen Abschnitt verdeutlicht, sollte ein mobiles GIS die wesentlichen Merkmale eines GIS aufweisen. Es muss also Geodaten erfassen, speichern, verwalten, aktualisieren, analysieren, modellieren und präsentieren können[6]. Desweiteren sollte es Positionsdaten von einem Empfangsgerät anzeigen

[3] (Vgl. Gislounge 2004, Mobile and Field GIS)
[4] marktdominanter Hersteller von GPS und Mobile GIS Geräten (The Online Investor 2004, James Hale, Trimble Navigation, GPS Era Comes on Strong)
[5] (Christl, Rothstein: Mobile GIS-Lösungen - ein Anwendungsbeispiel aus dem öffentlichen Bereich in: Zipf, Strobl 2002)
[6] (Vgl. Abschnitt 2.2.1 auf Seite 10)

und verarbeiten. Aus den praktischen Beispielen lassen sich gemeinsame Funktionen für die mobile GIS-Software eingrenzen. Ein generalisierter Ablauf der Nutzung kann wie in Abbildung 3.1 beschrieben werden.

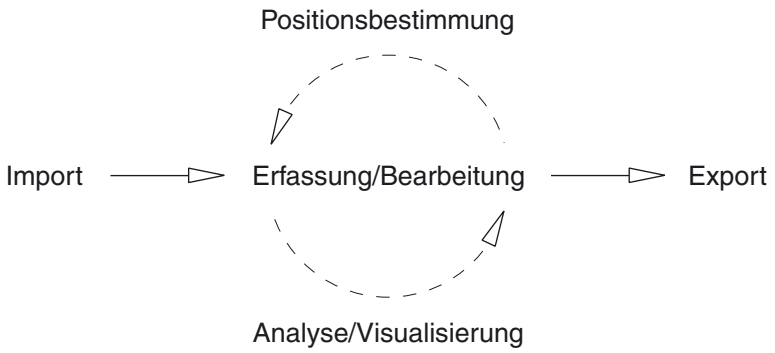


Abbildung 3.1: Allgemeines Nutzungsschema eines mobilen GIS

Vorhandene Geodaten oder zumindest definierte Strukturen (z.B. das Datenbankschema) eines vorhandenen Systems werden in das mobile GIS geladen. Die Lokalisierung des Einsatzortes erfolgt anhand eventuell vorhandener Geodaten (z.B. Stadtplan) und die gleichzeitige Auswertung der Positionsdaten vom Positionsempfänger. Vor Ort wird die aktuelle Position zur Erstellung von neuen Einträgen oder das Auffinden bereits eingetragener Geoobjekte benutzt. Die Attribute der Geoobjekte werden auf den vorgefundenen aktuellen Stand gebracht. Schlussendlich werden die Daten vom mobilen GIS wieder in das Hauptsystem oder andere Systeme zur Weiterverarbeitung exportiert. Je nach Einsatzzweck können vor Ort noch Analysefähigkeiten zum Einsatz kommen (zum Beispiel eine Kontrolle der Gültigkeit von Datensätzen).

3.1.5 Die Vorteile

Generell können drei Vorteile für den Einsatz von mobilen GIS festgestellt werden[7].

Beseitigung von Medienbrüchen Ein generelles Argument, das für die Einführung von Informationstechnologie (IT) verwendet wird. Die Übertragung von Informationen von oder auf andere Medien entfällt durch den konsequenten Einsatz von IT und digitalen Daten.

Effizienzsteigerung Die Minimierung des Aufwandes zur Erreichung des Zieles ist sicherlich auch als Folge des ersten Punktes anzuführen. Doch durch

[7] (Vgl. Ranzinger: Einsatz mobiler GIS bei Energieversorgungsunternehmen in den Bereichen Instandhaltung und Störungsmanagement, in: Zipf, Strobl 2002)

den Einsatz der Positionsbestimmung sind weitere Einsparungen möglich (z.B. Zeit zum Auffinden von Einsatzorten).

Zeitnahe Dokumentation Der Abstand zwischen verändernden Tätigkeiten und der Dokumentation wird verkürzt. Durch intelligente Formulare werden auch Details nicht vergessen.

3.2 Konkrete Verwendungsmöglichkeiten

Abschließend sollen drei Möglichkeiten des Einsatzes vorgestellt werden. Die vielfältigen Verwendungsmöglichkeiten des mobilen GIS sollen demonstriert und Abläufe der Nutzung aufgezeigt werden.

3.2.1 Baumkataster

Viele größere Städte und Kommunen führen ein Baumkataster[8]. In diesem sind alle städtischen Bäume mit Information zu Position, Art, Alter u.ä. enthalten. Ferner werden im Baumkataster die in regelmäßigen Abständen durchzuführenden Baumkontrollen und die sich daraus ergebenden Pflegearbeiten an den Bäumen dokumentiert. Jeder Baumbesitzer mit einer Vielzahl von Bäumen im öffentlichen Raum sollte ein Baumkataster führen. Nach deutschem Recht[9] ist der Besitzer eines Baumes schadensersatzpflichtig, sollte er ”fahrlässig Rechte eines anderen” verletzen.
Für die Pflege dieser Datenbank bietet sich der Einsatz eines mobilen GIS an. Der Ablauf könnte kurz zusammengefasst so aussehen:
Ein Mitarbeiter nimmt das mobile Endgerät und lädt die Baumdaten eines bestimmten Bereiches (z.B. Stadtteil). Ausgestattet mit diesem Auszug aktueller Daten bewegt er sich zum Einsatzgebiet und aktiviert dort die Positionsbestimmung. Mit dem Wissen um die aktuelle Position zeigt das mobile GIS die Position des Mitarbeiters im Kontext des Einsatzgebietes und erleichtert das Auffinden der schon eingetragenen Bäume. An einem Baum angelangt, wird der Datensatz dieses Baumes zur Bearbeitung geöffnet und papierlos auf den aktuellen Stand gebracht. Das Anlegen eines neuen Datensatzes ist problemlos möglich. Der Mitarbeiter begibt sich zu dem Zielbaum und aktiviert die Erfassung. Durch Auswertung der aktuellen Position und die automatische Erteilung einer Katasternummer nach standardisiertem Verfahren wird ein Datensatz erstellt. Komfortabel können nun benötigte Schlüsseldaten zum Baum eingeben werden. Zurück in der Zentrale

[8] Kataster - amtliches Verzeichnis
[9] (Vgl. Bürgerliches Gesetzbuch 2003, §823)

werden die neuen Daten bei Bedarf kontrolliert und (wieder) in der Hauptdatenbank gespeichert.
Denkbar ist eine Erweiterung der Idee durch Kombination mit einem Internetzugriff. Hier wird es möglich, den Schritt des Im/Exports zu übergehen und direkt auf der Datenbank zu arbeiten. Proprietäre und teilweise auch OGC-Standard-kompatible Baumkatasterlösungen gibt es bereits von verschiedenen Herstellern[10].

3.2.2 Ver- und Entsorgungsunternehmen / Kommunikationsanbieter

Unternehmen in diesen Bereichen besitzen Leitungen, Schaltstationen und andere Entitäten außerhalb des Betriebsgeländes. Alle diese Teile arbeiten zusammen und bilden in der Summe das Produkt oder die Dienstleistung des Unternehmens. Sollte ein Teil gestört sein, hat dies unter Umständen Auswirkungen auf eine Vielzahl von Kunden. Regelmäßige Begehung der Betriebsmittel sind also zur Einhaltung von Qualitäts- und Sicherheitsstandards notwendig. Ein Prüfprozess, der in der Regel zeitaufwendig und arbeitsintensiv ist. Nach der Kontrolle der Einzelteile werden die Ergebnisse auf Papier festgehalten und nachträglich im Büro in die zentrale Datenbank übernommen[11]. Das schnelle Auffinden von Betriebsmitteln und die vollständige zeitnahe Dokumentation sind bei Einsatz eines mobilen GIS garantierbar. Durch die einheitliche Datenstruktur im Außeneinsatz und im Büro[12] werden die Kommunikationswege optimiert. Die Gesamtdokumentation erscheint den Mitarbeitern transparenter und leichter verständlich. Für Störfälle kann ein mobiles GIS um die Funktionalitäten des unternehmenseigenen Störungsmanagements erweitert werden. Dazu gehört zum Beispiel die Bereitstellung von Analysefunktionen (Netzwerkverfolgung, Ermittlung betroffener Anschlüsse u.s.w.). Mit diesen Funktionen und (Status-)Daten aus der zentralen Datenbank wird die Entstörung wesentlich erleichtert. Die gewohnte Umgebung mit Abfrage- und Analysefunktionen steht nun Außendienstmitarbeitern zur Verfügung.

3.2.3 Landwirtschaft

Ohne eine exakte Flächenberechnung einer bestellten Fläche kann die tatsächliche Leistung eines Feldes nicht ermittelt werden. Dies trifft auf eingebrachte Saat

[10] u.A. ESRI, AED-SICAD
[11] (Vgl. Ranzinger, Monika: Einsatz mobiler GIS bei Energieversorgungsunternehmen in den Bereichen Instandhaltung und Störungsmanagement in: Zipf, Strobl 2002, S.207)
[12] Ver- und Entsorger benutzen seit den Anfängen der GIS-Entwicklung spezialisierte GIS, sogenannte Netz-Informationssysteme (NIS) zur Auswertung und Planung ihrer Netze (Vgl. Stahl et al. 1998, NIS Netz-Informationssysteme)

und Düngung genauso zu, wie auf die Ertragsleistung. Die gemeinsamen Marktorganisationen der Europäische Union gewähren einen Grossteil Stützungszahlungen flächenbezogen[13]. Dazu müssen vom Landwirt die Größen und Nutzungsarten der bestellten Flächen bei der Beantragung genau angegeben werden. Die zuständigen Ämter des verantwortlichen Bundeslandes sind von der EU angehalten, diese Angaben stichprobenartig zu kontrollieren. Bisher werden von den Landwirten Karten der Katasterämter erworben, auf denen die erwünschten Parzellen markiert und deren Größe berechnet wird. Sollten bei diesen Berechnungen Fehler passieren, läuft der Landwirt Gefahr die Unterstützungen zu verlieren und eventuell Strafen zahlen zu müssen.
Ein mobiles GIS würde die zur Antragstellung notwendige Flächenverortung und -bemessung wesentlich vereinfachen. Durch langsames Abgehen/Abfahren der Grenzen eines zu vermessenden Feldes können Position und Größe gleichermaßen exakt bestimmt werden. Die resultierenden Daten können digital weiterverarbeitet werden und Übertragungsfehler vom Papier in die zentrale Datenbank werden vermieden. Umgekehrt ist dieses Verfahren genauso zur Kontrolle von Flächenangaben geeignet. Zur Erhöhung der Qualität der Positionsbestimmung sollte DGPS (Vgl. Abschnitt 2.3.1 auf Seite 23) zum Einsatz kommen. Die Vermessung kann von einem spezialisierten Unternehmen als Dienstleistung den Betrieben angeboten werden. Der Erwerb und die Einarbeitung der Landwirte erscheint bei entsprechend großen Flächen oder dem Einsatz von Precision Farming[14] sinnvoll.

[13] (Vgl. Europäische Union 2004, Organisation der Agrarmärkte)
[14] dt. Präzisions-Landwirtschaft - bezeichnet den Einsatz von mobiler GIS-Technologie in der Landwirtschaft. Düngemittel- oder Saatmengen können so, abhängig z.B. von der Bodenqualität, in verschiedenen Konzentrationen auf dem Feld ausgebracht werden.

Kapitel 4

Synthese

Dieses Kapitel dokumentiert schrittweise die Erstellung der mobilen GIS-Anwendung. Zu Beginn wird ein abstraktes Konzept erstellt, das die benötigte Funktionalität definiert. Nach einer Recherche vorhandener Ressourcen und deren Evaluation wird das abstrakte Konzept konkretisiert. Ergebnis ist ein konkretes Konzept, anhand dessen die tatsächliche Umsetzung erfolgt.

4.1 Abstrakter Konzeptentwurf

Die einzelnen Funktionen sollen nun konkretisiert und zu einem abstrakten Konzept für ein multifunktionales mobiles GIS ausformuliert werden. Ziel ist es, Anwendungen, wie die konkreten Beispiele (Abschnitt 3.2 auf Seite 37), aber auch ähnliche Aufgaben zu ermöglichen. Dementsprechend werden die benötigten Funktionen konkretisiert und in das Konzept übernommen. Gleichzeitig erfolgt eine Betrachtung der Ist-Situation (Welchen Rahmenbedingungen der realen Welt unterliegt die Funktion?) und der zu realisierende Funktionsumfang wird entsprechend angepasst. Zuletzt wird der Mobilitätscharakter definiert, welcher sich am angestrebten Funktionsumfang der GIS-Anwendung orientiert (z.B. aufwändige Berechnungen sind auf Handhelds nicht möglich).
Das Modell in Abbildung 4.1 auf der nächsten Seite stellt den Konzeptentwurf dar. Die einzelnen Funktionen sind für den Nutzer nur über definierte Schnittstellen einer grafischen Oberfläche erreichbar. Die grafische Oberfläche teilt sich in zwei Schnittstellenbereiche: Bedienelemente (Menüs, Knöpfe, etc.) und Visualisierungsbereich.

Die grafische Oberfläche enthält Bedienelemente, die den Nutzer Funktionen anstoßen lassen, z.B. ”Daten laden” oder ”Alles anzeigen”. Falls eine Parametereingabe notwendig ist, öffnet sich ein entsprechender Dialog.

Der Visualisierungsbereich hat eine Doppelfunktion. Alle geladenen Geodaten bzw. die repräsentierenden Geometrien werden hier dargestellt. Außerdem können die Geoobjekte bearbeitet werden. Änderungen an Attributen und Geometrie der Geoobjekte sind möglich und werden in Echtzeit in die Darstellung übernommen.

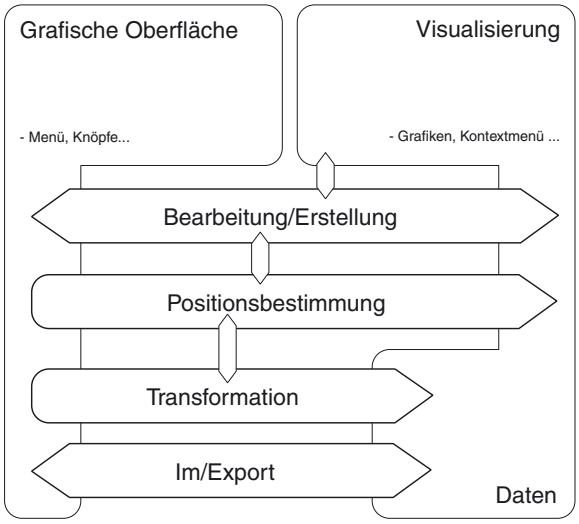


Abbildung 4.1: Funktionen und Kollaborationen

4.1.1 Im-/Export

Geodaten liegen normalerweise in unterschiedlichen Formaten, Genauigkeiten und Referenzsystemen vor. Diese für den Im-/Export von Daten problematischen Unterschiede haben folgende Ursachen:

Format: In der Geschichte der Geoinformatik haben sich verschiedene Firmen in unterschiedlichen Bereichen etabliert. Da diese Produkte keiner einheitlichen Normung unterlagen, existieren heute entsprechend viele Datenstrukturen und Dateiformate.

Genauigkeit: Wenn der Geocodierung von Objekten unterschiedliche Ziele zugrundeliegen, variieren die Resultate in Genauigkeit und Vollständigkeit der Metadaten. Für die Herstellung eines Stadtplanes werden z.B. andere Angaben und Genauigkeiten benötigt als für die Erfassung der städtischen Grünflächen und Spazierwege.

Referenzsystem: Je nach Herkunftsort der Geodaten werden verschiedene Referenzsysteme, entweder aus historischen Gründen oder aufgrund der höheren lokalen Genauigkeit, für das geocodierte Gebiet verwendet. In der Datenbank der European Petrol Survey Group[1] sind allein 2824 Referenzkoordinatensysteme aufgeführt, welche weltweit eingesetzt werden oder wurden.

Format und Referenzsystem sind Problemstellungen, die von der Zielanwendung zu lösen sind. Eine Fokussierung auf Datenformate, welche die OGC-Spezifikationen implementieren, erscheint sinnvoll. Diese Industriestandards sind mit dem Datenmodell des Shape-Dateiformates des Marktführers ESRI kompatibel. Desweiteren basieren die Implementierungen der führenden GIS-Datenbanken (Oracle, Informix, PostGIS) darauf. Für die gleichzeitige Bearbeitung von Geodaten aus verschiedenen Referenzsystemen muss eine Möglichkeit der Transformation zur Verfügung gestellt werden.
Die Genauigkeit kann nur vom Nutzer durch sorgfältige und passende Auswahl der Datenquellen zum Verwendungszweck gelöst werden.

4.1.2 Bearbeitung / Erstellung

Die Erstellung oder Bearbeitung eines Geodatensatzes sollte einfach sein, aber trotzdem zu einem vollständigen und gültigen Datensatz führen. Eine grafische Bearbeitung der Geometrie, welche das reale Phänomen repräsentiert, ist notwendig. Ähnlich einem Vektor- oder Grafikprogramm können die Geometrien mit dem Mauszeiger oder per Pen modifiziert und gestaltet werden. Die Bearbeitung nur über die Tastatur entspricht nicht dem Ansatz der Einfachheit. An der aktuellen Position sollten neue Geoobjekte durch eine Funktion (Positionsbestimmung) einfach zu erzeugen sein. Denkbar ist das automatische Zeichnen von Geometrien oder die Erzeugung auf ”Knopfdruck”. Desweiteren sind die Attribute des Geoobjektes an bestimmte Gültigkeitsbedingungen gebunden. Diese müssen schon während der Eingabe geprüft werden. Gegebenenfalls muss der Nutzer auf Fehler hingewiesen und zur Korrektur aufgefordert werden. Für die Bearbeitung mehrerer Grafiken mit Bezug zueinander hat sich das Ebenenmodell bewährt. Die Anordnung in einem Ebenenmanager wäre wünschenswert, wird aber vorerst als fakultativ eingestuft.

4.1.3 Positionsbestimmung

Für die Ermittlung der Position wird GPS als Satellitensystem ausgewählt. Dieses System erscheint in Anbetracht erhältlicher Hardware und Genauigkeit geeigneter

[1] (Vgl. European Petrol Survey Group 2004, EPSG Geodesy Parameters Database)

als GLONASS (vgl. Abschnitt 2.3.2 auf Seite 25). GPS-Empfangsgeräte werden zumeist über die serielle Schnittstelle[2] des Computers angeschlossen.
Trotz verschiedener herstellerspezifischer Protokolle zur Kommunikation mit dem Empfangsgerät hat sich das NMEA 0183 Protokoll etabliert. Dieses Protokoll ist Teil des NMEA 0183 Standards der namensgebenden National Marine Electronics Association, einer Industrievereinigung, die Standards und Qualität für maritime Elektronik fördert[3]. Ein neuer Standard NMEA 2000 ist verabschiedet, hat aber noch keine weite Verbreitung gefunden.
Im GPS-System wird das weltweite Referenzkoordinatensystem WGS 84 (benannt nach dem Datum zugrundeliegenden Ellipsoiden WGS 84) benutzt. Das heißt die Koordinatenangabe eines GPS-Empfängers liegt in diesem Bezugssystem vor. Später soll die aktuelle Position im Verhältnis zu vorhandenen Geodaten dargestellt und ausgewertet werden. Die Geodaten können in verschiedenen Bezugssystemen, andere als WGS 84, vorliegen. Teilweise sind GPS-Empfänger in der Lage Koordinatentransformationen zwischen fest definierten Bezugssystemen vorzunehmen. Dies trifft aber nur auf eine begrenzte Anzahl am Markt befindliche Modelle zu. Eine Koordinatentransformation der aktuellen Position in das Bezugssystem der Geodaten ist also notwendig.
Um eine Vielzahl an GPS-Empfangsgeräten mit dem mobilen GIS einsetzen zu können, sollten serielle Schnittstelle und das NMEA 0183 Protokoll unterstützt werden. Desweiteren muss eine Koordinatentransformation die Änderung des Referenzkoordinatensystems der aktuellen Position ermöglichen.

4.1.4 Transformation

Für die Funktionen Positionsbestimmung und Im-/Export wird eine Möglichkeit der Koordinatentransformation benötigt. Eine einmalige Realisierung dieser Funktion, welche allerdings von beiden anderen Funktionen genutzt wird, erscheint sinnvoll. Prinzipiell muss es dem Nutzer möglich sein pro Transformationsvorgang Quell- und Zielbezugssystem festzulegen. Dafür muss eine Liste mit Bezugssystemen zur Auswahl bereitstehen. Die einfache Neuaufnahme noch nicht eingetragener Bezugssysteme in die Liste muss möglich sein.

4.1.5 Mobile Einsatzfähigkeit

Die Spannbreite der mobilen Einsatzfähigkeit hängt direkt von der verwendbaren mobilen Computer-Hardware ab. Durch die ausgewählte Breite an Funktionalität

[2] emuliert bei Anschluss eines USB Gerätes
[3] (Vgl. National Marine Electronics Association 2004, Publications)

wird diese Spannbreite stark eingeschränkt. Tragbare und örtlich mobil einsetzbare computerbasierte Geräte[4] sind aktuell am Markt in sechs verschiedene Kategorien vorhanden:

1. Notebook
2. Tablet-PC
3. Handheld / PDA
4. Mobiltelefon (Handy)
5. Wearable Computer
6. ”single-purpose”-Systeme

Überblick

Diese Kategorien unterscheiden sich nicht nur in Bauform und Größe, sondern auch erheblich in ihrer Leistung und Ausstattung. Die Tabelle 4.1 auf der nächsten Seite[5] vergleicht wichtige Eigenschaften der verschiedenen Kategorien im Hinblick auf die Verwendung mit der konzipierten mobilen GIS Software. Wearable Computer werden aufgrund mangelnder Verbreitung, ”single-purpose”-Systeme wegen zu hoher Spezialisierung vom Autor als ungeeignet für das mobile GIS erachtet und nicht weiter betrachtet. In der Tabelle werden die potentiellen Zielplattformen anhand ihrer *Merkmale* (spezifische Eigenschaften), *Leistung* (die Rechenleistung), *Laufzeit* (im Betrieb ohne externe Stromversorgung), *OS* (Betriebssysteme) und *SDKs* (Softwareentwicklungs-Kits) verglichen.
Da ständig neue Konzepte auf den Markt kommen, sollten die Grenzen zwischen den Kategorien als fließend verstanden werden. Auch innerhalb einer Kategorie können, durch eine Vielzahl konkurrierender und parallel entwickelnder Anbieter in der Branche ”Mobile Computing”, die Ausstattungs- und Leistungsmerkmale stark voneinander abweichen (z.B. Subnotebook - Notebook mit geringerer Größe und Leistung). Es wurde deshalb versucht ein Mittelmaß der Eigenschaften zu bestimmen, das sich an der aktuellen oberen Grenze des Angebotes bewegt. Preisliche Überlegungen spielen keine Rolle.

Auswertung

Das mobile GIS stellt hohe Anforderungen an die Bedienungsoberfläche (grafische Bearbeitung). Komplexe grafische Oberflächen sollen dem Nutzer ein Maximum an Funktionalität bei intuitiver Bedienbarkeit bieten. Die SDKs für Handhelds

[4] (Vgl. Schwarz 2003, Folie 3)
[5] nach Schwarz (2003) u.a.

Kategorie	Merkmale	Leistung	Laufzeit	OS	SDKs
Notebook	vergleichbar Desktop-PC, nicht ergonomisch	wie Desktop-PC	mehrere Stunden	Win32, Linux, MacOS	C, Java, viele andere
Tablet-PC	DIN-A4 Touchscreen, Bedienung per Stift, Bildschirmtastatur	etwas geringer als Desktop-PC	mehrere Stunden	meist Win32	C, Java, viele andere
Handheld	kleiner Touchscreen, Bedienung per Stift oder Bildschirmtastatur, wenig Speicher	gering	viele Stunden	Windows CE / Pocket PC, Palm OS, EPOC / Symbian	WinCE-SDK (C), J2ME Profile (Java), andere
Mobiltelefon	sehr kleiner-Bildschirm, Zahlentastatur, sehr wenig Speicher	sehr gering	viele Tage	meist proprietär, Symbian, Linux	J2ME Profile (Java)

Tabelle 4.1: Merkmale mobiler Computer

und Mobiltelefone bieten momentan keine Möglichkeit solche Oberflächen zu erstellen. Die eingeschränkte Größe der Anzeige ist ein weiterer Grund diese Geräte nicht zur Zielplattform erklären zu können.
Für die Transformation von Geodaten[6] werden große Mengen Arbeitsspeicher benötigt. Desweiteren sind das Berechnen von Transformationen und die Darstellung von Vektordaten rechenintensive Vorgänge, die ein Mindestmaß an Rechenleistung von der Zielplattform abverlangen[7]. Die Limitierung von Speicher und Rechenleistung in Handhelds und Mobiltelefonen wird sicherlich durch die längere Laufzeit aufgewogen. Für das angestrebte mobile GIS werden die Erstgenannten allerdings als von entscheidender Wichtigkeit eingestuft und somit muss auf längere Laufzeit zugunsten höherer Leistungsfähigkeit verzichtet werden.
Die Anforderungen für das mobile GIS erfüllen von den betrachteten Geräten nur Notebooks und Tablet-PCs.

[6] Vektordaten belegen typischerweise viel Arbeitsspeicher
[7] (Vgl. Breunig, Brinkhoff, Bär, Weitkämpfer: XML-basierte Techniken für LBS, in: Zipf, Strobl 2002, S.26ff)

4.2 Vorhandene Komponenten

Für eine komplette Erarbeitung aller benötigten Funktionsmerkmale ist der veranschlagte Zeitrahmen von 3 Monaten ungenügend. Naheliegend ist die Suche nach frei (kostenlos) verfügbaren Software-Komponenten, die zumindest Teile der Anforderungen schon erfüllen und entsprechend verwendet werden können. Nachfolgend wird ein Überblick über die Ergebnisse dieser Recherche gegeben. Die Liste ist mitnichten als vollständig zu erachten. Vielmehr werden ausgesuchte Komponenten gelistet, die mindestens den folgenden Kriterien entsprechen:

1. enthält benötigte Funktionalität(en)
2. offener Quellcode
3. kostenfreie Wiederverwendbarkeit

Einen guten Überblick zu neuen Entwicklungen im Bereich Geoinformatik bietet die Webseite www.freegis.org. Hier werden freie GIS-Software-Projekte vorgestellt, welche sich von einer Open Source Lizenzierung eine höhere Verbreitung oder zügigere Entwicklung versprechen. Im Anschluss an die Vorstellung werden die Funktionalität und Lizenzierung der Programme direkt miteinander verglichen. Als Ergebnis dieses Vergleichs erfolgt die Auswahl der am besten geeigneten Komponenten für die Umsetzung der mobilen GIS-Anwendung.

4.2.1 GDAL[8]

Die Geospatial Data Abstraction Library (Geodaten Abstraktionsbibliothek) ermöglicht Programmen über eine standardisierte Schnittstelle auf Geodatenquellen zuzugreifen. GDAL enthält momentan circa 40 Treiber, unter anderen für GeoTIFF (lesen/schreiben) Dateien. Die OGR Simple Features Library (OGC Simple Features Specification basiert), eine C++ Open Source Bibliothek (und Kommandozeilen-Werkzeuge), ist in GDAL integriert und erlaubt Lese- und (nicht immer) Schreibzugriff auf verschiedene Vektorgeodatenformate in Dateien (z.B. ESRI Shapedateien, MapInfo MIF/MID) aber auch Datenbanken (z.B. PostGIS, Oracle). Es existiert eine Dokumentation der C-API. GDAL ist freie Software unter der Non-Copyleft MIT-Lizenz.

4.2.2 PROJ.4[9]

Die PROJ.4 Cartographic Projections Library ist eine C-Bibliothek, die Koordinatentransformationen zwischen verschiedenen Referenzkoordinatensystemen berechnet. PROJ.4 ist ebenfalls freie Software unter der Non-Copyleft MIT-Lizenz.

[8] (http://remotesensing.org/gdal)
[9] (http://remotesensing.org/proj)

4.2.3 GRASS[10]

Das Geographic Resource Analysis Support System wird auch GRASS GIS genannt und ist eine freie Desktop-GIS-Entwicklung basierend auf der Programmiersprache C. Es unterliegt der GPL. Zu den Funktionen gehören z.B. Datenmanagement, Bildverarbeitung, Grafikerstellung, Erfassung und Visualisierung von Geodaten. Von 1982 bis 1995 von der U.S. Armee als Werkzeug zur Landverwaltung und Umweltplanung entwickelt, ist GRASS inzwischen ein internationales Open-Source-Projekt, das kommerziell und in der Forschung Verwendung findet.
GRASS kann georeferenzierte Vektor- und Rasterdaten verarbeiten und besteht laut Handbuch aus über 350 Modulen, welche einzelne Funktionalitäten beinhalten. Die Bedienung des Programms erfolgt sowohl über eine Kommandozeile, was die Automatisierung von Befehlsfolgen vereinfacht, als auch über eine intuitiv bedienbare grafische Benutzeroberfläche.
Das in C programmierte Programm läuft auf unixkompatiblen Betriebssystemen und experimentell unter Windows. Die grafische Oberfläche ist javabasiert. Da für Im- und Export von Geodaten die GDAL implementiert ist, unterstützt GRASS alle GDAL Datenquellen. Die Koordinatentransformation wird über die PROJ.4 Bibliothek realisiert. Für Entwickler steht eine ausführliche Dokumentation der GRASS C-API zur Verfügung.

4.2.4 GPSDrive[11]

Das GPSDrive-Projekt entwickelt ein kartenbasiertes Navigationssystem für Linux-Computer. Unterstützt werden NMEA-fähige GPS-Empfänger mit deren Daten das Programm Informationen über Geschwindigkeit, Richtung und die aktuelle Position ausgeben kann. Kartenmaterial für die aktuelle Position kann aus dem Internet geladen und zoombar angezeigt werden. Sprachausgabe ist optional möglich. Die aktuelle Version 2.07 unterliegt der GNU GPL.

4.2.5 JTS[12]

Die Java Topology Suite ist eine OGC-Simple-Features-Implementierung für zweidimensionale georeferenzierte Geometrien für Java. Die LGPL lizenzierte API enthält zweidimensionale Auswertungsmechanismen (z.B. die Berechnung von Schnittmengen) und definiert Möglichkeiten der Manipulation von Geometrien, z.B. durch den Einsatz von Filtern.

[10] (http://grass.itc.it)
[11] (http://www.gpsdrive.de/)
[12] (http://www.vividsolutions.com/jts)

4.2.6 Geotools 2 (GT2)[13]

Die javabasierenden Geotools (Geowerkzeuge) basieren auf den 1996 an der Universität von Leeds entwickelten Geotools 1 (auch Geotools-Lite). Diese für Java Applets geschriebene API wurde komplett überarbeitet und liegt nun als Java-API (Application-Programming-Interface, dt. Anwendungs-Programmier-Schnittstelle) oder besser Sammlung von Java-Bibliotheken (externe wiederverwendbare Programmfragmente) als erste Beta Version 1.0 vor. Es wird versucht offene Standards, wie von OGC und ISO, zu implementieren und als freie Softwarebausteine zur Verfügung zu stellen. GT2 unterliegt der LGPL.
GT2 enthält unter anderem Module für Koordinatentransformationen, Im-/(nicht immer)Export von Raster- und Vektorgeodaten, Visualisierung (mit Styling) basierend auf JTS.

4.2.7 JUMP[14]

Die Java Unified Mapping Platform (JUMP) liegt seit Juni 2003 in der ersten Version 1.0 vor (Nov. 2003 - Version 1.1). JUMP ermöglicht die Bearbeitung und Visualisierung (per JTS) von Geodaten. Die Bearbeitung ähnelt herkömmlichen Vektorgrafikprogrammen und ist intuitiv zu bedienen. Funktionalitäten zur Auswertung von zweidimensionalen Geometrien werden in der komfortablen grafischen Oberfläche zur Verfügung gestellt. Der Im-/Export von wenigen Dateiformaten (u.a. ESRI Shapefile - per GT2, GML2.0) ist integriert. Es ist keine Verarbeitung von Koordinatenreferenzsystemen vorhanden. Alle Geodaten müssen also für die Bearbeitung im selben Bezugssystem vorliegen. JUMPs Datenmodell basiert auf OGC Simple Feature Specification. Vom Hersteller als Framework betitelt, verfügt JUMP über eine eingeschränkt dokumentierte aber umfassende Schnittstelle für die Entwicklung von Erweiterungen.

4.2.8 GpsTool[15]

Einige freie Java-Projekte implementieren GPS-Funktionalitäten. Ausgewählt wurde Gpsmap/Gpsylon, vorliegend in der Version 0.4.15-pre6 unter LGPL, da es im Unterschied zu vielen anderen Projekten mehrere GPS-Empfänger-Protokolle unterstützt. Das Programm nutzt die Bibliothek GPSTool, laut Quellcode von 2001-2003 an der Universität Graz entwickelt, die als GPS-Datenquelle Garmin-, NMEA-, Sirfempfänger unterstützt. Die Bibliothek ist nur im Quellcode dokumentiert.

[13] (http://www.geotools.org)
[14] (http://www.vividsolutions.com/jump/)
[15] (http://sourceforge.net/projects/gpsmap/)

4.2.9 THUBAN[16]

Der Projektname Thuban stammt vom Hauptstern des Sternbildes Drachen. Hierbei handelt es sich um einen Geodaten-Betrachter (vorliegend in Version 1.0rc1, Entwurfsphase), der auf der Scriptsprache Python basierend multiplattformfähig ist. Besonders rechenintensive Module sind allerdings in C realisiert, was die Unabhängigkeit auf die vorhandenen Portierungen - Linux, Win32; begrenzt. Thuban unterliegt der GPL und im-/exportiert Geodaten aus/in Datenbanken (GRASS, PostGIS) und ESRI Shapefiles. Koordinatentransformationen sind möglich. Zukünftig sollen zusätzliche Datenquellformate (per GDAL) eingebunden und Bearbeitungsfunktionen integriert werden.

4.2.10 Vergleich der Funktionen und Lizenzen / Auswahl

Alle Produkte sind auf Notebooks einsetzbar. Sie entsprechen damit dem Kriterium der mobilen Einsatzfähigkeit, das in Abschnitt 4.1.5 festgelegt wurde. Die Tabelle 4.2 fasst den Funktionsumfang und die Lizenzvarianten der einzelnen Software-Komponenten zur Entscheidungsfindung zusammen.

Software / Funktion	Im-/Export	Edit	Trans	GPS	Sprache	Lizenz
GRASS (inkl. GDAL, PROJ4)	x	x	x		C	GPL
GPSDrive				x	C	GPL
GT2	x	x	x		Java	LGPL
JUMP (inkl. JTS)	Dateien	x			Java	GPL
GPSTool				x	Java	LGPL
THUBAN	x				Python	GPL

Tabelle 4.2: Vorhandene Software - Funktionen und Lizenzen

Jede Komponente unterliegt einer Open-Source-Lizenz und ist kostenfrei erhältlich. Die für das mobile GIS benötigten Funktionen, definiert im abstrakten Konzept 4.1 auf Seite 40, sind als C, Java oder Python Komponenten vorhanden. Da eine reine Lösung, also keine Mischung von Programmiersprachen, angestrebt wird, kommen nur Java und C Komponenten infrage. In diesen Sprachen liegen *alle* Teilfunktionen in der jeweiligen Programmiersprache vor. Die Java Komponenten weisen intuitive Bearbeitungsmöglichkeiten in JUMP und eine Vielfalt der

[16] (http://thuban.intevation.org)

unterstützten Geräte von GPSTool auf. Die Plattformunabhängigkeit von Java, die den (relativ problemlosen) Einsatz derselben Anwendung unter Win32, Linux und MacOS ermöglicht ist ein weiterer Vorteil. Diese Vorteile führen zu der Entscheidung: Das mobile GIS wird mit den Java Komponenten entwickelt.

4.3 Evaluation

Nachdem im vorangegangenen Abschnitt eine Auswahl von Software-Komponenten getroffen wurde, sollen diese nun genauer untersucht werden. Es gilt herauszuarbeiten, wie die Schnittstellen beschaffen sind und die Funktionalität der Komponenten für den Anwendungszweck verwendet werden kann. Jeder Abschnitt schließt mit der Entwicklung von Test-Software, in der benötigte (Teil-)Funktionalität umgesetzt wird. Dabei auftretende Probleme werden gelöst oder zumindest dokumentiert. Ziel ist es, herauszufinden, ob die Komponenten für den angestrebten Funktionszweck verwendbar sind. Sollte dies nicht der Fall sein, muss die Auswahl vorhandener Software-Komponenten wiederholt werden. Die im Erfolgsfall entstandene Testanwendung (engl. Proof of Concept) kann später als Vorlage der finalen Implementierung dienen.

4.3.1 Erweiterungen und JUMP

JUMP enthält Mechanismen und Schnittstellen, welche die Integrierung von Erweiterungen erleichtern sollen. Im (noch unvollständigen) JUMP Developer Guide Version 0.5 sind diese dokumentiert[17].

Aufbau

Eine Erweiterung (engl. Extension) kann aus mehreren Einzelfunktionen bestehen. Diese werden in der JUMP-Architektur Plugin genannt, ein durchaus verbreiteter Begriff für externe, nachträglich installierbare Erweiterungen von Software, die sozusagen in die Software eingesteckt (engl. to plug) werden.
Die Extension ist eine Implementierung des Configuration Interface. Dessen definierte *#configure* Methode wird vom Framework aufgerufen, um die Plugin-Klassen zu installieren und andere notwendige Initialisierungen vorzunehmen. Beispielhaft wurde die TestExtension-Klasse (Listing 4.1) erzeugt.

```
1  package de.soldin.gt2jump.test.jumpplugin;
2  import com.vividsolutions.jump.workbench.plugin.Extension;
3  import com.vividsolutions.jump.workbench.plugin.PlugInContext;
4
5  public class TestExtension
```

[17] (Vgl. Vivid Solutions 2003, JUMP Developer Guide)

```
6     extends Extension
7     {
8     public void configure(PlugInContext context) throws Exception {
9       new TestPlugin().initialize(context);
10      new TestThreadedPlugin().initialize(context);
11    }
12
13    public String getVersion() {
14      return "1.0";
15    }
16
17    public String getName() {
18      return "Test Extension ("+TestPlugin.NAME+","+TestThreadedPlugin.NAME+")";
19    }
20
21  }
```

Listing 4.1: Testextension.java

Die TestExtension erzeugt Instanzen der TestPlugins und initialisiert diese. Der Parameter PlugInContext enthält Referenzen auf Manager und wichtige Objekte des JUMP-Frameworks und ermöglicht so, auf grafische und konzeptionelle Komponenten der elterlichen Anwendung zuzugreifen. Weiterhin stellt die Extension Information über Namen und Version der Erweiterung zur Verfügung.
Plugin-Klassen müssen eines der beiden Interfaces PlugIn oder ThreadedPlugIn implementieren. Der Einfachheit halber stehen die abstrakten Klassen[18] AbstractPlugIn und ThreadedBasePlugIn zur Ableitung zur Verfügung. Ein einfaches Plugin ist im Listing A.1 auf Seite VIII zu finden.
Dieses "Standard"-Plugin, eine Implementierung des JUMP PlugIn-Interface, stellt mit einem Plugincontext parametrisierte Methoden zur Initialisierung (#initialize) und Ausführung (#execute) zur Verfügung. Die Initialisierung fügt das Plugin normalerweise der grafischen Oberfläche hinzu. Sollte es von dort aufgerufen werden, wird die Ausführung gestartet und blockiert die Anwendung bis zur Beendigung der Plugin-Aktivität. Eine erweiterte nicht-blockierende Version ist durch Implementierung von ThreadedPlugIn möglich (Listing A.2 auf Seite IX).
Hier wird das Plugin um eine zweite Ausführungsmethode (#run) erweitert. Zusätzlich zum bekannten Plugincontext wird ein Monitorobjekt übergeben. Sollte die erste Methode (#execute) den boolschen Wert "wahr" zurückgeben, wird ein *paralleler* Prozess gestartet, der wiederum die zweite Ausführungsmethode (#run) startet und somit die Anwendung nicht mehr blockiert. Vorgesehen ist diese Erweiterung für Prozesse, die lange Rechenzeit benötigen aber im Hintergrund arbeiten können. Diese Parallelläufigkeit von Prozessen wird in Java als Threading bezeichnet. Der entstehende Prozess ist ein Thread.

[18] Konkrete Klassen die als Definitionen betrachtet werden, da bestimmte Eigenschaften nur definiert und nicht implementiert sind (z.B. abstrakte Methoden). Sie sind also nicht instanzierbar, können aber abgeleitet werden. Die resultierende untergeordnete Klasse muss fehlende Eigenschaften implementieren und kann dann über die Eigenschaften der übergeordneten Klasse verfügen.

Einbindung in die grafische Oberfläche

Menüpunkte stellen die Verbindung zwischen Benutzer und Plugin her. Der Plugincontext enthält Referenzen auf verschiedene Objekte (z.B. FeatureInstaller), die die Erstellung und Bearbeitung von Menüeinträgen ermöglichen[19]. Zusätzlich können Attribute wie ein grafisches Symbol (engl. Icon)[20] und Bedingungen für die (De-)Aktivierung[21] festgelegt werden (Abbildung 4.2). Während der Feature Installer (zuständig für Hauptmenü und Ebenen- und Kategoriekontextmenüs) hinreichend dokumentiert ist, muss für Einträge in das Optionsmenü der Quellcode zu Rate gezogen werden.

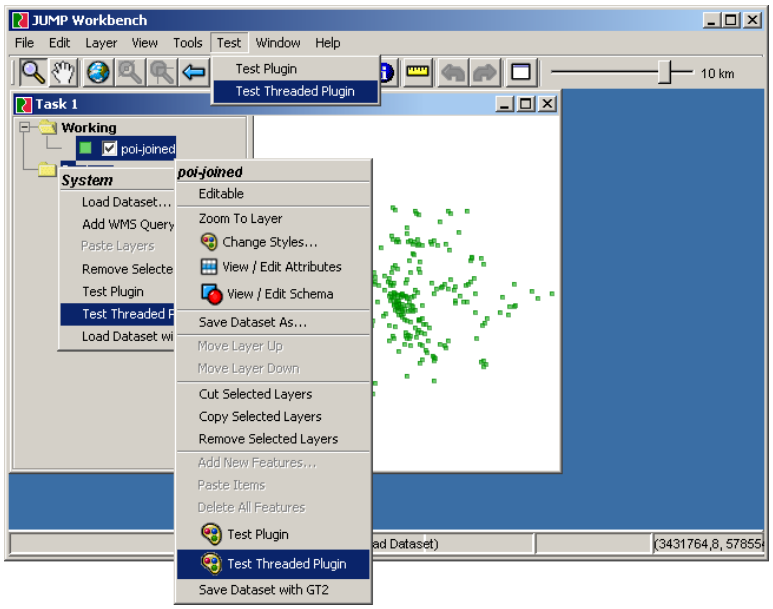


Abbildung 4.2: alle Menüpunkte der Test-Extension auf einen Blick

Aktivierung

Das JUMP-Framework definiert zwei Methoden Plugins zu aktivieren bzw. zu laden. Methode 1 sollte für fertige Erweiterungen zum Einsatz kommen, da Java-Archive erwartet werden. Methode 2 ist für die Entwicklung von Erweiterungen gedacht.

Methode 1: Angabe des Verzeichnisses beim Start der JUMP Anwendung als Kommandozeilenparameter (Listing 4.2 auf der nächsten Seite).

[19] Listings A.1 auf Seite VIII, Zeilen 18-52 und A.2 auf Seite IX, Zeilen 20-54)
[20] Vgl. Listing A.2 auf Seite IX, Zeile 52
[21] Vgl. Listing A.2 auf Seite IX, Zeile 43

```
C:\>java com.vividsolutions.jump.workbench.JUMPWorkbench  -plug-in-directory /pfad/mit/plugins
```

Listing 4.2: Angabe des Plugin-Verzeichnisses beim Start

Ist dieser Parameter definiert, werden bei Anwendungsstart alle in diesem Verzeichnis vorhandenen zipkomprimierten Dateien (z.B. *.jar Java-Archive) geöffnet und nach Instanzen der Extension-Klasse durchsucht. Wenn die Datei einer gefundenen Instanz dem Namensschema "[NAME]Extension-.class" folgt, wird deren *#configure* Methode ausgeführt und stößt die Initialisierung an.

Methode 2: Definition der Plugins in der Datei *workbench-properties.xml* und Angabe beim Start als Kommandozeilenparameter. Anstelle der Extension werden die Namen der Plugins in die Datei eingetragen. Die Datei muss im XML-Format vorliegen (Listing 4.3). Als Kommandozeilenparameter bei Anwendungsstart bekanntgemacht (Listing 4.4), ist JUMP nun in der Lage die Plugins zu finden und zu initialisieren.

```
<workbench>
  <plug-in>TestPlugin</plug-in>
  <plug-in>TestThreadedPlugin</plug-in>
</workbench>
```

Listing 4.3: Beispiel für die Datei workbench-properties.xml

```
C:\Sandbox\>java com.vividsolutions.jump.workbench.JUMPWorkbench -properties C:\Sandbox\
      HelloWorld\workbench-properties.xml
```

Listing 4.4: Angabe der workbench-properties.xml in der Kommandozeile

Test der Test-Extension

Für die Evaluierung der Funktionalitäten wurde eine Test-Extension mit zwei Test-Plugins (normal, threaded) erstellt. Beide Plugins tragen sich in Hauptmenü (Menüpunkt: Test) und die Kontextmenüs für Ebene und Kategorien ein. Bei Auswahl eines dieser Punkte wird das entsprechenden Plugin ausgeführt und gibt eine Meldung aus. Zusätzlich zu dieser Meldung zählt das threaded Plugin bis 1000, während die Oberfläche bedienbar bleiben sollte (Abbildung 4.3 auf der nächsten Seite).
Obwohl das threaded Plugin vom Interface ThreadedPlugin abgeleitet ist, kann bei einer Ausführung nicht auf die Hauptanwendung zugegriffen werden. Dies liegt in der gegenwärtigen Architektur von JUMP begründet. Ein Dialog mit dem hauptverantwortlichen Entwickler ergab, dass JUMP nicht thread-safe ist. Das heißt gleichzeitige Zugriffe von mehr als einem Prozess würden zu unerwarteten, wahrscheinlich unerwünschten Ergebnissen führen (Korrespondenz 6 auf Seite XIX).

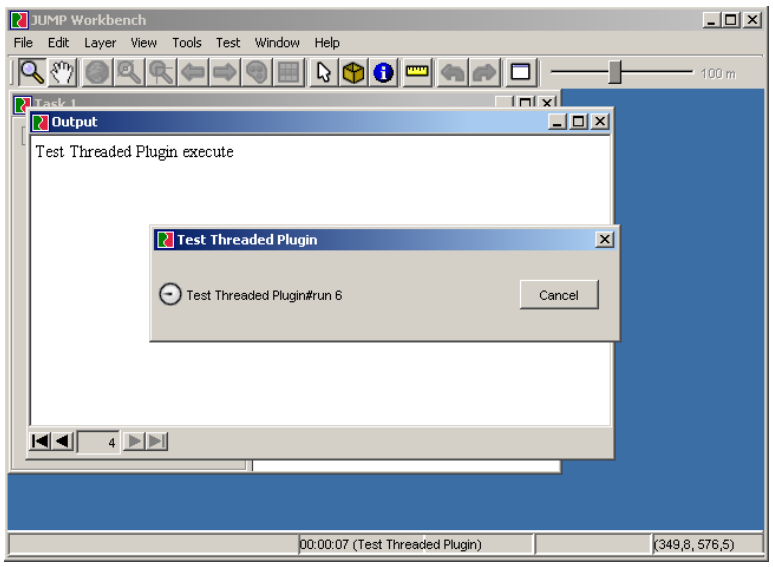


Abbildung 4.3: Test-Threaded-Plugin gestartet

Deshalb ist der Taskmonitor (die Statusanzeige im Vordergrund) momentan ein modaler Dialog. Problematisch ist weiterhin die mangelhafte Dokumentation. Die Analyse des Quellcodes bei Problemen ist mühselig und zeitintensiv. Die Einbindung von weiteren Java-Bibliotheken (*.jar Dateien) ist nicht vorgesehen, kann aber durch die manuelle Erweiterung des Klassenpfades beim Anwendungsstart gelöst werden.

4.3.2 Datenquellen in JUMP und GT2

Beide Projekte haben das Datenmodell der Simple-Features-Spezifikation des OGC umgesetzt, allerdings unabhängig und in voneinander abweichender Weise.

Generelles Datenmodell

Das Ergebnis des Lesens einer Datenquelle ist bei beiden immer eine Feature-Collection, welche eine Anzahl an Features enthält, die wiederum Geometrien und weitere Attribute haben. Auch zum Schreiben in eine Datenquelle wird dem entsprechenden Datenquellenmodul jeweils eine Feature-Collection übergeben. Doch die Objekttypen für Feature-Collection (Feature, Attribut usw.) benutzen nicht dieselben Schnittstellen. So heißt z.B. die Methode, um die Geometrie eines Features zu erhalten, bei GT2 ”getDefaultGeometry” und bei JUMP ”getGeometrie”. Beide Komponenten benutzen Javaprimitive (grundsätzliche Datentypen) für die Attribute und JTS-Primitive für die Geometrien.

Datenquellen in JUMP

Ein Datasource-Objekt (dt. Datenquelle) ist laut JUMP-Definition ein Objekt, das Daten zwischen der Anwendung und einer Datei oder anderen Datenquelle bewegt. Dieses ersetzt zukünftig die Reader- und Writer-Objekte, wie sie momentan in der Anwendung zu finden sind. Es ist nicht geplant die vorhandenen Datenquellen umzuschreiben. DataSourceQueryChooser (dt. Datenquellen-Anfrage-Auswähler - Parameterauswahl für die Datenquelle z.B. Dateiname, Datenbanktabelle etc.) sind grafische Komponenten einer JUMP-Datasource. Datasources und Reader/Writer werden erwähnt, sind aber undokumentiert.

Datenquellen in GT2

Datenquellen in Geotools basieren alle auf dem Modul ”org.geotools.data”. Die darin enthaltenen Schnittstellen und konkreten Klassen bilden die Grundlage für Datenquellenmodule mit gleichen Methoden für Schreib- und Lesezugriffe. Datenquellenmodule sind momentan für Datenbanken[22] und Dateiformate[23] vorhanden. Es ist eine Methodik zur Reflektion[24] integriert und über eine DatasourceFinder-Klasse können im Klassenpfad befindliche DatasourceFactories ausfindig gemacht und instanziiert werden. Noch nicht alle Module unterstützen diesen Suchprozess.

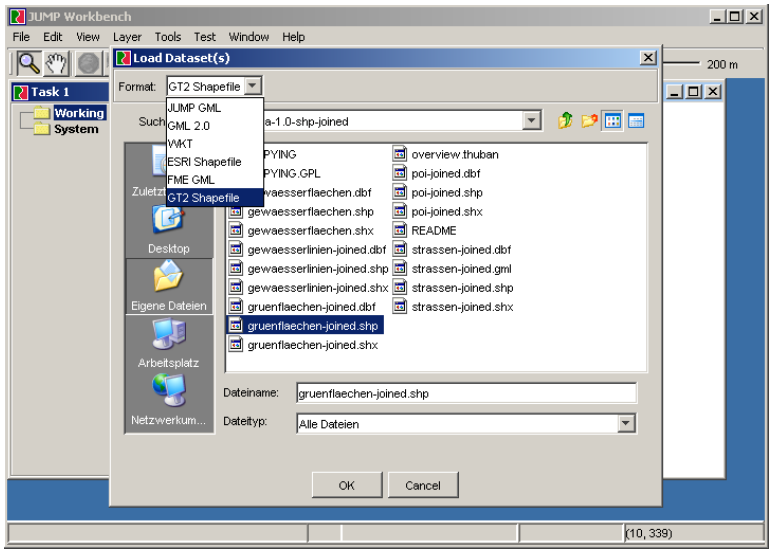


Abbildung 4.4: GT2-Shapefile-Datenquelle als Plugin in JUMP

[22] Arc Grid, GML 2.0, GTOPO30, MapInfo, Pickle, Shapefile, VPF
[23] MySQL, Oracle, PostGIS
[24] Objekte beschreiben ihre Funktionen und Eigenschaften, z.B. Lese- oder Schreibfähigkeit

Test einer GT2-JUMP-Datenquelle

Beide Pakete dokumentieren die Schnittstellen zu Datenquellen nur mangelhaft. Für GT2 stehen eine Beispielanwendung, die aus einer Datei Daten importiert, und die ausführliche Dokumentation im Quellcode zur Verfügung. Die JUMP-Dokumentation ist nicht erwähnenswert, so muss eine Interpretation des funktionierenden Quellcodes diese ersetzen. Da nur Reader/Writer basierte Datenquellen existieren, bietet es sich an, deren Funktionsweise zu analysieren und zu imitieren.
Die Verwendung unterschiedlicher, aber kompatibler Datenmodelle erfordert eine Lösung. Denkbar ist eine hybride JUMP/GT2-Datenstruktur, das heißt eine Kapselung, bei der z.B. die GT2-Daten in ein JUMP-kompatibles Objekt gekapselt werden. Das würde eine zeitaufwändige Entwicklung einer kompletten Objektkette (Featurecollection, Feature, Attributtypen etc.) bedeuten. Alternativ ist die Entwicklung eines Konverters denkbar, der die Features Stück für Stück in das andere ”Format” übersetzt. Das bedeutet einen extra Zwischenschritt bei jeder Bewegung von Daten zwischen Anwendung und Datenquelle, ist aber Aufgrund der Kompatibilität innerhalb kurzer Zeit herzustellen.

Für den Test wurde ein Datenquellenmodul nach dem Reader/Writer-Modell erstellt. Als GT2 Datenquelle wurde das Shapefile-Modul gewählt, da es Lese- und Schreibzugriff ermöglicht (Abbildung 4.4 auf der vorherigen Seite).
Als Ergebnis dieses Tests kann festgehalten werden, dass das Reader/Writer Modell sehr stark in die Anwendung integriert und somit schwer nachzuvollziehen ist. Es ist möglich nachträglich Reader/Writer zu integrieren, was aber fragwürdig für eine als depreciated (dt. Wert verlieren, auslaufend) gekennzeichnete Methodik ist.
Der Verzicht auf das Datasource-Modell von JUMP und eine zukünftige Übernahme des vollständigen Modells von GT2, inklusive dem Finden von Datenquellenmodulen, erscheint sinnvoll. Die Umsetzung dieser Idee überschreitet den Rahmen dieser Arbeit, wird vom Autor aber in der JUMP-Mailing-Liste angeregt. Es existieren kleinere Inkompatibilitäten zwischen den Datenmodellen. So ist z.B. die FID (Feature Identifikation)[25] in GT2 als Zeichenkette, in JUMP aber als Zahl definiert. Aufgrund der Tatsache, dass die FID momentan kaum tatsächliche Verwendung findet, relativiert sich dieses Problem. Beide Komponenten werden noch aktiv entwickelt und erweitert. Dadurch können sich definierte Schnittstellen und benutzte Klassen verändern. Um mit dieser fortschreitenden Entwicklung Schritt halten zu können, muss die Konvertierungslösung regelmäßig angepasst werden.

[25] vom OGC spezifizierte einmalige Identifikation eines Geodatensatzes

4.3.3 Koordinatentransformation mit GT2

Die Georeferenzierung von Features geschieht im Allgemeinen über die Zuweisung mindestens einer Geometrie (vgl. Abschnitt 2.2.3 auf Seite 16). Diese Geometrie enthält Koordinaten, die wiederum in einem zuvor festgelegten Bezugssystem Gültigkeit besitzen. Werden die Koordinaten der Geometrie verändert, ändert sich auch die Georeferenz.

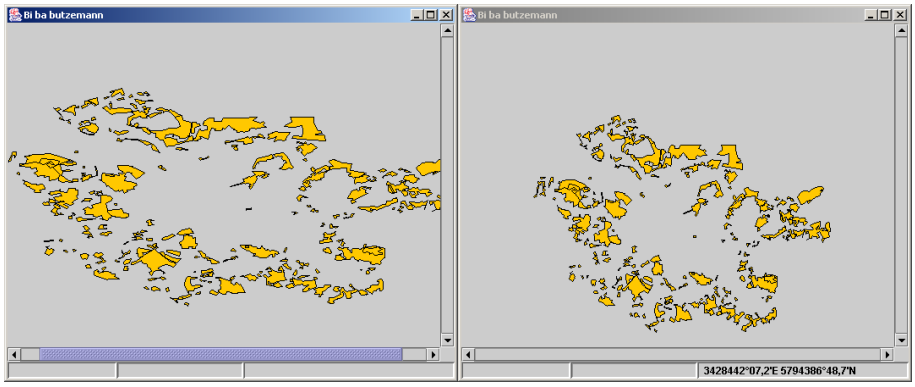


Abbildung 4.5: Die Grünflächen von Osnabrück (hier gelb) - transformiert und original

Koordinatentransformationdienste des CTS

Das GT2-Modul CTS (Coordinate Transformation Services) implementiert die CTS-Spezifikation des OGC. Nach dem bekannten Factory-Konzept werden Factories für die zur Koordinatentransformation notwendigen Schritte zur Verfügung gestellt. Anhand von Datumsparametern (z.B. als WKT-Zeichenkette, oder aus der EPSG[26] Datenbank) erstellt eine Fabrik Koordinatensystem-Objekte. Sind Quell- und Zielkoordinatensystem erzeugt, können diese einer Fabrik zur Erstellung eines Koordinatentransformationsobjektes übergeben werden. Von diesem wiederum kann ein mathematisches Transform-Objekt erzeugt werden, welches letztendlich Koordinaten transformieren kann. Die Listings 4.5 und 4.6 verdeutlichen die Vorgehensweise.

Koordinatenfilter in JTS

Zur Bearbeitung von Geometrien stellt JTS ein Filterkonzept zur Verfügung. Für die Veränderung oder Auswertung der Koordinaten von Geometrien ist das *com.-*

[26] European Petroleum Survey Group; dt. Europäische Erdöl-Vermessungsgruppe - diese Vereinigung stellt kostenlos eine Datenbank von Referenzkoordinatensystemen und deren Daten zur Verfügung

```
159     // Ed added: Convert features to WGS84
160     CoordinateSystemFactory csFactory = CoordinateSystemFactory.getDefault();
161     CoordinateSystem sourceCSWKT =
162       csFactory.createFromWKT(WKTFactory.getGaussKruegerZone3());
163     System.out.println(sourceCSWKT.getDimension());
164     CoordinateSystem targetCSWKT =
165       csFactory.createFromWKT(WKTFactory.getGeographicWGS84());
166     System.out.println(targetCSWKT.getDimension());
167     MathTransform transform = this.transformInit(sourceCSWKT, targetCSWKT);
168     CoordinateTransformFilter trans =
169       new CoordinateTransformFilter(sourceCSWKT, targetCSWKT);
170     trans.setYx(true);
171
172     for (Iterator iter = features.iterator(); iter.hasNext();) {
173       String foobar = new String();
174       Feature feature = (Feature) iter.next();
175       Geometry geom = feature.getDefaultGeometry();
176       //transformiere
177       //geom.apply(trans);
178
179       // simple output
180       Object[] attribs = feature.getAttributes(null);
181       for (int i = 0; i < attribs.length; i++) {
182         Object attrib = attribs[i];
183         foobar += "<" + attrib.toString() + ">";
184       }
185       System.out.println(foobar);
186     }
```

Listing 4.5: Auszug aus MapViewerEd.java

```
25    public static String getGaussKruegerZone3() {
26      return "PROJCS[\"DHDN / Gauss-Kruger zone 3\", GEOGCS[\"DHDN\",
            DATUM[\"Deutsches Hauptdreiecksnetz\", SPHEROID[\"Bessel
             1841\", 6377397.155, 299.1528128, AUTHORITY[\"EPSG\",\"7004\"]], TOWGS84
            [598.1, 73.7, 418.2, 0.2019999999999998, 0.04499999999999995, -2.4549999999999974, 6.7],
            AUTHORITY[\"EPSG\",\"6314\"]], PRIMEM[\"Greenwich\", 0.0, AUTHORITY[\"EPSG\",\"8901\"]],
            UNIT[\"degree of angle\",0.017453292519943295], AXIS[\"Geodetic latitude\",NORTH
            ], AXIS[\"Geodetic longitude\",EAST], AUTHORITY[\"EPSG\",\"4314\"]], PROJECTION[\"
            Transverse_Mercator\"], PARAMETER[\"semi_major\", 6377397.155], PARAMETER[\"semi_minor
            \", 6356078.962818189], PARAMETER[\"central_meridian\", 8.999999999999991], PARAMETER[\"
            latitude_of_origin\", 0.0], PARAMETER[\"scale_factor\", 1.0], PARAMETER[\"false_easting
            \", 3500000.0], PARAMETER[\"false_northing\", 0.0], UNIT[\"metre\",1.0], AXIS[\"Northing
            \",NORTH], AXIS[\"Easting\",EAST], AUTHORITY[\"EPSG\",\"31467\"]]";
27    }
28
29    public static String getGeographicWGS84() {
30      return "GEOGCS[\"WGS 84\", DATUM[\"World Geodetic System 1984\", SPHEROID[\"WGS
             84\", 6378137.0, 298.257223563, AUTHORITY[\"EPSG\",\"7030\"]], AUTHORITY[\"EPSG
            \",\"6326\"]], PRIMEM[\"Greenwich\", 0.0, AUTHORITY[\"EPSG\",\"8901\"]], UNIT[\"degree of
            angle\",0.017453292519943295], AXIS[\"Geodetic latitude\",NORTH], AXIS[\"Geodetic
            longitude\",EAST], AUTHORITY[\"EPSG\",\"4326\"]]";
31    }
```

Listing 4.6: Auszug aus WKTFactory.java

vividsolutions.jts.geom.CoordinateFilter-Interface vorgesehen. Dieses definiert eine Methode (#filter, Vgl. Listing A.3 auf Seite X Zeile 276ff), deren Parameter eine einzige Koordinate zur Verarbeitung als Referenz übergeben wird. Wird dieser Filter auf ein Geometrie-Objekt angewendet (#apply), werden alle Koordinaten mit dem Filter bearbeitet.

Test eines Koordinaten-Transformation-Filters

Als Grundlage für die Testanwendung wurde ein Beispiel aus GT2 verwendet. Der Mapviewer (dt. Kartenbetrachter) ist in der Lage Shape-Dateien zu importieren und die resultierende Feature-Collection anzuzeigen. Die Anwendung verdeutlicht das Zusammenspiel einzelner GT2 Module. Zwischen dem Ladevorgang und der Visualisierung liegen die Geodaten als Feature-Collection vor und können bear-

beitet werden. Hier wurde die Transformation eingefügt, indem in einer Schleife die Geometrien sämtlicher Features mit einem Koordinaten-Filter manipuliert werden (Vgl. Listing 4.5 auf der vorherigen Seite, Zeilen 172-186). Das Ergebnis der Transformation ist auf Abbildung 4.5 auf Seite 57 abgebildet.
Bei der Umsetzung fiel auf, dass das mathematische Transformationsobjekt entgegen der Dokumentation bei Angabe der Achswerte in der Reihenfolge geographische Länge (x, Rechtswert) und Breite (y, Hochwert) zu falschen Ergebnissen kommt. Erst ein Vertauschen vor und ein Zurücktauschen nach der Transformation führt zum richtigen Ergebnis. Wenn die Definition des Referenzkoordinatensystemes für das zweite Bezugssystem (Listing 4.6 auf der vorherigen Seite) näher betrachtet wird, findet man `AXIS["Geodetic latitude",NORTH], AXIS["Geodetic longitude",EAST]` und etwas später `AXIS["Northing",NORTH], AXIS["Easting",EAST]`. Das Koordinatensystem definiert also explizit Länge vor Breite. Ein Austausch der jeweiligen Abschnitte behebt das Problem.

4.3.4 Positionsbestimmung mit GPStool

Für Testzwecke steht ein Roadpilot-USB-GPS-Empfänger zur Verfügung. In Verbindung mit GPSTool soll dieser die Grundlage für die satellitengestützte Positionsbestimmung bilden.

GPS-Empfänger

Erste Versuche der Inbetriebnahme scheitern an der Notwendigkeit eines plattformspezifischen Treibers. Nach der Installation des Treibers und dem Anschluss des Gerätes wird ein serieller Port simuliert, der Daten im Textformat sendet und empfängt (Listing 4.7). Mit einem Terminalprogramm, z.B. Hyperterminal, kann der Datenstrom visualisiert werden.

```
$GPGSV,2,1,08,07,43,083,28,29,26,164,31,05,26,234,41,23,19,316,40*75
$GPGSV,2,2,08,26,43,173,32,18,24,267,38,09,71,288,37,28,20,058,*7D
$GPRMC,121504.848,A,5205.4930,N,01135.1454,E,000.0,277.8,080384,001.3,E*6E
$GPVTG,277.8,T,276.5,M,000.0,N,0000.0,K*72
$GPGGA,121505.848,5205.4930,N,01135.1454,E,1,07,01.1,00077.7,M,44.6,M,,*62
$GPGLL,5205.4930,N,01135.1454,E,121505.848,A*3C
$GPGSA,A,3,07,29,05,23,,26,,18,09,,,,02.4,01.1,02.1*0B
$GPGSV,2,1,08,07,43,083,30,29,26,164,32,05,26,234,41,23,19,316,40*7F
$GPGSV,2,2,08,26,43,173,32,18,24,267,38,09,71,288,37,28,20,058,*7D
$GPRMC,121505.848,A,5205.4930,N,01135.1454,E,000.0,277.8,080384,001.3,E*6F
$GPVTG,277.8,T,276.5,M,000.0,N,0000.0,K*72
$GPGGA,121506.848,5205.4930,N,01135.1455,E,1,07,01.1,00077.6,M,44.6,M,,*61
$GPGLL,5205.4930,N,01135.1455,E,121506.848,A*3E
```

Listing 4.7: rohe NMEA-Datensätze direkt von seriellen Schnittstelle

Diese zeilenweisen Datensätze sind im NMEA-Format abgefasst und enthalten z.B. Daten zu aktiven Satelliten ($GPGSA), Satelliten in Sicht ($GPGSV) oder

der Position ($GPGLL)[27]. Das NMEA-Protokoll ist Teil des NMEA Standards der namensgebenden National Marine Electronics Association, einer Industrievereinigung, die Standards und Qualität für maritime Elektronik fördert[28].

GPSTool

GPSTool, eine einfache Kommandozeilenanwendung, demonstriert wie Daten per GPS-API vom GPS-Empfänger gelesen werden können. Durch die Angabe von Parametern in der Kommandozeile können Position, Höhe, Geschwindigkeit, Richtung und Informationen über die Satelliten und andere Informationen ausgegeben werden. NMEA- und Garmin-Protokoll werden unterstützt.

Java Communications API

Für den Zugriff aus Java auf die serielle Schnittstelle des Computers ist eine Erweiterung der Java-Laufzeitumgebung zu installieren. Die Java Communications API ermöglicht den Zugriff auf parallele (IEEE 1284) und serielle Schnittstellen. Dafür kommen plattformspezifische Bibliotheken zum Einsatz. Die API ermöglicht die Aufzählung aller verfügbaren Schnittstellen, ereignisbasierte Überwachung von Statusveränderungen, asynchrone und synchrone Ein-/Ausgabe, Auflösung des Eigentümers und Übernahme von Schnittstellen[29].

Test von GPSTool

Für den vorgesehenen Anwendungszweck ist es notwendig, die aktuelle Position vom GPS-Empfänger zu erhalten. GPSTool enthält diese Funktion (Listing 4.8). Eine Analyse des Quellcodes und der Quellcodekommentare zeigt die Funktionsweise auf.

```
>java -jar gpstool.jar -d COM4 -s 4800 --printalt --printpos

  location: GPSPosition[lat: 52.09153, long:11.585673333333334]
  altitude: 83.7
  location: GPSPosition[lat: 52.09153, long:11.585675]
  altitude: 83.6
  location: GPSPosition[lat: 52.09153, long:11.58568]
  altitude: 83.3
  location: GPSPosition[lat: 52.09153, long:11.585678333333334]
```

Listing 4.8: Ausführung von GPSTool in der Kommandozeile

Die dem GPSTool zugrundeliegende Java-GPS-API ist in der Lage NMEA-, Garmin- und Sirf-Protokoll Datensätze zu verarbeiten, die wahlweise aus Dateien, von der seriellen Schnittstelle oder dem Netzwerk (GPS-Service) gelesen werden

[27] (Vgl. www.nmea.de 2004, NMEA 0183 Datensätze)
[28] (Vgl. National Marine Electronics Association 2004, NMEA Publications and Standards)
[29] (Vgl. Sun Microsystems 2004, Java Communications API)

können. Die API basiert im Wesentlichen auf drei abstrahierten Objekten: GPS-Device (GPSDevice, dt. GPS-Gerät), GPSDataProcessor (dt. Daten-Prozessor), PropertyChangeListener (dt. Eigenschaftsveränderungshörer).
Nach Auswahl eines Gerätes, kann dieses einem Datenprozessor des benötigten Protokolls zugewiesen werden. Nach Registrierung des Listeners (dt. Hörer), wird dieser über Änderungen der GPS Daten benachrichtigt. Das Listing 4.9 ist eine gekürzte Zusammenfassung der Initialisierung der API und Registrierung von Listenern. Die lauffähige Version für den Test GPSToolSimple ist im Listing A.5 auf Seite XIII zu finden. Dabei handelt es sich um eine vereinfachte, nicht interaktive Version des GPSTools.

```
1   // Define device to read data from
2   GPSDevice gps_device;
3   Hashtable environment = new Hashtable();
4   environment.put(GPSSerialDevice.PORT_NAME_KEY,"COM4");
5   environment.put(GPSSerialDevice.PORT_SPEED_KEY,new Integer(4800));
6   gps_device = new GPSSerialDevice();
7   gps_device.init(environment);
8
9   // Define a processor
10  GPSDataProcessor gps_data_processor = new GPSNmeaDataProcessor();
11  gps_data_processor.setGPSDevice(gps_device);
12  gps_data_processor.open();
13
14  // Create a listener that prints out only
15  PropertyChangeListener listener =
16    new PropertyChangeListener(){
17      public void propertyChange(PropertyChangeEvent event) {
18        System.out.println(event.getPropertyName()+": "+event.getNewValue());
19      }
20    };
21
22  // Add the listener for selected events only
23  gps_data_processor.addGPSDataChangeListener(GPSDataProcessor.LOCATION, listener);
24  gps_data_processor.addGPSDataChangeListener(GPSDataProcessor.ALTITUDE, listener);
```

Listing 4.9: simplifiziertes GPSTool mit Angabe der Parameter im Quellcode

Der GPS-Empfänger ist an der seriellen Schnittstelle COM4 mit einer Geschwindigkeit 4800 Byte/Sekunde angeschlossen und ein NMEA-Datenprozessor wird initialisiert. Der Listener wird nur über Höhen- und Ortsverbindungen informiert und gibt diese aus. Die Ausgabe ist mit der in Listing 4.8 auf der vorherigen Seite identisch.
Die Benutzung der seriellen Schnittstelle erweist sich als komplex. Der GPS-Empfänger, genauso wie die Schnittstelle des Computers und die Anwendung müssen die Geschwindigkeit und weitere Parameter kennen und konfigurieren. Die Java Communication API ist plattformspezifisch und nicht Teil der Standard-Laufzeitumgebung. Da die Erweiterung benötigt wird, sollten Informationen zum Bezug und zur Installation in die Dokumentation der Zielanwendung einfließen. Die Trennung von GPS-Datenquelle und GPS-Daten-Prozessor ermöglicht verschiedene Gerät/Protokoll-Kombinationen einzusetzen. Dies sollte in die Anwendung integriert werden.

4.4 Konkreter Konzeptentwurf

Aus dem abstrakten Konzeptentwurf des vorigen Kapitels soll nun das konkrete Konzept des mobilen GIS erstellt werden. Dazu werden die Funktionsbeschreibungen des abstrakten Konzeptes auf die ausgewählten Software-Komponenten und erarbeiteten Schnittstellen angewendet. Rekapitulierend werden folgende Funktionalitäten benötigt, deren Ausgangsstatus in JUMP nach dem Bindestrich angegeben ist:

1. Im-/Export - einige Dateiformate unterstützt, Datenbankanbindung fehlt
2. Bearbeitung/Erstellung - umfassende Bearbeitungs-/Erstellungswerkzeuge
3. Positionsbestimmung - keine Positionsbestimmung möglich
4. Transformation - kein Bezugssystem zuweisbar, keine Koordinatentransformation integriert

Die nicht oder ungenügend im JUMP-Framework enthaltenen Funktionen werden, wie in Abbildung 4.6 verdeutlicht als Extensions integriert. Da Bearbeitung und Erstellung bereits vorhanden sind, erübrigt sich die Erstellung einer Extension hierfür.

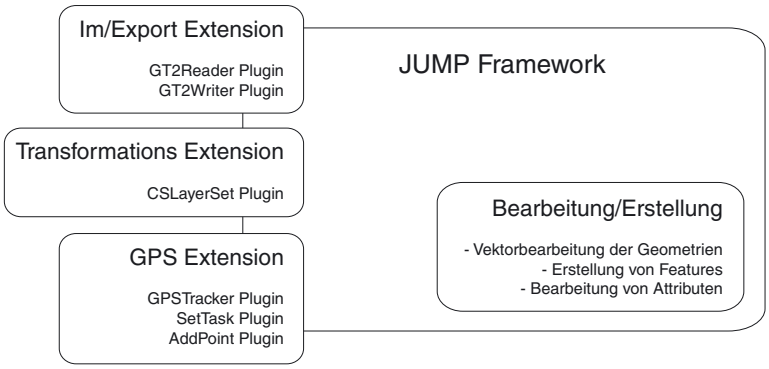


Abbildung 4.6: konkreter Konzeptentwurf: herzustellende Extensions und JUMP

Jede Extension enthält ein oder mehrere Plugins, welche mit der grafischen Oberfläche von JUMP verknüpft werden. Ein Menüpunkt ist immer mit genau einem Plugin verknüpft. Von dort aufgerufen, wird die pluginspezfische Teilfunktion ausgeführt. So ist zum Beispiel die Im-/Export Extension (GTReaderWriterExtension) in zwei Plugins (GT2ReaderPLugin, GT2WriterPlugin) unterteilt, die über zwei Menüpunkte (”Load Dataset with GT2”, ”Save Dataset with GT2”) aufgerufen werden können.
Jedes Plugin wiederum benutzt die JUMP API und Java-Schnittstellen um Aktionen durchzuführen oder durchführen zu lassen. Z.B. greift GPSPlugin über

den PluginContext auf die aktuellen Anzeigeausschnitt zu und verändert diesen, gleichzeitig werden Komponenten eines anderen Plugins (CSLayerSetPlugin) zur Transformation der aktuellen Position in ein anderes Bezugssystem benutzt. Details dieser Realisierung werden im folgenden Abschnitt besprochen.

4.5 Details der Umsetzung

Da jede Erweiterung des JUMP-Frameworks eine komplexe kleine Anwendung darstellt, werden diese jetzt abschnittsweise erläutert. Anhand von Klassen- und Sequenzdiagrammen der Erweiterung werden allgemein Struktur und Funktionsweise der Lösung vorgestellt. Um den Umfang einzugrenzen, werden abschließend nur besonders interessante Teilaspekte detailliert beschrieben.

4.5.1 Im-/Export-Extension

Die strukturell einfachste der drei Extensions besteht aus vier Komponenten: eine Extension (*GT2ReaderWriterExtension*), zwei Plugins für je Import (*GT2ReaderPlugin*) und Export (*GT2WriterPlugin*) von Geodatensätzen, eine grafischen Oberfläche für die Auswahl eines Datenformates und die Eingabe von Parametern (*GT2DataSourceFactoryChooser*) und ein Konverter für Geodatenlisten (*FC-Converter*). Das Klassendiagramm 4.7 auf der nächsten Seite verdeutlicht diesen Aufbau. Der Konverter konnte direkt aus der Test-Extension im Abschnitt 4.3.2 übernommen werden.
Die Plugins teilen sich eine abstrakte elterliche Klasse. Dies liegt in der ähnlichen Verfahrensweise beider Plugins begründet. Der Auswahl der Datenquelle und der Eingabe von Parametern (Sequenzdiagramm 4.8 auf Seite 65, Schritt 1) liegt dieselbe Routine zugrunde.
Das Sequenzdiagramm 4.8 auf Seite 65 stellt den programmatischen Ablauf des Ladens von Geodaten dar. Verallgemeinert repräsentiert durch das Objekt Workbench startet das JUMP-System das Plugin. Daraufhin wird die vom grafischen Dialog abgeleitete Klasse GT2DatasourceFactoryChooser angezeigt. Je nach Datenquellentyp[30] werden unterschiedliche Eingabemasken angezeigt. Entweder muss der Nutzer eine Datei auswählen oder die Daten für den Verbindungsaufbau zur Datenbank eintragen. Sollten die eingegebenen Parameter keine Erstellung eines Datenquellobjektes ermöglichen, wird der Nutzer darauf hingewiesen. Im zweiten Schritt (wird nur ausgeführt, sollte der erste nicht abgebrochen worden sein) werden die Daten gelesen, konvertiert, und als neue Ebene in die Anzeige eingefügt. Der Ablauf beim Schreiben von Daten ist analog. Der maßgebliche

[30] unterstützt werden momentan Dateien und Datenbanken

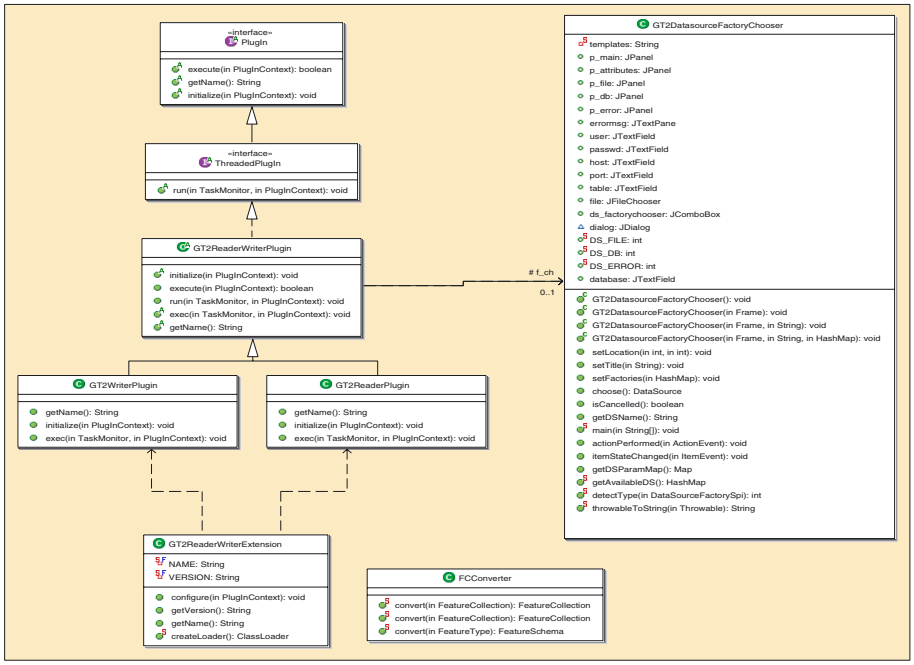


Abbildung 4.7: Klassendiagramm der Im-/Export-Extension

Unterschied ist das Schreiben von Daten im zweiten Schritt. Nach der Extraktion der Daten aus der ausgewählten Ebene, werden diese konvertiert und in die Datenquelle geschrieben.

Auswahl der Eingabemaske

Ein interessanter Teilaspekt der Extension ist die Auswahl der Eingabemaske. Die GT2-Datenquellen-Architektur sieht das automatische Finden von Datenquellmodulen im Pfad vor. Dabei wird der Klassenpfad nach *META-INF/services/org.geotools.data.DataSourceFactorySpi*-Dateien durchsucht. Jedes suchbare GT2-Datenquellmodul enthält einen entsprechende Datei im Java-Archiv. Der Inhalt der Datei ist eine Zeile, die auf die modulspezifische Implementierung des *DataSourceFactorySpi*-Interfaces verweist (z.B. GT2-Shapefile-Module 4.10). Diese Implementierung ist ein Fabrik-Objekt, das in der Lage ist anhand einer Liste von Parametern (HashMap) ein Datenquellobjekt zu erzeugen.

```
org.geotools.data.shapefile.ShapefileDataStoreFactory
```

Listing 4.10: Datei ”META-INF/services/org.geotools.data.DataSourceFactorySpi” des GT2-Shapefile-Moduls

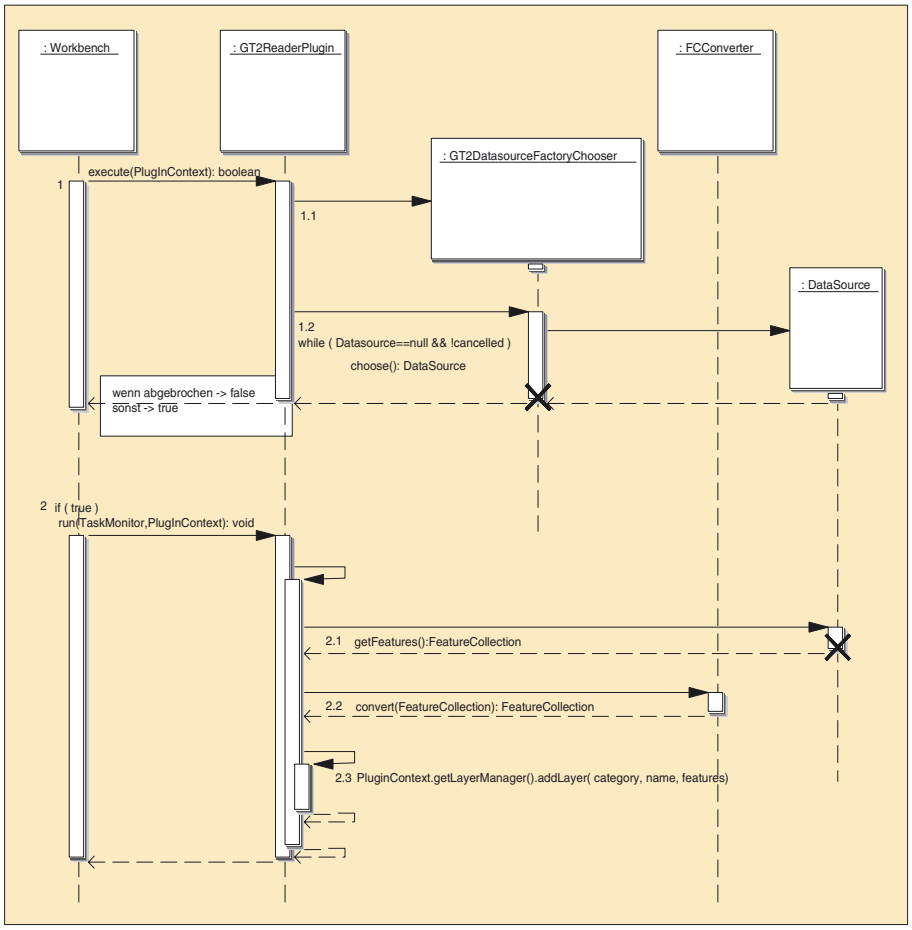


Abbildung 4.8: Sequenzdiagramm der Im-/Export-Extension

Mit dem vorliegenden GT2-Konzept ist es nicht möglich die Natur oder andere Eigenschaften einer Datenquelle zu ermitteln. Einzig Instanzen von Datenquellobjekten können Auskunft über bestimmte Eigenschaften geben. Eine Instanz kann aber nur mit den richtigen Parametern, die der Nutzer (in der Eingabemaske) angeben muss, erzeugt werden. Zur Auswahl der passenden Eingabemaske muss wiederum die Art der Datenquelle bekannt sein. Dieses rekursive Problem ist über einen kleinen Umweg lösbar.
Von der Fabrik kann eine Bezeichnung der Datenquelle eingeholt werden (#getDescription), die eigentlich zur Identifikation einer Datenquell-Objekt-Fabrik in einer Liste mit anderen Datenquellfabriken vorgesehen ist. Eine Kontrolle der aktuellen Implementierungen ergab, dass alle dateibasierten Datenquellen das Wort ”File” und die datenbankbasierten das Wort ”Database” in dieser Bezeichnung enthalten. Die Eingabemaske wurde nun von dem Vergleich dieser Bezeichnungen mit

einem regulären Ausdruck abhängig gemacht und somit das Problem gelöst.

4.5.2 Transformations-Extension

Die umfassendste der drei Extensions besteht aus insgesamt 12 Dateien, von denen die wichtigsten neun hier vorgestellt werden sollen (Klassendiagramm 4.9).

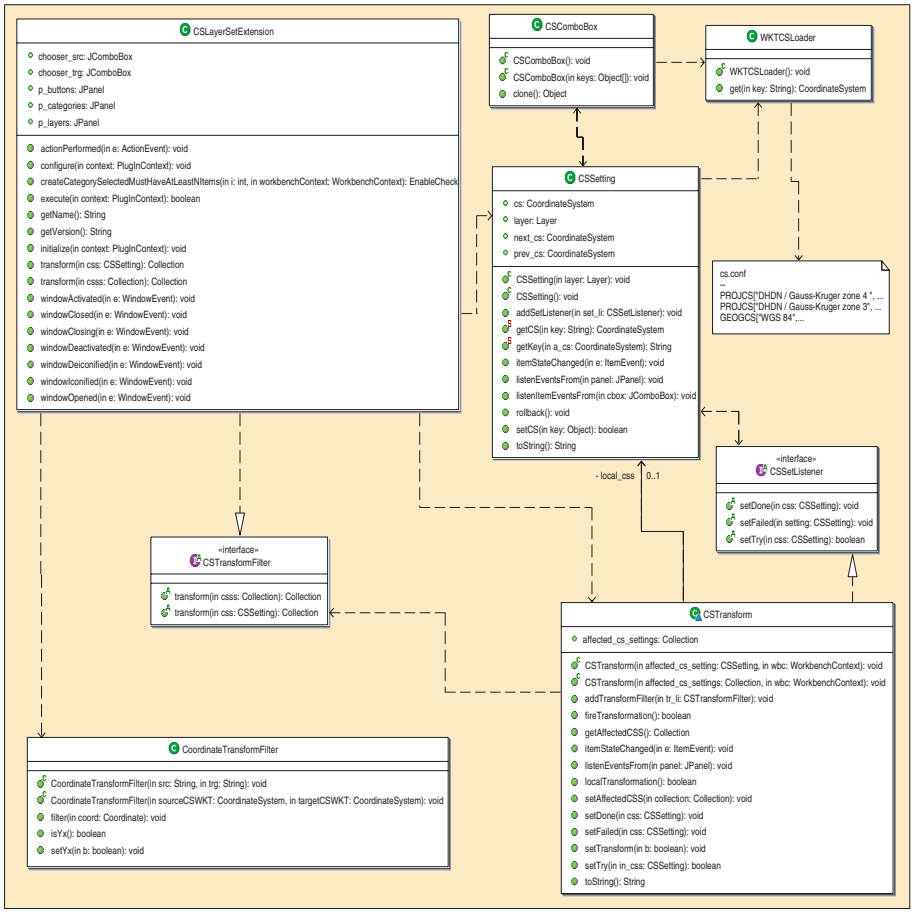


Abbildung 4.9: Klassendiagramm der Transformations-Extension

Ausgangspunkt ist die multifunktionale Klasse *CSLayerSetExtension*. Sie dient gleichzeitig als Extension, Plugin und CoordinateTransformFilter. Extension und Plugin sind bekanntermaßen für Initialisierung und Einbindung in das JUMP-System verantwortlich. Die Implementierung des *CSTransformfilter*-Interfaces ermöglicht es der Klasse, Transformationen an *CSSetting*-Objekten vorzunehmen. Für die Transformation kommt der *CoordinateTransformFilter* aus dem Abschnitt 4.3.3 zum Einsatz. CSSettings repräsentieren das zugewiesene Bezugssystem eines

Layers[31]. Dementsprechend werden sie mit einem Layer-Objekt als Parameter instanziiert. CSSettings implementieren das Swing-Interface ItemListener und können somit bei Java-Swing-Komponenten (z.B. Comboboxes) als Listener registriert werden. *CSComboBox* ist eine solche erweiterte Swing-Komponente[32], die bei Initialisierung alle definierten Bezugssysteme von der Klasse WKTCSLoader abfragt. *WKTCSLoader* lädt diese Daten aus der Datei "cs.conf". Sollte die Datei im Klassenpfad nicht auffindbar sein, erfolgt eine Fehlermeldung und es stehen in CSComboBox keine Bezugssysteme zur Auswahl.
CSSettings unterstützen selbst das Listener-Konzept. Sollte eine Klasse über den Versuch, das Scheitern oder den Erfolg einer Änderung des Bezugssystems eines Layers informiert werden müssen, so kann diese nach Implementierung des *CSSetListener*-Interfaces als entsprechender Listener registriert werden. *CSTransform* ist eine solche Implementierung und nutzt wiederum die *CSTransformFilter*-Implementierung der Extension Klasse für die eigentliche Transformation. Das Konzept ist offen. Es können CSTransform mehrere (oder andere) Filter für die eigentliche Verarbeitung zugewiesen werden, welche dann nacheinander abgearbeitet werden.
Der Funktionsablauf dieser Extension ist sehr komplex und wird deshalb stark vereinfacht vorgestellt. Er kann in zwei Unterschritte aufgeteilt werden: Registrierung und Bearbeitung.

Die Registrierung (vereinfacht dargestellt in Sequenzdiagramm 4.10) initialisiert die benötigten Komponenten für den zweiten Schritt. Einmal initialisiert, werden diese für den nächsten Aufruf des Plugins wiederverwendet. Wie im Diagramm ersichtlich, wird in diesem Schritt der Dialog zur späteren Interaktion mit dem Benutzer erzeugt. Danach werden pro ausgewählte Ebene die benötigten Objekte zur Verwaltung und Transformation erzeugt. Dabei werden jeweils eine CSComboBox, ein CSSetting- und ein CSTransform-Objekt erzeugt. Das CSSetting-Objekt wird am CSComboBox-Objekt und das CSTransform-Objekt entsprechend am CSSetting-Objekt als Listener registriert. CSTransform erhält den Transformations-Filter CSLayerSetExtension zugewiesen. Abschliessend werden die CSComboBoxes als Elemente in den grafischen Dialog eingefügt und dieser angezeigt. Sollte der Nutzer den Dialog schließen, wird das Plugin beendet.

Die Bearbeitung (Sequenzdiagramm 4.11 auf Seite 69) findet während der Darstellung des Dialoges statt. Über die Auswahl eines ersten Bezugsystems, in dem die Ebene momentan vorliegt, wird das Quellbezugssystem durch

[31] dt. Ebene - meint in diesem Fall das Java-Objekt, welches eine Feature-Collection in JUMP enthält und visualisiert
[32] abgeleitet vom DropDown-Listenelement JCombobox des Java-Swing-Paketes

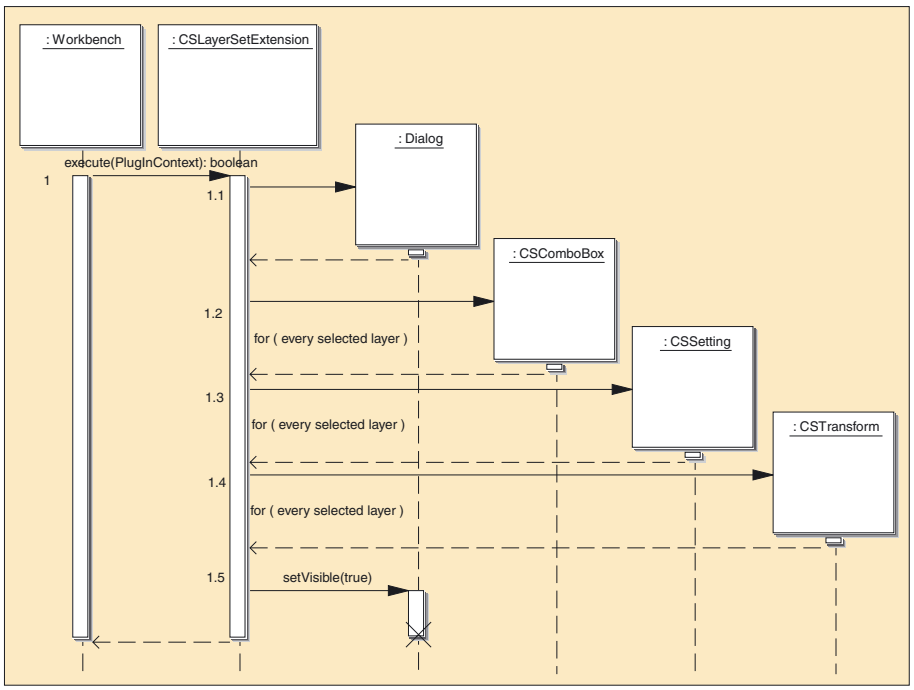


Abbildung 4.10: Sequenzdiagramm 1 der Transformations-Extension - Registrierung

den Nutzer festgelegt. Daraufhin werden im Hintergrund nacheinander CSSetting und CSTransform informiert. Die Einstellung wird kontrolliert und angenommen oder abgewiesen, wobei die ursprüngliche Einstellung wiederhergestellt wird. Normalerweise ist die Zuweisung des Quellbezugssystems fehlerfrei, da damit keine Transformation verbunden ist. Im darauf folgenden Schritt kann nun ein anderes Bezugssystem, das Zielbezugsystem, aus der Liste gewählt werden und analog zum ersten Schritt werden wieder die interessierten Komponenten benachrichtigt. Da diesmal zwei Bezugsysteme definiert sind, kann eine Transformation stattfinden. Sollten sich Fehler ereignen, wird der komplette Prozess rückgängig gemacht und die Anzeige in der CSComboBox zurückgesetzt.

Undo/Redo-Funktion

JUMP unterstützt das Undo/Redo-Modell aus dem Java-Swing-Paket. In diesem existiert ein globaler Manager, der Implementierungen des Interfaces *UndoableEdit* registrieren kann und in der Reihenfolge der Registrierung rückgängig macht oder erneut ausführt. Diese Implementierungen definieren die Methoden #undo und #redo, welche die entsprechenden Aktionen auslösen sollten. Ein Plugin wird

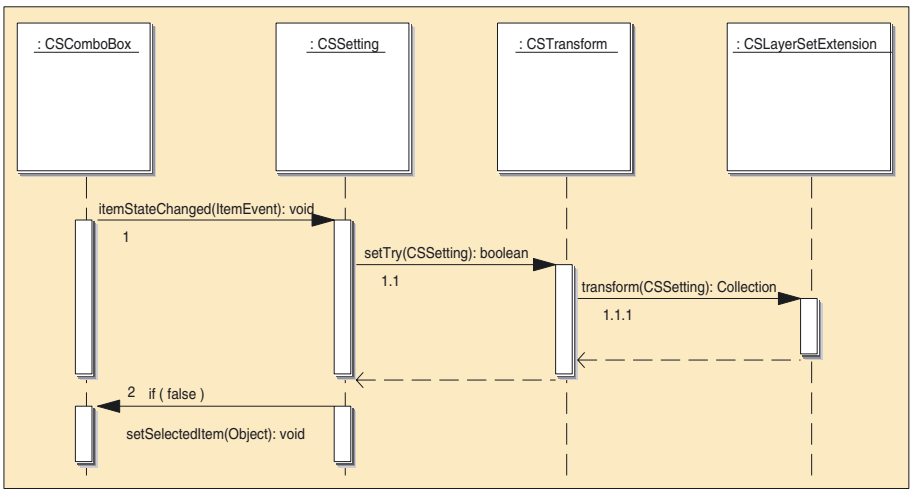


Abbildung 4.11: Sequenzdiagramm 2 der Transformations-Extension - Bearbeitung

undoable, wenn es seine Aktionen in solchen Implementierungen umsetzt und diese mit dem Manager registriert.

```
273         UndoableSetGeometry transformation =
274           new UndoableSetGeometry(
275             "Transformation: ",
276             css.layer.getName()
277               + "("
278               + CSSetting.getKey(css.cs)
279               + "->"
280               + CSSetting.getKey(css.next_cs)
281               + ")",
282             css);
283
284         for (Iterator iter = features.iterator(); iter.hasNext();) {
285           Feature feature = (Feature) iter.next();
286
287           Geometry geom = transformation.getGeom(feature);
288           geom.apply(transfilter);
289           geom.geometryChanged();
290           transformation.setGeom(feature, geom);
291         }
```

Listing 4.11: Auszug aus CSLayerSetExtension.java

Teil der Transformations-Extension ist die Klasse *UndoableSetGeometry*. Ihre Aufgabe ist die Zuweisung von neuen (veränderten) Geometrien zu Features. Gleichzeitig wird die alte Originalgeometrie zwischengespeichert, um bei Bedarf (#undo) diese wiederherstellen zu können. Der Vorteil einer Kapselung dieses Vorgangs ist, dass notwendige Aktionen, wie das Auslösen von Events im JUMP-System zur Aktualisierung der Anzeige o.ä., hier zentral platziert werden können. Die Geometrie kann von verschiedenen Komponenten modifiziert werden, die spezifische Details nicht mehr implementieren müssen. Weiterhin können die Änderung mit dem globalen Manager registriert werden, woraufhin die Bearbeitung für den Nutzer leicht rückgängig zu machen ist oder wiederholt werden kann. UndoableSetGeo-

metry implementiert ausserdem das Interface *java.util.Collection* (dt. Sammlung) des Standard-Java-Paketes. Diese Funktionalität ist in die Undoable-Funktion integriert. Ein einzelnes UndoableSetGeometry-Objekt kann also eines oder mehrere UndoableSetGeometry-Objekte repräsentieren, die eine komplexe Aktion bilden und ausgeführt oder komplett rückgängig gemacht werden können. Die Verwendung wird in Methode #transform der Klasse CSLayerSetExtension illustriert (Listing 4.11 auf der vorherigen Seite).

4.5.3 GPS-Extension

Die GPS-Extension realisiert mit der Positionsbestimmung die Schlüsselfunktion des mobilen GIS. Sie besteht aus acht Komponenten in sechs Dateien. Im Klassendiagramm 4.12 werden diese vorgestellt.

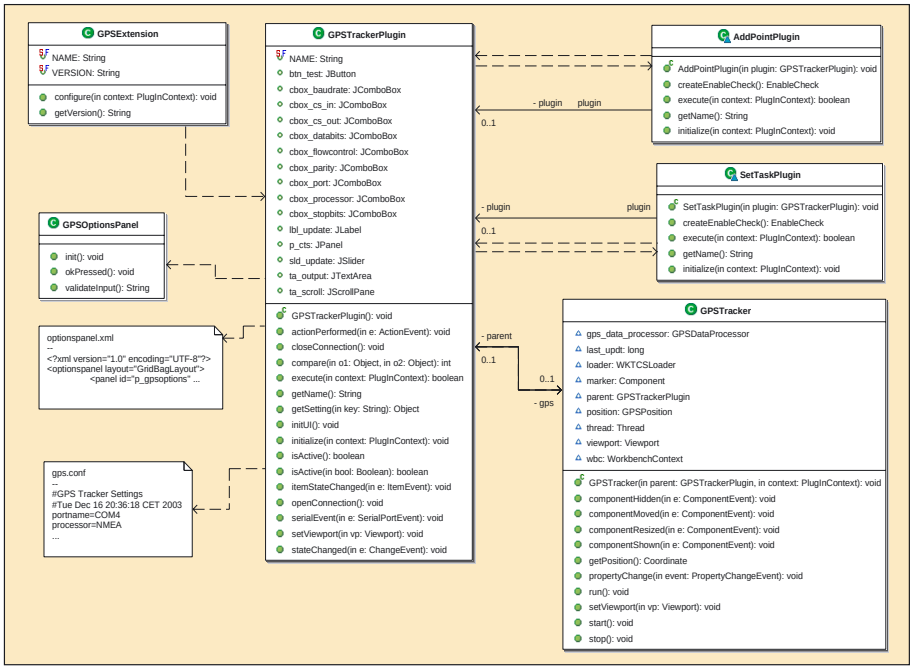


Abbildung 4.12: Klassendiagramm der GPS-Extension

Die *GPSExtension*-Klasse initialisiert das *GPSTrackerPlugin*. Dieses wiederum enthält (die privaten Klassen) *AddPointPlugin* und *SetTaskPlugin*, Hilfs-Plugins, die zwei weitere Funktionen des elterlichen Plugins in die grafische JUMP-Oberfläche einbinden (Abbildung 4.14 auf Seite 72). Die Klasse *GPSTracker* ist vom Java-Interface Runnable abgeleitet und kann als Parallelprozess (Thread) gestartet werden. Sie ist die Implementierung der Positionsbestimmung mit der Java-GPS-API aus Abschnitt 4.3.4. Die Einstellungen der Extension werden im *GPSOp-*

tionsPanel vorgenommen. Diese Klasse implementiert das JUMP-Interface OptionsPanel und kann somit in den vorhandenen Options-Dialog der Anwendung integriert werden. Die Einstellungen werden zwischen einzelnen Anwendungssitzungen in der Datei *gps.conf* gespeichert. Diese sollte sich im Klassenpfad befinden oder der Nutzer wird auf dieses Problem hingewiesen und keine Einstellung geladen.
Der Funktionsablauf der GPS-Extension ist im Sequenzdiagramm 4.13 dargestellt. Das Diagramm besteht aus drei Schritten, die den Ablauf der Funktionen jedes einzelnen Plugins darstellen. Durch die Zusammenfassung soll die Abhängigkeit der Plugins AddPointPlugin und SetTaskPlugin vom GPSTrackerPlugin verdeutlicht werden.

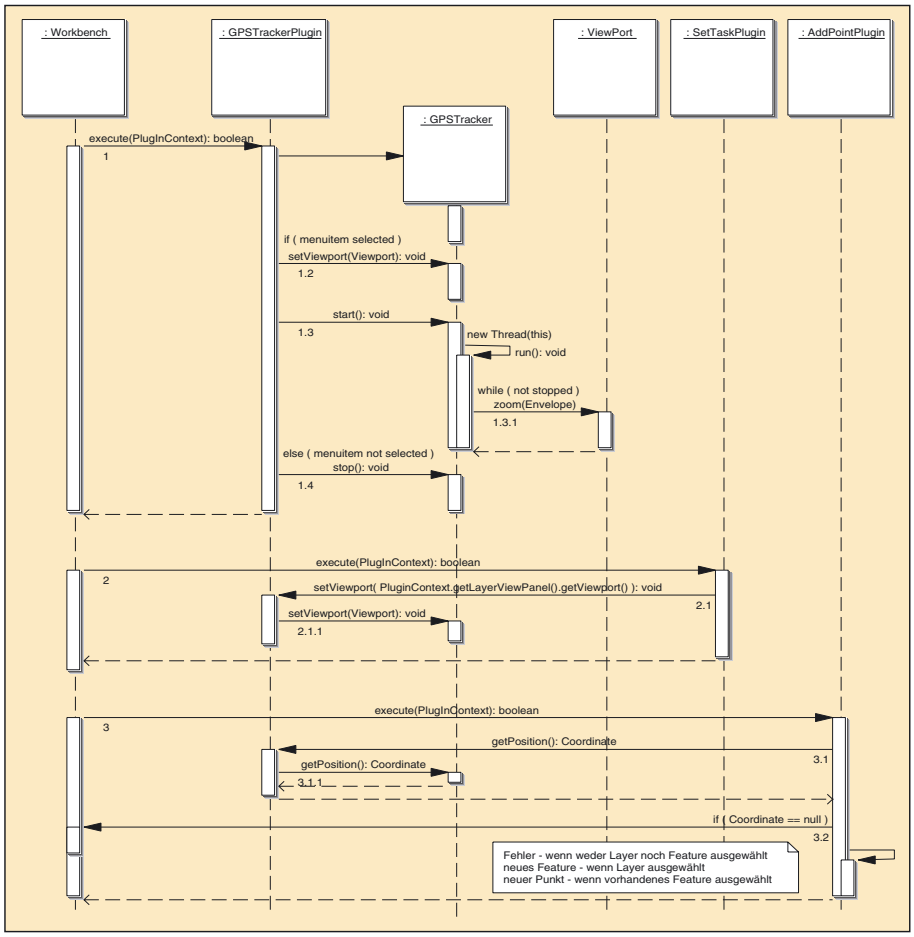


Abbildung 4.13: Sequenzdiagramm der GPS-Extension

Im Schritt 1 wird das GPSTrackerPlugin gestartet und erzeugt (sofern nicht schon vorhanden) ein GPSTracker-Objekt (Schritt 1.1). Danach wird diesem Objekt der

Viewport des aktiven Task-Fensters der JUMP-Anwendung (aus dem Plugincontext) zugewiesen (Schritt 1.2, #setViewPort) und das Tracking gestartet (Schritt 1.3, #start). Das GPSTracker-Objekt erzeugt daraufhin ein Kreuz in der Mitte des Viewports und in einer unendlichen Schleife wird dieser an die aktuelle Position versetzt. Der aktuelle Zoomfaktor bleibt erhalten. Die Frequenz ist als Sekundenintervall zusammen mit den anderen Parametern zur GPS-Extension im Optionsmenü definierbar (Abbildung 4.14).

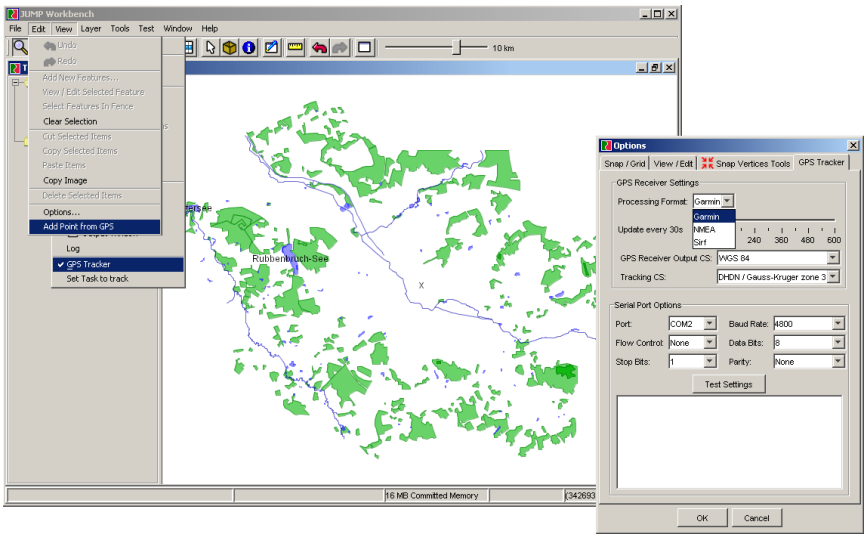


Abbildung 4.14: Optionen der GPS-Extension

Über einen erneuten Aufruf des Menüpunktes ”GPS Tracker” kann das Tracking gestoppt werden (Schritt 1.4, #stop). Ist der GPSTracker inaktiv, sind auch die Menüpunkte ”Edit–>Add Point from GPS” und ”View–>Set Task to Track” deaktiviert. Die Ausführung dieser Funktionen ist nur möglich, wenn auch Positionsdaten empfangen werden.

Der Schritt 2 demonstriert die Änderung des Viewports (SetTaskPlugin). Während das GPSTracker-Objekt aktiv ist, wird ein anderer Viewport gesetzt (Schritt 2.1.1, #setViewPort) und dann anstelle des alten mit der aktuellen Position synchronisiert (Schritt 1.3.1, #zoom).

Schritt 3 zeigt die Funktionsweise des AddPointPlugins auf. Über die Methode #getPosition in Schritt 3.1 wird die aktuelle Position vom GPSTracker bezogen. Das Ergebnis ist eine Koordinate in dem Bezugssystem, das als Zielbezugssystem (Tracking CS) im Optionendialog angegeben ist. Die darauf folgende Aktion (Schritt 3.2) ist abhängig von den in der grafischen Oberfläche ausgewählten Objekten. Ist eine Ebene markiert, aber kein existierendes Feature, wird ein neues Feature in der markierten Ebene erzeugt und der Dialog zur Bearbeitung der At-

tribute öffnet sich. Ist ein Feature markiert, wird der Geometrie dieses Features lediglich ein Punkt an der aktuellen Koordinate hinzugefügt. Sollten weder Ebene noch Feature markiert sein, erhält der Nutzer einen Hinweis über die Funktionsweise des Plugins.

Grafische Oberflächen mit SwiXml[33]

Für das Optionspanel musste eine relativ komplexe grafische Oberfläche realisiert werden. Um diese Aufgabe zu erleichtern kam SwiXml zum Einsatz. SwiXml ist eine 49kByte kleine API, mit der grafische Oberflächen aus XML-Dateien erzeugt werden können. Diese Beschreibungsdateien werden zur Laufzeit gelesen und die darin definierten Swing-Komponenten erzeugt. Die Open-Source-API vereinfacht durch diese Vorgehensweise hauptsächlich die Erstellung grafischer Oberflächen. XML-Code ist wesentlich besser lesbar als Java-Quellcode. Änderungen oder Design-Tests sind dadurch leichter durchführbar und Fehler werden früher erkannt. Durch das späte Einbinden der grafischen Oberfläche können auch verschiedene Gestaltungen einer Anwendung (engl. Themes) sehr leicht realisiert werden. Die funktionalen Elemente bleiben dieselben, nur Anordnung und Gestaltung ändern sich. Nachteilig ist wieder einmal die mangelnde Dokumentation. Es existieren Quellcodekommentare, die auch als fertige Java-Docs vorliegen und einige Beispiele, welche die Funktionalität erläutern. Doch aufgrund der Einfachheit und der sich wiederholenden Syntax ist die Einarbeitung in kürzester Zeit möglich. Am Beispiel des Optionspanels soll die Funktionsweise demonstriert werden.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <optionspanel layout="GridBagLayout">
3    <panel id="p_gpsoptions" layout="GridBagLayout">
4      <gridbagconstraints insets="1,5,1,5" gridx="0" gridy="0"/>
5
6      <panel layout="GridBagLayout" border="TitledBorder(EtchedBorder,GPS Receiver Settings)">
7        <gridbagconstraints gridx="0" gridy="0" insets="5,5,5,5"/>
8
9        <panel layout="GridBagLayout">
10         <gridbagconstraints gridx="0" gridy="0"/>
11           <label text="Processing Format:">
12             <gridbagconstraints gridx="0" gridy="0" insets="5,5,5,5" anchor="GridBagConstraints.WEST
                 "/>
13           </label>
14           <combobox id="cbox_processor">
15             <gridbagconstraints gridx="1" gridy="0" anchor="GridBagConstraints.WEST"/>
16           </combobox>
17
18           <label id="lbl_update" text="Update">
19             <gridbagconstraints gridx="0" gridy="1" insets="5,5,5,5" anchor="GridBagConstraints.WEST
                 "/>
20           </label>
21           <slider id="sld_update" Minimum="0" Maximum="600" MinorTickSpacing="60" MajorTickSpacing="
                 120" PaintLabels="true" PaintTicks="true" PreferredSize="200,50">
22             <gridbagconstraints gridx="1" gridy="1"/>
23           </slider>
24       </panel>
25
26       <panel id="p_cts" layout="GridBagLayout">
```

[33] http://www.swixml.org

```
27          <gridbagconstraints gridx="0" gridy="1"/>
28          <label text="GPS Receiver Output CS:">
29            <gridbagconstraints gridx="0" gridy="0" insets="5,5,5,5" anchor="GridBagConstraints.WEST"/
                >
30          </label>
31          <combobox id="cbox_cs_in">
32            <gridbagconstraints gridx="1" gridy="0" gridwidth="3" fill="GridBagConstraints.HORIZONTAL"
                />
33          </combobox>
34          <label text="Tracking CS:">
35            <gridbagconstraints gridx="0" gridy="1" insets="5,5,5,5" anchor="GridBagConstraints.WEST"/
                >
36          </label>
37          <combobox id="cbox_cs_out">
38            <gridbagconstraints gridx="1" gridy="1" gridwidth="3" fill="GridBagConstraints.HORIZONTAL"
                />
39          </combobox>
40        </panel>
41
42      </panel>
```

Listing 4.12: Auszug aus optionspanel.xml

Der Auszug aus optionspanel.xml (Listing 4.12 auf der vorherigen Seite) z.B. definiert die oberen beiden Abschnitte des Optionspanels mit den Parametern für den GPS-Datenprozessor (GPS Receiver Settings im Optionendialog, Abbildung 4.14). Ein einfaches Vertauschen der Panel-Definitionen in den Zeilen 9–24 und 26–40 und natürlich die Anpassung der GridBagConstraints vertauscht die Position der beiden Panels im Dialog ohne eine Zeile Java-Quellcode verändern zu müssen. Die Syntax der XML-Beschreibungen (engl. Descriptors) ist leicht nachvollziehbar. Eine Anzahl von Swing-Klassen stehen als Tags zur Verfügung. Das Tag `<panel>` z.B. repräsentiert die Swing-Klasse *Jpanel*. Die Liste der vordefinierten Klassen befindet sich in *org.swixml.SwingTagLibrary*. Sollte wie im Fall des Optionspanels eine andere Klasse benötigt werden, kann diese zusätzlich registriert werden (Listing 4.13, Zeile 172).

```
71    private static String TEMPLATES = "de/soldin/gt2jump/gps/";
72    private SwingEngine swix = new SwingEngine(this);
73    public JSlider sld_update;
74    public JLabel lbl_update;
75    public JPanel p_cts;
76    public JComboBox cbox_processor,
77      cbox_cs_in,
78      cbox_cs_out,
79      cbox_port,
80      cbox_baudrate,
81      cbox_flowcontrol,
82      cbox_databits,
83      cbox_stopbits,
84      cbox_parity;
85    public JScrollPane ta_scroll;
86    public JTextArea ta_output;
87    public JButton btn_test;
88    private JCheckBoxMenuItem menuitem;
        ...
172     this.swix.getTaglib().registerTag("optionspanel", GPSOptionsPanel.class);
        ...
185       GPSOptionsPanel panel =
186         (GPSOptionsPanel) swix.render(TEMPLATES + "optionspanel.xml");
```

Listing 4.13: SwiXml in Aktion - Auszug aus GPSTrackerPlugin.java

Mit den Attributen der XML-Tags werden die Eigenschaften der Swing-Objekte gesetzt. Nach der Erfahrung des Autors sind die meisten öffentlichen (public) und die per Getter/Setter veränderbaren Variablen der Swing-Objekte mit dieser Methode veränderbar. Als Wert werden Zeichenketten oder Zahlenwerte angegeben. Die API scheint diese auszuwerten und in gültiger Form zu übergeben. Das Statement `border="TitledBorder(EtchedBorder,GPS Receiver Settings)"` setzt z.B. die Eigenschaft Border (#setBorder) des betreffenden Panels auf ein neues TitledBorder-Objekt, das mit den angegebenen Parametern instanziiert wurde.
Es gibt zwei Methoden die SwiXml-API in den eigenen Code einzubinden.

1. Die betreffende Klasse wird von *org.swixml.SwingEngine* abgeleitet.
2. Die Erzeugung einer Instanz, der die nutzende Klasse im Konstruktor übergeben wird (Listing 4.13, Zeile 72).

Die in der XML-Datei mit dem `id`-Attribut gekennzeichneten Swing-Komponenten sollten in der nutzenden Klasse unter diesem Namen definiert sein (z.B. `id="p_cts"` im Descriptor und `public JPanel p_cts;` in der nutzenden Klasse). Kann die SwingEngine darauf zugreifen, werden diese Objekte zugewiesen und stehen für die weitere Nutzung zur Verfügung.
Die SwiXml-API wurde auch für die anderen beiden Extensions erfolgreich verwendet.

4.6 Lizenzierung

Das Produkt dieser Arbeit sind drei eigenständige Erweiterungen für JUMP. Diese basieren sowohl auf LGPL lizensiertem (GT2, GPSTool) als auch auf GPL lizenziertem (JUMP) Quellcode. Nach den Bedingungen der restriktiveren Copyleft-Lizenz GPL muss eine Anwendung, die auf GPL'ed Quellcode aufbaut (z.B. Ableitung von Klassen) selbst wieder der GPL unterliegen. Um diese Lizenzierung kenntlich zu machen sind folgende Einträge in jeder Quelldatei notwendig[34]:

1. ein Urheberrechtshinweis (engl. Copyright Notice) z.B. "Copyright 2004 Edgar Soldin."
2. eine Kopiererlaubnis (engl. Copying Permission Statement) wie im Anhang A.6 direkt nach dem Urheberrechtshinweis.

Die entsprechenden Hinweise werden den Quellen vor der ersten Veröffentlichung hinzugefügt. Desweiteren wird eine Kopie der GNU GPL als Datei im Plain-Text-Format Teil der Distribution sein.
Die zusätzlich verwandte SwiXml-API unterliegt einer eigenen Open-Source-Lizenz (Anhang A.7 auf Seite XV), die für binäre Verteilung (engl. Distribution) vorsieht, dass

1. der Urheberrechtshinweis in der SwiXml-Lizenz,
2. die Liste der Lizenzbedingungen aus der SwiXml-Lizenz und
3. die Erklärung (engl. Disclaimer) zur Gewährleistung

in der Dokumentation oder anderen Dateien angegeben werden. Für den vorliegenden Fall sollte eine Kopie der SwiXml-Lizenz-Datei (swixml-license.txt) im selben Verzeichnis wie die Jar-Datei (swixml.jar) den Bedingungen genügen.

[34] (Vgl. GNU-Project 2004, How to use the GPL or LGPL)

Kapitel 5

Zusammenfassung

Die Erarbeitung der Grundlagen hat aufgezeigt, dass sich die Geoinformatik noch am Anfang ihrer Entwicklung befindet. Diese gewisse Unreife der Basis verursacht Probleme, die zu einem unnötigen Mehraufwand in der Herstellung des mobilen GIS geführt haben. Folgende Probleme konnten festgestellt werden:

- inkompatible Software aufgrund fehlender Standards (Abschnitt 4.3.2)
- mangelnde Dokumentation aufgrund des frühen Entwicklungsstadiums (allgemein Abschnitt 4.3)
- eingeschränkter Funktionsumfang aufgrund des frühen Entwicklungsstadiums oder eingestellter Entwicklung (einige Open-Source-Projekte)

Die in der Entwicklung befindlichen offenen Standards (Abschnitt 2.2.2) sollten diese Probleme in naher Zukunft relativieren. Desweiteren kann aufgrund des (immer noch) steigenden Interesses der Wirtschaft an GIS-Technologien[1] und Open-Source[2] gleichermaßen ein Zuwachs an Qualität, Dokumentation und Leistungsfähigkeit im Softwarebereich erwartet werden.
Die eingeschränkte Leistungsfähigkeit mobiler Computer (Problem Laufzeit vs. Rechenleistung/Speicher, Abschnitt 4.1.5) ist dagegen eine Problematik für die im Moment keine Abhilfe in Sicht ist. Die Produktion hochspezialisierter Lösungen, ”die benötigte Information zur richtigen Zeit am richtigen Ort verfügbar [...] machen - und das 100-prozentig verlässlich”[3] kann eine Lösung darstellen. Der Ansatz dieser Arbeit hingegen ist im entgegengesetzten Extrem angesiedelt. Der Verzicht auf eine Online-Datenanbindung und ein generalisierter Satz Funktionen sollen den Einsatz des mobilen GIS für viele Zwecke (Erfassung, Bearbeitung,

[1] (Abschnitt 2.2.1)
[2] (Abschnitt 2.4.3)
[3] (Ranzinger: Einsatz mobiler GIS bei Energieversorgungsunternehmen in den Bereichen Instandhaltung und Störungsmanagement, in: Zipf, Strobl 2002, S.209)

Transfer etc.) und an vielen Orten (Büro, Außeneinsatz u.a.) ermöglichen. Die dadurch wesentlich vergrößerte Zielgruppe macht a) die (Weiter-)Entwicklung für Dritte attraktiver und sorgt b) wahrscheinlich für eine höhere Verbreitung. Dank der (relativen) Plattformunabhängigkeit einer Java-Anwendung ist die Wahrscheinlichkeit hoch, dass das mobile GIS auf der leistungsfähigeren mobilen Hardware der Zukunft lauffähig ist.
Die durchschnittliche Ungenauigkeit (ca. 10 Meter bei Stillstand) der Positionsbestimmung im zivilen Bereich (Abschnitt 2.3.1) kann allerdings ohne weitere Anpassungen nicht gesenkt werden. Damit wird die Anwendung entgegen der ursprünglichen Zielsetzung für einige Einsatzformen ungeeignet sein. Der Einsatz unterstützender Systeme, wie RTK/DGPS oder WAAS, ist möglich, bedarf allerdings einer Anpassung der GPS-Erweiterung des mobilen GIS, die den Rahmen der Diplomarbeit bei weitem überschritten hätte.

Zusammenfassend kann gesagt werden, dass die Vorteile der mobilen GIS-Technologie[4] schon heute eine Auseinandersetzung mit dem Thema rechtfertigen. Die Untersuchung existierender Abläufe in einem Unternehmen oder einer Verwaltung auf die Anwendbarkeit von GIS kann Möglichkeiten zur Einsparung von Zeit, Geld und Nerven aufzeigen. Liegt Einsparpotenzial vor, ist die Beobachtung der aktuellen Entwicklungen anzuraten. Fehlende Standards, kaum ausgereifte Open-Source-Produkte[5] und eine Handvoll marktdominierender Hersteller von proprietärer GIS-Software lassen jedoch die Anschaffung von GIS zum jetzigen Zeitpunkt riskant erscheinen. Dem Erwerb, dem Einsatz oder der Entwicklung von GIS-Systemen sollte dementsprechend eine kritische Kosten-Nutzen-Analyse vorausgehen. Erst mittelfristig kann mit der zunehmenden Verbreitung von Standards und einer größeren Auswahl ausgereifter Open-Source-Produkte gerechnet werden, so dass GIS als kostengünstiges Werkzeug (z.B. zur Inventarisierung oder Planung des Vorgartens) für jedermann einsetzbar wird.

Ergebnis

Die Zielsetzung (Seite 4) eine funktionstüchtige mobile GIS-Anwendung herzustellen, kann als erreicht betrachtet werden. Auch die Umsetzung als freie Software unter Einbindung offener Standards ist gelungen (Abschnitt 3.1). Das mobile GIS ist dem Prototypen-Stadium entwachsen. Kleinere Fehler und Unzulänglichkeiten, die auf den kurzen Entwicklungszeitrahmen (sechs Wochen) und die Reife der zugrundeliegenden Komponenten zurückzuführen sind, können durchaus Teil der ersten öffentlichen Version sein. Sie werden im Laufe der nächsten Monate behoben.

[4] (Abschnitt 3.1.5)
[5] auffällige Ausnahme ist das GRASS-Projekt

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Korrespondenzverzeichnis

Anhang A

Listings

Listing A.1: TestPlugin.java

```
package de.soldin.gt2jump.test.jumpplugin;
import com.vividsolutions.jump.workbench.plugin.EnableCheck;
import com.vividsolutions.jump.workbench.plugin.EnableCheckFactory;
import com.vividsolutions.jump.workbench.plugin.MultiEnableCheck;
import com.vividsolutions.jump.workbench.plugin.PlugIn;
import com.vividsolutions.jump.workbench.plugin.PlugInContext;
import com.vividsolutions.jump.workbench.ui.images.IconLoader;
import com.vividsolutions.jump.workbench.ui.plugin.FeatureInstaller;

public class TestPlugin
  implements PlugIn
  {
  public static final String NAME = "Test Plugin";
  public static final String VERSION = "1.0";

  public void initialize(PlugInContext context){

    FeatureInstaller featureInstaller = context.getFeatureInstaller();
    EnableCheckFactory checkFactory = new EnableCheckFactory(context.getWorkbenchContext());

    // create a check object for main menu item
    // conditions must be fulfilled, or item is disabled
    EnableCheck savecheck =  new MultiEnableCheck()
      .add(checkFactory.createWindowWithLayerNamePanelMustBeActiveCheck());

    // add item to main menu 'Test' with a check object
    featureInstaller.addMainMenuItem(
      this,
      "Test",
      NAME,
      null,
      savecheck);

    // add item to category popup menu with a check object
    context.getFeatureInstaller().addPopupMenuItem(
      context.getWorkbenchContext().getWorkbench().getFrame().getCategoryPopupMenu(),
      this,
      NAME,
      false,
      null,
      new MultiEnableCheck().add(checkFactory.createAtLeastNCategoriesMustBeSelectedCheck(1))
    );

    // add item to layer popup menu w/o conditions
    featureInstaller.addPopupMenuItem(
      context.getWorkbenchContext().getWorkbench().getFrame().getLayerNamePopupMenu(),
      this,
      NAME,
      false,
      IconLoader.icon("Palette.gif"),
      new MultiEnableCheck()
    );

    // tell everybody
    System.out.println(getName()+" initialize");
  }

  public boolean execute(PlugInContext context) throws Exception {

    // tell everybody GUI
    context.getWorkbenchFrame().getOutputFrame().createNewDocument();
    context.getWorkbenchFrame().getOutputFrame().addText(getName()+" execute");
    context.getWorkbenchFrame().getOutputFrame().surface();

```

```
65      // tell everybody Commandline
66      System.out.println(getName()+" execute");
67
68      return true;
69    }
70
71    public String getName() {
72      return NAME;
73    }
74
75  }
```

Listing A.2: TestThreadedPlugin.java

```
1   package de.soldin.gt2jump.test.jumpplugin;
2   import com.vividsolutions.jump.task.TaskMonitor;
3   import com.vividsolutions.jump.workbench.plugin.EnableCheck;
4   import com.vividsolutions.jump.workbench.plugin.EnableCheckFactory;
5   import com.vividsolutions.jump.workbench.plugin.MultiEnableCheck;
6   import com.vividsolutions.jump.workbench.plugin.PlugIn;
7   import com.vividsolutions.jump.workbench.plugin.PlugInContext;
8   import com.vividsolutions.jump.workbench.plugin.ThreadedPlugIn;
9   import com.vividsolutions.jump.workbench.ui.images.IconLoader;
10  import com.vividsolutions.jump.workbench.ui.plugin.FeatureInstaller;
11
12  public class TestThreadedPlugin
13    implements ThreadedPlugIn
14    {
15    public static final String NAME = "Test Threaded Plugin";
16    public static final String VERSION = "1.0";
17
18    public void initialize(PlugInContext context){
19
20      FeatureInstaller featureInstaller = context.getFeatureInstaller();
21      EnableCheckFactory checkFactory = new EnableCheckFactory(context.getWorkbenchContext());
22
23      // create a check object for main menu item
24      // conditions must be fulfilled, or item is disabled
25      EnableCheck savecheck =  new MultiEnableCheck()
26        .add(checkFactory.createWindowWithLayerNamePanelMustBeActiveCheck());
27
28      // add item to main menu 'Test' with a check object
29      featureInstaller.addMainMenuItem(
30        this,
31        "Test",
32        NAME,
33        null,
34        savecheck);
35
36      // add item to category popup menu with a check object
37      context.getFeatureInstaller().addPopupMenuItem(
38        context.getWorkbenchContext().getWorkbench().getFrame().getCategoryPopupMenu(),
39        this,
40        NAME,
41        false,
42        null,
43        new MultiEnableCheck().add(checkFactory.createAtLeastNCategoriesMustBeSelectedCheck(1))
44      );
45
46      // add item to layer popup menu w/o conditions
47      featureInstaller.addPopupMenuItem(
48        context.getWorkbenchContext().getWorkbench().getFrame().getLayerNamePopupMenu(),
49        this,
50        NAME,
51        false,
52        IconLoader.icon("Palette.gif"),
53        new MultiEnableCheck()
54      );
55
56      // tell everybody
57      System.out.println(getName()+" initialize");
58    }
59
60    public boolean execute(PlugInContext context) throws Exception {
61
62      // tell everybody GUI
63      context.getWorkbenchFrame().getOutputFrame().createNewDocument();
64      context.getWorkbenchFrame().getOutputFrame().addText(getName()+" execute");
65      context.getWorkbenchFrame().getOutputFrame().surface();
66
67      // tell everybody Commandline
68      System.out.println(getName()+" execute");
69
70      return true;
71    }
72
73    public void run(TaskMonitor monitor, PlugInContext context) throws Exception {
74      monitor.allowCancellationRequests();
75
76      for (int i = 0; !monitor.isCancelRequested() && i < 1000; i++) {
77        Thread.sleep(1000);
78        // tell everybody GUI
79        monitor.report(getName()+"#run "+new Integer(i).toString());
80        // tell everybody Commandline
81        System.out.println(getName()+"#run "+new Integer(i).toString());
```

```
    }
  }

  public String getName() {
    return NAME;
  }

}
```

Listing A.3: MapViewerEd.java

```
/*
 *    Geotools2 - OpenSource mapping toolkit
 *    (C) 2002, Geotools Project Managment Committee (PMC)
 *
 *    This library is free software; you can redistribute it and/or
 *    modify it under the terms of the GNU Lesser General Public
 *    License as published by the Free Software Foundation;
 *    version 2.1 of the License.
 *
 *    This library is distributed in the hope that it will be useful,
 *    but WITHOUT ANY WARRANTY; without even the implied warranty of
 *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 *    Lesser General Public License for more details.
 */
//import java.lang.reflect.Array;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Iterator;
import java.awt.Color;
import java.io.File;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.awt.BorderLayout;
import java.awt.Container;
import javax.swing.JFrame;

// Geotools dependencies
import org.geotools.map.Layer;
import org.geotools.map.Context;
import org.geotools.map.ContextFactory;
import org.geotools.styling.Style;
import org.geotools.styling.StyleBuilder;
import org.geotools.feature.Feature;
import org.geotools.feature.FeatureCollection;
import org.geotools.data.DataSource;
import org.geotools.data.DataSourceException;
import org.geotools.data.shapefile.ShapefileDataSource;
//import org.geotools.pt.CoordinatePoint;
import org.geotools.pt.MismatchedDimensionException;
import org.geotools.renderer.j2d.RenderedMapScale;
import org.geotools.gui.swing.StyledMapPane;
import org.geotools.gui.swing.StatusBar;

// Eds added imports
import com.vividsolutions.jts.geom.*;
import org.geotools.cs.*;
import org.geotools.ct.CannotCreateTransformException;
import org.geotools.ct.CoordinateTransformation;
import org.geotools.ct.CoordinateTransformationFactory;
import org.geotools.ct.MathTransform;
import org.geotools.ct.TransformException;

/**
 * Load and display a shape file. At the difference of the {@link MapViewer} demo, this demo
 * use {@link MapPane} with the {@linkplain Renderer J2D renderer}. This renderer has the
 * following advantages:
 * <ul>
 *   <li>faster;</li>
 *   <li>progressive rendering of tiled image;</li>
 *   <li>supports arbitrary map projections (different geometries to be rendered on the same
 *       map can have different coordinate systems);</li>
 *   <li>supports zooms, translations and rotations through mouse drag, mouse wheel, keyboard
 *       and contextual menu localized in English, French, Portuguese and partially in Spanish
 *       and Greek;</li>
 *   <li>provides a magnifier (accessible from the contextual menu, right button click);</li>
 *   <li>arbitrary amount of offscreen buffering;</li>
 *   <li>can display scroll bars;</li>
 *   <li>can display a status bar with mouse coordinates in an arbitrary coordinate system
 *       (it doesn't have to be the same coordinate system than the renderer one);</li>
 *   <li>has a more precise scale factor taking in account the physical size of the output
 *       device (when this information is available);</li>
 * </ul>
 *
 * The inconvenient is a more complex renderer, which is more difficult to modify for new users.
 * <br><br>
 * NOTE: While not essential, it is recommanded to run this demos in server mode, with:
 * <blockquote><pre>
 * java -server org.geotools.demos.MapViewer2 <I>thefile.shp</I>
 * </pre></blockquote>
 *
 * @author Martin Desruisseaux
 * @version $Id: MapViewer2.java,v 1.27 2003/08/20 22:04:06 desruisseaux Exp $
 */
public class MapViewerEd {
```

```
85      /**
86       * Run the test from the command line. If arguments are provided, then the first
87       * argument is understood as the filename of the shapefile to load.
88       *
89       * @throws IOException is a I/O error occured.
90       * @throws DataSource if an error occured while reading the data source.
91       */
92      public static void main(final String[] args)
93        throws
94          MismatchedDimensionException,
95          MalformedURLException,
96          IOException,
97          DataSourceException,
98          FactoryException,
99          TransformException {
100       final MapViewerEd viewer = new MapViewerEd();
101       final Context context;
102       switch (args.length) {
103         default : // Fall through
104         case 1 :
105           context = viewer.loadContext(new File(args[0]).toURL());
106           break;
107         case 0 :
108           context = viewer.loadContext();
109           break;
110       }
111       viewer.showMapPane(context);
112     }
113
114     /**
115      * Load the data from the shapefile <code>&quot;testData/statepop.shp&quot;</code>.
116      * This file must be on the class path.
117      *
118      * @return Context The data from the shape file.
119      * @throws FileNotFoundException if the shape file was not found.
120      * @throws IOException is some other kind of I/O error occured.
121      * @throws DataSource if an error occured while reading the data source.
122      */
123     protected Context loadContext()
124       throws
125         MismatchedDimensionException,
126         MalformedURLException,
127         IOException,
128         DataSourceException,
129         FactoryException,
130         TransformException {
131       //return loadContext(getClass().getClassLoader().getResource((new File("G:\\_Projekte\\
              MobileGISApplication\\mobiharz\\vektor_daten\\gemeinde_grenzen.shp")).toURL());
132       return loadContext((new File("gruenflaechen-joined.shp")).toURL());
133     }
134
135     /**
136      * Load the data from the specified shapefile and construct a {@linkplain Context context}
137      * with a default style.
138      *
139      * @param  url The url of the shapefile to load.
140      * @return Context The data from the shape file.
141      * @throws IOException is a I/O error occured.
142      * @throws DataSource if an error occured while reading the data source.
143      */
144     protected Context loadContext(final URL url)
145       throws
146         IOException,
147         DataSourceException,
148         FactoryException,
149         MismatchedDimensionException,
150         TransformException {
151
152       // Load the file
153       if (url == null) {
154         throw new FileNotFoundException("Resource not found");
155       }
156       final DataSource datasource = new ShapefileDataSource(url);
157       final FeatureCollection features = datasource.getFeatures();
158
159       // Ed added: Convert features to WGS84
160       CoordinateSystemFactory csFactory = CoordinateSystemFactory.getDefault();
161       CoordinateSystem sourceCSWKT =
162         csFactory.createFromWKT(WKTFactory.getGaussKruegerZone3());
163       System.out.println(sourceCSWKT.getDimension());
164       CoordinateSystem targetCSWKT =
165         csFactory.createFromWKT(WKTFactory.getGeographicWGS84());
166       System.out.println(targetCSWKT.getDimension());
167       MathTransform transform = this.transformInit(sourceCSWKT, targetCSWKT);
168       CoordinateTransformFilter trans =
169         new CoordinateTransformFilter(sourceCSWKT, targetCSWKT);
170       trans.setYx(true);
171
172       for (Iterator iter = features.iterator(); iter.hasNext();) {
173         String foobar = new String();
174         Feature feature = (Feature) iter.next();
175         Geometry geom = feature.getDefaultGeometry();
176         //transformiere
177         //geom.apply(trans);
178
179         // simple output
180         Object[] attribs = feature.getAttributes(null);
181         for (int i = 0; i < attribs.length; i++) {
182           Object attrib = attribs[i];
```

```
183          foobar += "<" + attrib.toString() + ">";
184        }
185        System.out.println(foobar);
186      }
187
188      // Create the style
189      final StyleBuilder builder = new StyleBuilder();
190      final Style style =
191        builder.createStyle(
192          builder.createPolygonSymbolizer(Color.ORANGE, Color.BLACK, 1));
193
194      // Create the context
195      final ContextFactory factory = ContextFactory.createFactory();
196      final Context context = factory.createContext();
197      final Layer layer = factory.createLayer(features, style);
198
199      layer.setTitle("The shapefile");
200      context.getLayerList().addLayer(layer);
201      context.getBoundingBox().setAreaOfInterest(features.getBounds());
202      context.setTitle("Bi ba butzemann");
203
204      return context;
205    }
206
207    /**
208     * Create and show the map pane.
209     *
210     * @param context The context to show.
211     */
212    protected void showMapPane(final Context context) {
213      StyledMapPane mapPane;
214
215      // Create the map pane and add a map scale layer to it.
216      try {
217        CoordinateSystemFactory csFactory = CoordinateSystemFactory.getDefault();
218        CoordinateSystem sourceCSWKT =
219          csFactory.createFromWKT(WKTFactory.getGaussKruegerZone4());
220        mapPane = new StyledMapPane(sourceCSWKT);
221
222        mapPane.setContext(context);
223        mapPane.setPaintingWhileAdjusting(true);
224        mapPane.getRenderer().addLayer(new RenderedMapScale());
225        //mapPane.setCoordinateSystem(sourceCSWKT);
226      } catch (Exception e) {
227        mapPane = new StyledMapPane();
228        e.printStackTrace();
229      }
230
231      // Create the frame, add the map pane and a status bar.
232      final JFrame frame = new JFrame(context.getTitle());
233      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
234      final Container container = frame.getContentPane();
235      container.setLayout(new BorderLayout());
236      container.add(mapPane.createScrollPane(), BorderLayout.CENTER);
237      container.add(new StatusBar(mapPane), BorderLayout.SOUTH);
238
239      frame.pack();
240      frame.show();
241    }
242
243    private MathTransform transformInit(
244      CoordinateSystem sourceCSWKT,
245      CoordinateSystem targetCSWKT)
246      throws CannotCreateTransformException {
247      CoordinateTransformationFactory trFactory =
248        CoordinateTransformationFactory.getDefault();
249      CoordinateTransformation transformation =
250        trFactory.createFromCoordinateSystems(sourceCSWKT, targetCSWKT);
251      MathTransform transform = transformation.getMathTransform();
252      return transform;
253    }
254
255    private Coordinate transform(Coordinate in) {
256      return new Coordinate();
257    }
258  }
259
260  class CoordinateTransformFilter implements CoordinateFilter {
261    private CoordinateSystem sourceCSWKT, targetCSWKT;
262    private MathTransform transform;
263    private boolean yx = false;
264
265    public CoordinateTransformFilter(
266      CoordinateSystem sourceCSWKT,
267      CoordinateSystem targetCSWKT)
268      throws CannotCreateTransformException {
269      CoordinateTransformationFactory trFactory =
270        CoordinateTransformationFactory.getDefault();
271      CoordinateTransformation transformation =
272        trFactory.createFromCoordinateSystems(sourceCSWKT, targetCSWKT);
273      this.transform = transformation.getMathTransform();
274    }
275
276    public void filter(Coordinate coord) {
277      double[] ord = new double[2];
278      if (this.yx) {
279        ord[0] = coord.y;
280        ord[1] = coord.x;
281      } else {
```

```
      ord[0] = coord.x;
      ord[1] = coord.y;
    }
    try {
      this.transform.transform(
        ord,
        0,
        ord,
        0,
        ord.length / this.transform.getDimSource());
      if (this.yx) {
        coord.setCoordinate(new Coordinate(ord[1], ord[0]));
      } else {
        coord.setCoordinate(new Coordinate(ord[0], ord[1]));
      }

    } catch (Exception e) {
      System.out.println(e.toString());
    }
  }

  public boolean isYx() {
    return this.yx;
  }

  public void setYx(boolean b) {
    this.yx = b;
  }

}
```

Listing A.4: WKTFactory.java

```
/*
 * Created on 24.09.2003
 *
 * To change the template for this generated file go to
 * Window>Preferences>Java>Code Generation>Code and Comments
 */

/**
 * @author ed
 *
 * Dies ist nur eine Hilfsklasse fuer diejenigen, die Koordinatensysteme
 * nicht per Hand oder aus der EPSG-Datenbank erzeugen wollen/koennen. Sind
 * natuerlich nicht alle hier enthalten ;-) Zurueckgeben wird jeweils
 * der WKT-String
 */
public class WKTFactory {

  public static void main(String[] args) {
  }

  public static String getGaussKruegerZone4() {
    return "PROJCS[\"DHDN / Gauss- Kruger zone 4 \", GEOGCS[\" DHDN \", DATUM[\" Deutsches Hauptdreiecksnetz
            \", SPHEROID[\" Bessel 1841 \", 6377397.155, 299.1528128, AUTHORITY[\" EPSG \",\" 7004 \"]],
            TOWGS84[598.1, 73.7, 418.2, 0.2019999999999998, 0.04499999999999995, -2.4549999999999974, 6.7],
            AUTHORITY[\" EPSG \",\" 6314 \"]], PRIMEM[\" Greenwich \", 0.0, AUTHORITY[\" EPSG \",\" 8901 \"]],
            UNIT[\" degree of angle \",0.017453292519943295], AXIS[\" Geodetic latitude \",NORTH], AXIS
            [\" Geodetic longitude \",EAST], AUTHORITY[\" EPSG \",\" 4314 \"]], PROJECTION[\" Transverse_Mercator
            \"], PARAMETER[\" semi_major \", 6377397.155], PARAMETER[\" semi_minor \", 6356078.962818189],
            PARAMETER[\" central_meridian \", 11.999999999999988], PARAMETER[\" latitude_of_origin \", 0.0],
            PARAMETER[\" scale_factor \", 1.0], PARAMETER[\" false_easting \", 4500000.0], PARAMETER[\"
            false_northing \", 0.0], UNIT[\" metre \",1.0], AXIS[\" Northing \",NORTH], AXIS[\" Easting
            \",EAST], AUTHORITY[\" EPSG \",\" 31468 \"]]";
  }

  public static String getGaussKruegerZone3() {
    return "PROJCS[\"DHDN / Gauss-Kruger zone 3\", GEOGCS[\"DHDN\", DATUM[\"Deutsches Hauptdreiecksnetz
            \", SPHEROID[\"Bessel 1841\", 6377397.155, 299.1528128, AUTHORITY[\"EPSG\",\"7004\"]], TOWGS84
            [598.1, 73.7, 418.2, 0.2019999999999998, 0.04499999999999995, -2.4549999999999974, 6.7], AUTHORITY
            [\"EPSG\",\"6314\"]], PRIMEM[\"Greenwich\", 0.0, AUTHORITY[\"EPSG\",\"8901\"]], UNIT[\"degree
            of angle\",0.017453292519943295], AXIS[\"Geodetic latitude\",NORTH], AXIS[\"Geodetic longitude
            \",EAST], AUTHORITY[\"EPSG\",\"4314\"]], PROJECTION[\"Transverse_Mercator\"], PARAMETER[\"semi_major
            \", 6377397.155], PARAMETER[\"semi_minor\", 6356078.962818189], PARAMETER[\"central_meridian
            \", 8.999999999999991], PARAMETER[\"latitude_of_origin\", 0.0], PARAMETER[\"scale_factor\", 1.0],
            PARAMETER[\"false_easting\", 3500000.0], PARAMETER[\"false_northing\", 0.0], UNIT[\"metre\",1.0],
            AXIS[\"Northing\",NORTH], AXIS[\"Easting\",EAST], AUTHORITY[\"EPSG\",\"31467\"]]";
  }

  public static String getGeographicWGS84() {
    return "GEOGCS[\"WGS 84\", DATUM[\"World Geodetic System 1984\", SPHEROID[\"WGS
            84\", 6378137.0, 298.257223563, AUTHORITY[\"EPSG\",\"7030\"]], AUTHORITY[\"EPSG
            \",\"6326\"]], PRIMEM[\"Greenwich\", 0.0, AUTHORITY[\"EPSG\",\"8901\"]], UNIT[\"degree of
            angle\",0.017453292519943295], AXIS[\"Geodetic latitude\",NORTH], AXIS[\"Geodetic
            longitude\",EAST], AUTHORITY[\"EPSG\",\"4326\"]]";
  }

}
```

Listing A.5: GPSToolSimple.java

```java
package de.soldin.gt2jump.test.gpstool;

import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.util.Hashtable;

import org.dinopolis.gpstool.gpsinput.GPSDataProcessor;
import org.dinopolis.gpstool.gpsinput.GPSDevice;
import org.dinopolis.gpstool.gpsinput.GPSException;
import org.dinopolis.gpstool.gpsinput.GPSSerialDevice;
import org.dinopolis.gpstool.gpsinput.nmea.GPSNmeaDataProcessor;

//-----------------------------------------------------------------------
/**
 * Simplified GPSTool - demonstrates the usage of the GPS Java API
 */

public class GPSToolSimple {
  protected GPSDataProcessor gps_data_processor;
  protected GPSDevice gps_device;
  Hashtable environment = new Hashtable();

  public GPSToolSimple(String[] arguments) {

    // Set default values
    String filename = null;
    String serial_port_name = null;
    int serial_port_speed = -1;

    // Define a file
//    environment.put(GPSFileDevice.PATH_NAME_KEY,ClassLoader.getSystemResource("MD_NMEA.txt").
      getPath());
//    gps_device = new GPSFileDevice();

    // Define serial port
    serial_port_name = "COM4";
    serial_port_speed = 4800;
    if (serial_port_name != null)
      environment.put(GPSSerialDevice.PORT_NAME_KEY, serial_port_name);
    if (serial_port_speed > -1)
      environment.put(GPSSerialDevice.PORT_SPEED_KEY, new Integer(serial_port_speed));
    gps_device = new GPSSerialDevice();

    // define ONE processor
    //gps_data_processor = new GPSGarminDataProcessor();
    //gps_data_processor = new GPSSirfDataProcessor();
    gps_data_processor = new GPSNmeaDataProcessor();

    try {
      // set params needed to open device (file,serial, ...):
      gps_device.init(environment);
      // connect device and data processor:
      gps_data_processor.setGPSDevice(gps_device);

      // start receiving
      gps_data_processor.open();

      // use progress listener to be informed about the number
      // of packages to download
      //gps_data_processor.addProgressListener(this);

      // Create a listener that prints out only
   PropertyChangeListener listener =
     new PropertyChangeListener(){
       public void propertyChange(PropertyChangeEvent event) {
         System.out.println(event.getPropertyName()+": "+event.getNewValue());
       }
     };

      gps_data_processor.addGPSDataChangeListener(listener);

      // don't quit now,, let's wait and receive some first
      Thread.sleep(200000);

      // Requests the gps device to send changes periodically. Device may ignore it.
      gps_data_processor.startSendPositionPeriodically(1000L);

      // close device and processor:
      gps_data_processor.close();
    } catch (GPSException e) {
      e.printStackTrace();
    } catch (InterruptedException e) {
      e.printStackTrace();
    }

  }

  public static void main(String[] arguments) {
    new GPSToolSimple(arguments);
  }
}
```

Listing A.6: GPL - Copying Permission Statement

```
For a one-file program, the statement (for the GPL) should look like this:
```

```
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

For programs that are more than one file, it is better to replace "this program" with the name of
       the program, and begin the statement with a line saying "This file is part of NAME". For
       instance,


    This file is part of Foobar.

    Foobar is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    Foobar is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with Foobar; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

Listing A.7: SwiXml Lizenz

```
LICENSE.txt,v 1.2 2003/11/01 carlsbadcubes

  Copyright (C) 2003 Carlsbad Cubes. All rights reserved.

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions
  are met:

  1. Redistributions of source code must retain the above copyright
     notice, this list of conditions, and the following disclaimer.

  2. Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions, and the disclaimer that follows
     these conditions in the documentation and/or other materials
     provided with the distribution.

  3. The name "SWIXML" must not be used to endorse or promote products
     derived from this software without prior written permission. For
     written permission, please contact info@swixml.org.

  4. Products derived from this software may not be called "SWIXML", nor
     may "SWIXML" appear in their name, without prior written permission
     from the SWIXML Project Management (info@swixml.org).

  In addition, we request (but do not require) that you include in the
  end-user documentation provided with the redistribution and/or in the
  software itself an acknowledgement equivalent to the following:

      "This product includes software developed by the
       SWIXML Project (http://www.swixml.org/)."

  Alternatively, the acknowledgment may be graphical using the logos
  available at http://www.swixml.org/img/logo/
  (See LOGO PERMISSION AGREEMENT below)


  THIS SOFTWARE IS PROVIDED ‘‘AS IS’’ AND ANY EXPRESSED OR IMPLIED
  WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
  OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  DISCLAIMED.  IN NO EVENT SHALL THE SWIXML AUTHORS OR THE PROJECT
  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
  LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
  USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
  ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
  OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
  SUCH DAMAGE.

  This software consists of voluntary contributions made by many
  individuals on behalf of the SWIXML Project and was originally
  created by Wolf Paulus <wpaulus@swixml.org>.

  For more information on the SWIXML Project, please see
  <http://www.swixml.org/>.
```

Anhang B

Korrespondenz

Korrespondenz 1: Anfrage an license@gnu.org

```
To: <licensing@gnu.org>
From: Edgar Soldin <edgar@soldin.de>
Subject: Copyright Holder, GPL and Proprietary License - Fwd: Re: Jump License - Re: [Geotools-gt2-
       users] New User of Geotools

Hello,

I've got some questions regarding GPL'ed software. There is this software called jump, developed and
        copyrighted by vividsolutions. Is it possible for them to stop developing it, take it from
       the web and only develop it further as proprietary product, because they own the complete
       copyright on
it?

Please read the dialog I had with a colleague too..

Thanks for your help
Ed
--
>> Accidently I came across this message today ... because I am currently
>> connecting parts of GT2 with JUMP, I had to workout their licenses as well.
>> Quite astonishing to me was the fact, that
>>
>> ">* JUMP is GPL, Geotools2 is LGPL. Which is the right license for you,
>>
>> >  I don't know, but if you want to develop a commercial application on top
>> >  of JUMP you'll have to enter into commercial agreements with
>> > VividSolutions"
>>
>> I couldn't understand how already free and available Software should be
>> unfree again.
>
>The copyright holder can provide the same software with different licensing.
>This means that VividSolutions is the sole copyright holder of JUMP, it
>can relicense to you the JUMP library under a different agreement than
>the GPL
>
>> So to say if I already have the full product or can get it
>> for free, why renegotiate with vividsolutions? I doublechecked
>> <http://www.gnu.org/licenses/gpl-faq.html> especially
>> <http://www.gnu.org/licenses/gpl-faq.html#GPLInProprietarySystem> and
>> <http://www.gnu.org/licenses/gpl-faq.html#DoesTheGPLRequireAvailabilityToPu>>blic> and <http://
       www.gnu.org/licenses/gpl-faq.html#GPLCommercially>
>> again .. and had a request in the <jump_discuss@yahoogroups.ca> list. John
>> from vividsolution didn't knew about a license except the GPL, which JUMP
>> is licensed with. And according to the faqs above ... a commercial
>> relicensing is simply made impossible by the GPL.
>
>You can't change the license of a product, the copyright holder can.
>
>> Even more the GPL urges
>> all extensions, like the ones I've written to be GPL as well. And in case
>> I'd decide to sell them the customer again has the right to sell, modify
>> etc. them. Strictly freedom going on.
>
>If you want to go with the GPL license and this is good for you, you don't
>have to do anything, just use JUMP as it is. Problem arises if you don't
>want to release the source of your work, the current GPL license of
>JUMP firbids this, so you would have to get a different agreement with
>the copyright holder, in that case, Vividsolutions.
>
>Best regards from a foggy Italy :-)
>Andrea
```

Korrespondenz 2: Antwort von license@gnu.org

```
Subject Re [gnu.org #156522] Copyright Holder, GPL and Proprietary License - Fwd Re Jump License -
       Re [Geotools-gt2-users] New User of Geotools
From "novalis@fsf.org via RT" <licensing@fsf.org>
Reply-To licensing@fsf.org
To edgar@soldin.de


On Wed, 2004-01-14 at 1450, Edgar Soldin via RT wrote

> Hello,
>
> I've got some questions regarding GPL'ed software. There is this
> software called jump, developed and copyrighted by vividsolutions.
> Is it possible for them to stop developing it, take it from the web
> and only develop it further as proprietary product, because they own
>  the complete copyright on it?

Yes, although anyone who has copies of it is free to redistribute them
under the terms of the GPL.

> Please read the dialog I had with a colleague too..

It sounds like Andrea has the right idea.

--
-Dave Turner
GPL Compliance Engineer
Support my work http//svcs.affero.net/rm.php?r=novalis&p=FSF
```

Korrespondenz 3: Zweite Anfrage an license@gnu.org

```
To: licensing@fsf.org
From: Edgar Soldin <edgar@soldin.de>
Subject: Re: [gnu.org #156522] Copyright Holder, GPL and Proprietary License - Fwd: Re: Jump License
        - Re: [Geotools-gt2-users] New User of Geotools

Thank you for your prompt answer,

but if it's like you suggest I've a following question. Lets say: Me is the company having a free
       software out there and people using the software a) advise me on bugs or b) send patches which
        I use or base my solutions on. These people still hold the copyright on this pieces. Right (
       for a and b)? So in theory, I've to get their permission to get proprietary again. On the
       other hand ... who is going to enforce that?

Is it in this case not generally a bad idea to support the free software as long ONE copyright
       holder persists on holding all the copyrights?

Thank you agin
Greetings from rainy germany Ede

>
>On Wed, 2004-01-14 at 14:50, Edgar Soldin via RT wrote:
>
>> Hello,
>>
>> I've got some questions regarding GPL'ed software. There is this
>> software called jump, developed and copyrighted by vividsolutions.
>> Is it possible for them to stop developing it, take it from the web
>> and only develop it further as proprietary product, because they own
>>  the complete copyright on it?
>
>Yes, although anyone who has copies of it is free to redistribute them
>under the terms of the GPL.
>
>> Please read the dialog I had with a colleague too..
>
>It sounds like Andrea has the right idea.
>
>--
>-Dave Turner
>GPL Compliance Engineer
>Support my work: http://svcs.affero.net/rm.php?r=novalis&p=FSF
```

Korrespondenz 4: Zweite Antwort von license@gnu.org

```
Subject Re [gnu.org #156522] Copyright Holder, GPL and Proprietary License - Fwd Re Jump License -
       Re [Geotools-gt2-users] New User of Geotools
From "novalis@fsf.org via RT" <licensing@fsf.org>
Reply-To licensing@fsf.org
To edgar@soldin.de


On Thu, 2004-01-15 at 0535, Edgar Soldin via RT wrote
```

```
> Thank you for your prompt answer,
>
> but if it’s like you suggest I’ve a following question. Lets say
> Me is the company having a free software out there and people using
> the software a) advise me on bugs or b) send patches which I use or
> base my solutions on. These people still hold the copyright on this
> pieces. Right (for a and b)?

For b, right. For a, someone who merely gives advice or information does
not get copyrights.

> So in theory, I’ve to get
> their permission to get proprietary again. On the other hand ...
> who is going to enforce that?

Contributors may enforce their rights in court.

> Is it in this case not generally a bad idea to support the free software
> as long ONE copyright holder persists on holding all the copyrights?

I can’t parse this.

I think that diffusion of copyrights has advantages and disadvantages.
On one hand, diffuse copyrights prevents anyone from making the software
proprietary.  Also, it doesn’t require copyright assignments, which are
extra work.  On the other hand, it makes the sometimes necessary task of
relicensing harder.

We recommend that you do not assign copyright to anyone on Free Software
work that you do unless the assignee promises to always release the
software as Free Software (even if they also offer proprietary
licenses).

--
-Dave Turner
GPL Compliance Engineer
Support my work http//svcs.affero.net/rm.php?r=novalis&p=FSF
```

Korrespondenz 5: Modal Taskmonitor jump_discuss@yahoogroups.ca

```
To <jump_discuss@yahoogroups.ca>
From Edgar Soldin <edgar.soldin@web.de>
Subject [jump_discuss] modal taskmonitor
Reply-To jump_discuss@yahoogroups.ca

Hi john,

attached you’ll find a java archive containing an ’example of plugins’ extension. The sources are
       included.
These should install themselfs under various menus and were done for test purposes ... using the
       threaded one i found that the taskmonitor dialog is per default modal (constructor). Wouldn’t
       it make sense to make it non-modal in case the computation of whatever the threaded plugin is
       doing AND the user want’s to go on with some other task? ..

this works com.vividsolutions.jump.workbench.ui.task.TaskMonitorDialog, Line 79
--
    public TaskMonitorDialog(Frame frame, ErrorHandler errorHandler) {
        this(frame, errorHandler, /*true*/false);
    }
--

If you don’t want to default it to non-modal, may be you could include an option to switch it to a
       given modal state?

Thanks alot
Ede


To unsubscribe from this group, send an email to
jump_discuss-unsubscribe@yahoogroups.ca



Your use of Yahoo! Groups is subject to http//ca.yahoo.com/docs/info/tos.html


 JumpPlugin2.jar

 JumpPlugin3.jar
```

Korrespondenz 6: Antwort von jump_discuss@yahoogroups.ca

```
To <jump_discuss@yahoogroups.ca>
From "Jonathan Aquino" <jaquino@vividsolutions.com>
Subject RE [jump_discuss] modal taskmonitor
Reply-To jump_discuss@yahoogroups.ca

Hi Ede -- When I see your efforts to make use of the JUMP framework, I'm delighted because I want to
        contribute to the work of others.

Regarding your question, the reason I made the TaskMonitorDialog modal was for safety reasons -- the
        data model (Layers, LayerManager, etc.) makes no provision for threading, and unexpected
       things may happen if two threads are modifying a FeatureCollection, for example. I'm not sure
       what would happen to the UndoHistory ... or if the active TaskFrame changes while a plugin is
       running ...

Then again, maybe some plugins are safe enough to allow the user to keep working -- plugins that don
       't do much. So maybe an option to switch to non-modal would be OK (for developers who know
       what they are doing).

I feel a bit distant from this problem at the moment as I am working on a couple of other projects
        -- how about if we keep it in mind for now and wait to see if other developers voice a need
       for this change?

--
Jon Aquino  Programmer/Analyst  250-385-6040  www.vividsolutions.com
```

Anhang C

Datenträger mit Software und Quellen

Quellenverzeichnis

Literatur

Behlendorf, Brian; Bradner, Scott; Hamerly, Jim; McKusick, Kirk; O'Reilly, Tim; Paquin, Tom; Perens, Bruce; Raymond, Eric; Stallman, Richard; Tiemann, Michael; Torvalds, Linus; Vixie, Paul; Wall, Larry; Young, Bob: Open Sources, Voices from the Open Source Revolution, hrsg. v. Chris Dibona, Mark Stone, Sam Ockman, O'Reilly & Associates, o.O. 1999.

Bürgerliches Gesetzbuch, Beck/dtv, München 2003.

Fornefeld, Martin; Oefinger, Peter: Marktstudie, Aktivierung des Geodatenmarktes in Nordrhein-Westfalen, Düsseldorf 2001, Download unter *www.micus.de*.

Fornefeld, Martin; Oefinger, Peter; Rausch, Ulrike: Der Markt für Geoinformationen, Potenziale für Beschäftigung, Innovation und Wertschöpfung, Düsseldorf 2001, Download unter *www.micus.de*.

Hang, Jiayin; Hohenson, Heidi: Eine Einführung in das Opensource Konzept aus Sicht der wirtschaftlichen und rechtlichen Aspekte, Report des C-Lab, Paderborn 2003.

Lange, Norbert de: Geoinformatik, in Theorie und Praxis, Berlin 2002.

Olbrich, Gerold; Quick, Michael; Schweikart, Jürgen: Desktop Mapping, Grundlagen und Praxis der Kartographie und GIS, Berlin 2002.

O'Reilly: Open Source, kurz und gut, O'Reilly, o.O. 1999, online *http://www.oreilly.de/german/freebooks/os_tb/toc.html*.

Urheber- und Verlagsrecht, Beck/dtv, München 2003.

Werth, Stefan: Open Source Beziehungen - Urheberschaft / Schenkung oder gesellschaftsrechtlicher Ansatz, Diplomarbeit an der Universität Paderborn, hrsg. v. Wolfgang Kern, Franz-Josef Rammig, Report des C-Lab, Paderborn 2003.

Zipf, Alexander; Strobl, Josef (Hrsg.): Geoinformation Mobil, Heidelberg 2002.

Internet

AED-SICAD: Webseite, *http://www.sicad.com/*, 2004.

Ashley, Steven (Scientific American): Next-Generation GPS, *http://www.sciam.com/article.cfm?chanID=sa004\&articleID=000B4F14-3F8A-1F45-B0B980A841890000* 2003, 19.1.2003.

Desbonnet, Joe: SA finally off!!, *http://www.wombat.ie/gps/*, 2004, 18.1.2004.

ESRI: Webseite, *http://www.esri.com/*, 2004.

European Petrol Survey Group: Webseite, *http://www.epsg.org/*, 2004.

European Space Agency: Webseite, *http://www.esa.int* 2004.

Europäische Union: Webseite, *http://europa.eu.int/* 2004.

Free Software Foundation: Webseite, *http://www.fsf.org/* 2004.

Galileo Projekt, Webseite, *http://europa.eu.int/comm/dgs/energy_transport/galileo* 2004.

Geoscience Online, Springer Verlag: Das Internet Magazin für Geo- und Naturwissenschaften, *http://www.g-o.de* 2004.

Gislounge: Webseite, *http://gislounge.com*, 2004, 4.1.2004.

GLONASS: Webseite, *http://www.glonass-center.ru* 2004.

GNU Project: Webseite, *http://www.gnu.org/* 2004.

Goodchild; Kemp: The NCGIA Core Curriculum in GIScience, *http://www.ncgia.ucsb.edu/education/curricula/giscc/giscc.html*, 2004.

Golem News: Immer mehr Autos mit Navigationssystemen, in: Golem.de, *http://www.golemnews.de/0307/26196.html*, 2003, 18.1.2004.

GRASS Projekt: Webseite, *http://www.geog.uni-hannover.de/grass/*, 2004.

IBM: Webseite, www.ibm.com, *http://www.ibm.com/* 2004.

Institut für Geodäsie und Geoinformatik (GG) AUF Universität Rostock: Geoinformatik-Service Webseite, *http://www.geoinformatik.uni-rostock.de/* 2003.

International Civil Aviation Organisation: Webseite, *http://www.icao.int* 2004.

Kaiser, Konrad: Dritte Generation der GPS-Datenerfassung für GIS mit Trimble Kinematik und RealTimeKinematik, Demonstration an praktischen Beispielen, *http://www.sbg.ac.at/geo/agit/papers94/kaiser.htm* o.J., 19.1.2004.

kowoma.de: Wie funktioniert GPS?, *http://www.kowoma.de/gps/*, 2004, 18.1.2004.

National Marine Electronics Association: Webseite, *http://www.nmea.org* 2004.

National Oceanic and Athmospheric Administration, U.S. Departement of Commerce: Webseite, *http://www.noaa.gov/* 2004.

NAVSTAR GPS Joint Office Program: Webseite, *http://gps.losangeles.af.mil/* 2004.

OpenGIS Consortium: Webseite, *http://www.opengis.org*, 2004.

Open Source Initiative: Webseite, *http://www.opensource.org/* 2004.

Page, Scott; Frost, Gerald; Lachow, Irving; Frelinger, David; Fossum, Donna; Wassem, Donald K.; Pinto Monica: The Global Positioning System, Assessing National Policies, *http://www.rand.org/publications/MR/MR614/*, 1995, 18.1.2004.

Satellitenpositionierungsdienst der deutschen Landesvermessung: Webseite, *http://www.sapos.de/* 2004.

SBAS Federal Aviation Administration: SBAS Technical Interoperability Working Group (IWG), *http://www2.faa.gov/asd/international/sbas.htm* 2002, 19.1.2004.

Schaeff, Alexander: Brisante Ergebnisse, Interview mit Winfried Kopperschmidt, *Kommune21*, 09/2003.

Schwarz, Hannes: Seminar Mobile Systeme SS03, *http://www.uni-koblenz.de/~agrt/lehre/ss2003/seminar/hannes_schwarz_folien.ppt*

Stahl, R.; Greve, K. (Hrsg.): GIS und Internet Tutorial (Version 3.0), o.O. 1998, *http://www.gis-tutor.de*, 2004.

Sun Microsystems: Webseite, *http://java.sun.com* 2004.

Telepolis: Magazin der Netzkultur, *http://www.telepolis.de/* 2004.

The Online Investor: Webseite, *http://www.theonlineinvestor.com*, 2004.

Trimble: Webseite, *http://www.trimble.com* 2004.

U.S. Coast Guard Navigation Center: GPS, *http://www.navcen.uscg.gov/gps/geninfo/*, 2004, 18.1.2004.

U.S. Naval Observatory: USNO NAVSTAR Global Positioning System, *http://tycho.usno.navy.mil/gpsinfo.html*, 2004, 18.1.2004.

Vivid Solutions: JUMP Webseite, *http://www.vividsolutions.com/jump* 2003.

Walter, Thomas B.: Geographic Information Science II, *http://everest.hunter.cuny.edu/~tbw/gtech362/lecture.notes/lectures.html*, 1998.

Wikipedia: freie Enzyklopädie, *http://de.wikipedia.org/*, 2004.

World Standard Services Network: Webseite, *http://www.wssn.net/*, 09/2003.

www.nmea.de: Informationen zum NMEA Protokoll, *http://www.nmea.de/* 2004.

Ehrenwörtliche Erklärung

Hiermit erkläre ich, die vorliegende Arbeit selbstständig unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt zu haben.

Edgar Soldin
Magdeburg, 1. Februar 2004