2015 GI FOSS

# Anwenderkonferenz für Freie und Open Source Software für Geoinformationssysteme

Münster, 11. - 13. März 2015







Goldsponsor:


Silbersponsoren:







Bronzesponsoren:











Mediasponsoren:


Anwenderkonferenz für Freie
und Open Source Software
für Geoinformationssysteme

ISBN 978-3-00-048886-3

Herausgeber: FOSSGIS e.V.
http://www.fossgis.de

# Inhaltsverzeichnis

# Herausforderungen bei der Umsetzung der INSPIRE-Richtlinie

Jürgen Weichand

In der Technical Guidance zu den INSPIRE-Downloaddiensten [1] werden „konforme“ und „interoperable“ Downloaddienste unterschieden.

**Konforme Downloaddienste** erfüllen die Vorgaben der Durchführungsbestimmung „Netzdienste“ [2]. Weiterhin stellen konforme, aber nicht interoperable Downloaddienste INSPIRE-relevante Geodaten in einem vom Datenanbieter definierten Datenmodell bereit. Meist werden hierfür einfache Feature-Modelle eingesetzt.

**Interoperable Downloaddienste** erfüllen zusätzlich die Vorgaben der Durchführungsbestimmung „Interoperabilität“ [3]. Die Geodaten werden gemäß INSPIRE-Datenspezifikationen bereitgestellt. Hierbei handelt es sich um komplexe Feature-Modelle in Form von GML-Anwendungsschemata.

Dieser Beitrag widmet sich den speziellen Anforderungen, die bei der Bereitstellung von komplexen Feature-Modellen über einen **Web Feature Service** (WFS) zu berücksichtigen sind [4].

## 1. Unterscheidung einfache und komplexe Feature-Modelle

Der Begriff einfaches Feature-Modell basiert auf den Vorgaben der Simple Feature Specification (SFS) [5] und des Simple Feature GML Profils [6]. In Tabelle 1 werden die Eigenschaften von einfachen und komplexen Feature-Modellen gegenüber gestellt:

Tab 1.: Gegenüberstellung Feature-Modelle [4], [6], [7]

| **Einfacher FeatureType**[1] | **Komplexer FeatureType** |
|---|---|
| Attribute basieren auf den primitiven Datentypen: String, Integer, Measurement, Date, Real, Binary, Boolean und URI | Attribute können aus vollständigen Objekten bestehen (z. B. Person, Eigentümer, Flurstück).<br>Attribute können auf Objekte referenzieren.<br>Attribute können auf spezielle GML-Datentypen zurückgreifen (z. B. Maßeinheiten).<br>Attribute können generisches XML enthalten. |
| Die Kardinalität der Attribute ist 0 oder 1. | Die Kardinalität der Attribute ist 0..*. |
| Geometrien basieren auf den Vorgaben der Simple Feature Specification.<br>Einschränkung auf die Geometrietypen: Punkt, Linie, Polygon, Multipunkt, Multilinie, Multipolygon sowie heterogene Multigeometrien<br>Die Interpolation zwischen den Stützpunkten erfolgt linear. | Geometrien basieren auf den Vorgaben der ISO 19107.<br>Zusätzlich können u. a. 3D-Körper sowie nicht-planare und nicht-linear interpolierte Geometrien beschrieben werden. |

1 Nach Simple Feature Compliance Level SF-0

Das folgende Beispiel für das INSPIRE-Datenmodell **Adressen** veranschaulicht die zuvor aufgeführten Eigenschaften von komplexen Datenmodellen und ist den Beispielen der Open-Source Software **Stetl** entnommen [8].

Beispiel für komplexes Feature-Modell – FeatureType **Address** [8]

```
<base:member>
    <AD:Address
        xmlns:AD="urn:x-inspire:specification:gmlas:Addresses:3.0"
        xmlns:gml="http://www.opengis.net/gml/3.2"
        xmlns:base="urn:x-inspire:specification:gmlas:BaseTypes:3.2"
        xmlns:GN="urn:x-inspire:specification:gmlas:GeographicalNames:3.0"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            gml:id="NL.KAD.BAG.AD.Address.362200000022695">

        <gml:identifier codeSpace="http://inspire.jrc.ec.europa.eu/">
            urn:x-inspire:object:id:NL.KAD.BAG.AD.Address.362200000022695
        </gml:identifier>
        <AD:inspireId>
            <base:Identifier>
                <base:localId>362200000022695</base:localId>
                <base:namespace>NL.KAD.BAG.AD.Address</base:namespace>
            </base:Identifier>
        </AD:inspireId>
        <AD:position>
            <AD:GeographicPosition>
                <AD:geometry>
                    <gml:Point gml:id="NL.KAD.BAG.AD.Address.362200000022695_P"
                            srsName="urn:ogc:def:crs:EPSG::4258">
                        <gml:pos>52.299493602339524 4.867708706943735</gml:pos>
                    </gml:Point>
                </AD:geometry>
                <AD:specification>entrance</AD:specification>
                <AD:method>byOtherParty</AD:method>
                <AD:default>true</AD:default>
            </AD:GeographicPosition>
        </AD:position>
        <AD:locator>
            <AD:AddressLocator>
                <AD:designator>
                    <AD:LocatorDesignator>
                        <AD:designator>1025</AD:designator>
                        <AD:type>2</AD:type>
                    </AD:LocatorDesignator>
                </AD:designator>
                <AD:level>unitLevel</AD:level>
            </AD:AddressLocator>
        </AD:locator>
        <AD:validFrom>2010-08-17T00:00:00</AD:validFrom>
        <AD:validTo xsi:nil="true" nilReason="other:unpopulated"/>
        <AD:beginLifespanVersion xsi:nil="true" nilReason="other:unpopulated"/>
        <AD:endLifespanVersion xsi:nil="true" nilReason="other:unpopulated"/>
        <AD:component xlink:href="#NL.KAD.BAG.AD.ThoroughfareName.362300000030694"/>
        <AD:component xlink:href="#NL.KAD.BAG.AD.AddressAreaName.1050"/>
        <AD:component xlink:href="#NL.KAD.BAG.AD.PostalDescriptor.1181WP"/>
    </AD:Address>
</base:member>
```

Der FeatureType **Address** besitzt in diesem Beispiel die drei Komponenten **ThoroughfareName** (Name des Verkehrsweges), **AddressAreaName** (Name des Adressbereichs) und **PostalDescriptor** (Postalischer Deskriptor), die als Referenz (xlink:href) realisiert sind [9]. Durch den Einsatz von Referenzen kann auf die redundante Beschreibung von

Instanzen der FeatureTypes ThoroughfareName, AddressAreaName und PostalDescriptor verzichtet werden. Nachfolgend wird die aufgelöste PostalDescriptor-Referenz dargestellt.

Beispiel für komplexes Feature-Modell – Aufgelöste **PostalDescriptor-Referenz** [8]

```
<base:member>
    <AD:PostalDescriptor
        xmlns:AD="urn:x-inspire:specification:gmlas:Addresses:3.0"
        xmlns:gml="http://www.opengis.net/gml/3.2"
        xmlns:base="urn:x-inspire:specification:gmlas:BaseTypes:3.2"
        xmlns:GN="urn:x-inspire:specification:gmlas:GeographicalNames:3.0"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                gml:id="NL.KAD.BAG.AD.PostalDescriptor.1181WP">
        <gml:identifier codeSpace="http://inspire.jrc.ec.europa.eu/">
            urn:x-inspire:object:id:NL.KAD.BAG.AD.PostalDescriptor.1181WP
        </gml:identifier>
        <AD:inspireId>
            <base:Identifier>
                <base:localId>1181WP</base:localId>
                <base:namespace>NL.KAD.BAG.AD.PostalDescriptor</base:namespace>
            </base:Identifier>
        </AD:inspireId>
        <AD:beginLifespanVersion xsi:nil="true" nilReason="other:unpopulated"/>
        <AD:endLifespanVersion xsi:nil="true" nilReason="other:unpopulated"/>
        <AD:validFrom xsi:nil="true" nilReason="other:unpopulated"/>
        <AD:validTo xsi:nil="true" nilReason="other:unpopulated"/>
        <AD:postCode>1181WP</AD:postCode>
    </AD:PostalDescriptor>
</base:member>
```

## 2. Bereitstellung von komplexen Feature-Modellen über WFS

Die Speicherung von komplexen Objekten in relationalen Datenbanken kann aufgrund vorhandener 1:n Beziehungen innerhalb der Objektstruktur zu einer großen Anzahl von Tabellen führen. Dieses und weitere Probleme, die sich aus der Speicherung von objektorientierten Strukturen in relationalen Datenbanken ergeben, werden unter dem Begriff „Impedance Mismatch“ („objektrelationale Unverträglichkeit“) zusammengefasst [7]. Über eine „objektrelationale Abbildung“ (ORA) werden die FeatureTypes des GML-Anwendungsschemas auf die Relationen (Tabellen) der Datenbank abgebildet. Für die Rekonstruktion eines Features, beispielsweise bei der Ausgabe als GML über WFS, müssen Daten aus mehreren Datenbanktabellen verbunden (engl. „Join“) werden. Hierzu ist eine komplexe und u. U. performancekritische SQL-Abfrage notwendig [4].

### 2.1 Objektrelationale Abbildung auf existierende Datenbankstruktur

Bei der Bereitstellung von INSPIRE-Datenmodellen auf Grundlage von existierenden Geodatenbanken kann die Datenmodelltransformation (DMT) und die objektrelationale Abbildung in einem Schritt zusammengefasst werden („on-the-fly Transformation“ – vgl. Abb. 1). Dieses Verfahren bietet sich an, wenn bei der Datenmodelltransformation keine **komplexen Transformationsfunktionen** [10] notwendig sind. Folgende performancekritische Transformationsfunktionen können beispielsweise bei einer Datenmodelltransformation notwendig werden:

- **Verschmelzung**
  Beispiel: Mehrere Quellobjekte werden geometrisch zu einem Zielobjekt zusammengefasst.

- **Teilung**
  Beispiel: Ein Quellobjekt mit komplexem Geometrietyp wird in mehrere Zielobjekte mit primitivem Geometrietyp aufgeteilt.

- **Wertumwandlung**
  Beispiel: Der Attributwert „Fläche“ wird aus einer vorhandenen Geometrie berechnet.

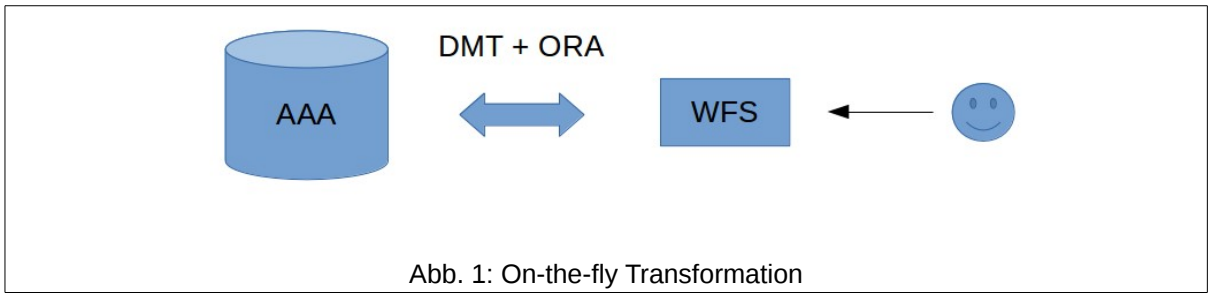



Abb. 1: On-the-fly Transformation

Vorteile:

- Auf die Bereitstellung einer Sekundärdatenhaltung kann verzichtet werden.
- Es wird nur eine einzige Konfiguration für die Datenmodelltransformation und objektrelationale Abbildung benötigt.

Nachteile:

- Eine on-the-fly Transformation ist insbesondere bei einer großen Anzahl von Relationen und komplexen Transformationsfunktionen (s. o.) nicht ausreichend performant.

### 2.2 Objektrelationale Abbildung auf Sekundärdatenhaltung

Im Falle einer komplexeren Datenmodelltransformation besteht die Notwendigkeit zum Aufbau einer Sekundärdatenhaltung. Diese hält die mittels Transformationsfunktionen umgeformten Geodaten vor. Für die Datenmodelltransformation sind hierbei zwei Vorgehensweisen möglich:

#### a) Zielschema entspricht „INSPIRE-like“ Datenbankschema

Bei dieser Vorgehensweise entspricht das Zielschema einem an das INSPIRE-Datenmodell angelehnten Datenbankschema.

Beispiel-Workflow (vgl. Abb. 2):

1. Automatisiertes Anlegen der Datenbankstruktur und der korrespondierenden objektrelationalen Abbildung aus dem GML-Anwendungsschema (z. B. deegree 3)
2. Datenmodelltransformation zwischen Quellschema und dem generierten Datenbankschema (z. B: HALE)
3. Transformation der Geodaten (Ergebnis: Gefüllte Datenbank) (z. B. HALE)

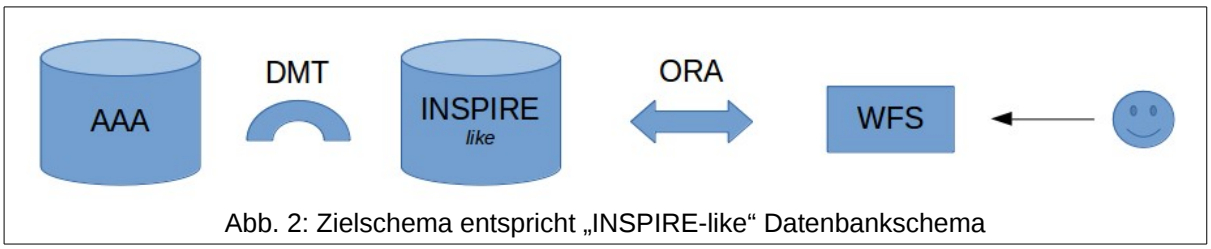



Abb. 2: Zielschema entspricht „INSPIRE-like“ Datenbankschema

Vorteile:

- Durch eine auf das Zielmodell abgestimmte Sekundärdatenhaltung kann bei der Ausgabe über WFS auf performancelastige Transformationsfunktionen verzichtet werden.
- Die Sekundärdatenhaltung in Form einer standardisierten Geodatenbank kann durch weitere Softwareprodukte nachgenutzt werden.

Nachteile:

- Die Sekundärdatenhaltung muss kontinuierlich aktualisiert werden.
- Während der Datenmodelltransformation müssen die Beziehungen zwischen den Datenbanktabellen (referenzielle Integrität) selbständig realisiert werden.

### b) Zielschema entspricht GML-Applikationsschema

Bei dieser Vorgehensweise entspricht das Zielschema dem GML-Anwendungsschema. Die als Ergebnis vorliegende GML-FeatureCollection (statische GML-Datei) wird anschließend in einem zusätzlichen Schritt über eine geeinigte Schnittstelle importiert.

Beispiel-Workflow (vgl. Abb. 3):

1. Automatisiertes Anlegen der Datenbankstruktur und der korrespondierenden objekrelationalen Abbildung (z. B. deegree 3)
2. Datenmodelltransformation zwischen Quellschema und dem GML-Applikationsschema (z. B: HALE)
3. Transformation der Geodaten (Ergebnis: GML-FeatureCollection) (z. B. HALE)
4. Import der GML-Datei über WFS-T (Ergebnis: Gefüllte Datenbank) (z. B. deegree 3)

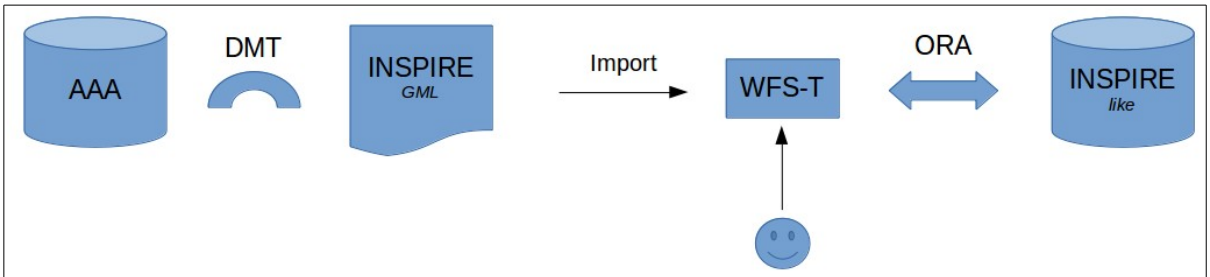



Abb. 3: Zielschema entspricht GML-Anwendungsschema – Import in OR-Datenbank

Vorteile:

- Durch eine auf das Zielmodell abgestimmte Sekundärdatenhaltung kann bei der Ausgabe über WFS auf performancelastige Transformationsfunktionen verzichtet werden.
- Die Sekundärdatenhaltung in Form einer standardisierten Geodatenbank kann durch weitere Softwareprodukte nachgenutzt werden.
- Die Beziehungen zwischen den Datenbanktabellen werden automatisiert beim Import hergestellt und die referenzielle Integrität somit sichergestellt.

Nachteile:

- Die Sekundärdatenhaltung muss kontinuierlich aktualisiert werden.

### 2.3 Sekundärdatenhaltung – BLOB-Store

Als Alternative zur relationalen Datenhaltung bietet deegree 3 einen **BLOB-Modus** an, der vollständige GML-Objekte in einer Tabellenzelle binär ablegt. Des Weiteren werden die Objekteigenschaften Identifikator (ID) und das umgebende Begrenzungsrechteck (BoundingBox) ermittelt und in gesonderte Tabellenzellen gespeichert. Über diese Attribute ist eine einfache Selektion der GML-Objekte möglich. Anfragen, deren Filterbedingungen auf anderen Objekteigenschaften beruhen, müssen durch die Serverkomponente nachgefiltert werden [4] [7]. Der Workflow für die Bereitstellung entspricht dem Vorgehen nach 2.2 b) (vgl. Abb. 4) .

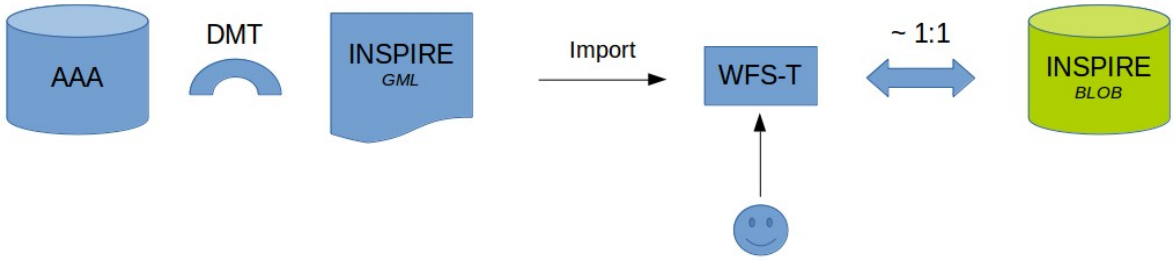



Abb. 4: Zielschema entspricht GML-Anwendungsschema – Import in BLOB-Store

Vorteile:

- Der BLOB-Modus liefert die beste Performance bei der Ausgabe von GML-Objekten, da keine Rekonstruktion der Objekte aus einer objektrelationalen Datenbank notwendig ist.

Nachteile:

- Die Sekundärdatenhaltung muss kontinuierlich aktualisiert werden.
- Die proprietäre Sekundärdatenhaltung in Form von BLOBs kann nicht durch standardisierte Softwareprodukte nachgenutzt werden.
- Die Attributfilterung über WFS ist nur eingeschränkt möglich oder wenig performant.

## 3. Fazit

Gelegentlich wird die Komplexität bei der Bereitstellung von Geodaten gemäß INSPIRE-Datenspezifikationen auf das Thema Datenmodelltransformation reduziert. Eine weitere große Herausforderung ergibt sich jedoch bei der anschließenden Bereitstellung der transformierten Geodaten über Darstellungs- und Downloaddienste.

Bereits bei der Datenmodelltransformation sollte die vorgesehene Bereitstellungsvariante berücksichtigt werden. In einfachen Fällen kann auf eine zusätzliche Sekundärdatenhaltung verzichtet werden (vgl. Kapitel 2.1).

Ist aufgrund einer zu komplexen Datenmodelltransformation keine on-the-fly Transformation möglich, sollte darauf geachtet werden, dass die benötigte Sekundärdatenhaltung möglichst umfangreich nachgenutzt werden kann (vgl. Kapitel 2.2). Weiterhin ist die stetige Aktualisierung der Sekundärdatenhaltung durch automatisierte Prozesse sicherzustellen.

Ist die Performance des WFS aufgrund einer zu komplexen objektrelationalen Abbildung weiterhin nicht ausreichend, können Speziallösungen wie der deegree BLOB-Modus in Erwägung gezogen werden (vgl. Kapitel 2.3).

Um bei der Bereitstellung von komplexen Feature-Modellen über WFS möglichst flexibel zu sein, sollten bei der Wahl eines geeigneten WFS-Servers die in Tabelle 2 aufgeführten Kriterien berücksichtigt werden.

Tab 2.: Kriterien für die Auswahl von Softwareprodukten

| | |
|---|---|
| Objektrelationale Abbildung | Kann ein GML-Anwendungsschema auf eine objektrelationale Datenbank abgebildet werden? |
| Automatische Erstellung der Datenbankstruktur und der korrespondierenden objektrelationalen Abbildung | Kann aus einem GML-Anwendungsschema automatisiert eine geeignete Datenbankstruktur erzeugt werden?<br>Wird die objektrelationale Abbildung zwischen Datenbank und GML-Anwendungsschema hierbei ebenfalls automatisiert erzeugt? |
| Import von GML | Können in ein GML-Anwendungsschema transformierte Geodaten über eine geeignete Schnittstelle (z. B. WFS-T) eingelesen werden? |

**Kontakt zum Autor**:

Jürgen Weichand
Landesamt für Digitalisierung, Breitband und Vermessung
Alexandrastraße 4
80538 München
juergen.weichand@ldbv.bayern.de

**Literatur:**

[1] *Initial Operating Capability Task Force*: **Technical Guidance for the implementation of INSPIRE Download Services**, Version 3.1, 2013, http://inspire.jrc.ec.europa.eu/documents/Network_Services/Technical_Guidance_Download_Services_v3.1.pdf, letzter Zugriff: 30.01.2015

[2] *Europäische Kommission*: **Verordnung (EG) Nr. 976/2009 der Kommission vom 19. Oktober 2009 zur Durchführung der Richtlinie 2007/2/EG des Europäischen Parlaments und des Rates hinsichtlich der Netzdienste**, Konsolidierte Fassung, 2009/2010, http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:2009R0976:20101228:DE:HTML, letzter Zugriff: 30.01.2015

[3] *Europäische Kommission*: **Verordnung (EG) Nr. 1089/2010 der Kommission vom 23. November 2010 zur Durchführung der Richtlinie 2007/2/EG des Europäischen Parlaments und des Rates hinsichtlich der Interoperabilität von Geodatensätzen und -diensten**, Konsolidierte Fassung, 2010/2011, http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:2010R1089:20110225:DE:HTML, letzter Zugriff: 30.01.2015

[4] *Weichand, J.*: **Entwicklung und Anwendung von Downloaddiensten im Kontext der europäischen Geodateninfrastruktur INSPIRE**, Masterarbeit, Hochschule Anhalt, Dessau, 2013, http://www.weichand.de/masterarbeit-inspire-downloaddienste/, letzter Zugriff: 30.01.2015

[5] *Open Geospatial Consortium Inc. (OGC)*: **OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture**, OGC 06-103r4, Version 1.2.1, 2011, http://portal.opengeospatial.org/files/?artifact_id=25355, letzter Zugriff: 30.01.2015

[6] *Open Geospatial Consortium Inc. (OGC)*: **Geography Markup Language (GML) simple features profile**, OGC 10-100r3, Version 2.0, 2012, http://portal.opengeospatial.org/files/?artifact_id=42729, letzter Zugriff: 30.01.2015

[7] *Schneider, M.; Fitzke, J.*: **Professionelle Implementierung der INSPIRE Data Themes mit Open Source-Technologien**, in: Schilcher M. (Hrsg.), Geoinformationssysteme - Beiträge zum 16. Münchner Fortbildungsseminar, München, 2012, Seite 134-144

[8] *van den Broecke, J.*: **Stetl**, http://stetl.org, letzter Zugriff: 30.01.2015

[9] *Koordinierungsstelle GDI-DE*: **Steckbrief zur INSPIRE-Datenspezifikation Adressen**, Version 1.0, 2010, http://www.geoportal.de/SharedDocs/Downloads/DE/GDI-DE/Steckbrief_Adressen.pdf?__blob=publicationFile, letzter Zugriff: 30.01.2015

[10] *Fichtinger, A.*: **Semantische Transformation im Kontext von INSPIRE - dargestellt am Beispiel der grenzüberschreitenden Bodenseeregion**, Dissertation, Technische Universität München, München, 2011, https://mediatum.ub.tum.de/?id=1079893, letzter Zugriff: 30.01.2015

# Welches Münster meinen Sie?

## Wie man Geocodern beibringt Orte mit gleichen Namen richtig zu sortieren

Marc Tobias Metten

Wie sortiert man Orte nach Wichtigkeit, wenn sie den gleichen Namen haben? Wir schauen uns Daten zu Suchverhalten, Größe, Bevölkerungsdichte, Tagging in OpenStreetMap und Verlinkung in Wikipedia/Wikidata anhand des Nominatim Geocoders an.

Der Nominatim Geocoder nutzt OpenStreetMap, minütlich aktualisiert. Die Suchergebnisse werden nach Relevanz sortiert. Einige Faktoren, wie dass Ländernamen wichtiger sind als Strassennamen sind jedem klar. Auf Mailinglisten kommt aber hin und wieder die Frage auf warum genau ein Ort wichtiger eingeschätzt wird als ein anderer.

Frankfurt gibt es zweimal, beide sind grosse Städte. Münster gibt es mehrfach (http://de.wikipedia.org/wiki/M%C3%BCnster). Sogar Paris, Berlin und Frankreich sobald man die ganze Welt betrachtet.

Nominatim nutzt u.a. einen vorberechneten Score (numerischer Wert), der auf den Verlinkungen innerhalb Wikipedia basiert. Die erste Version sogar auf Seitenabrufen. Das hat Vor- und Nachteile (und Bugs). Leider sind selbst für Nutzer (Administratoren) von Nominatim die Algorithmen dahinter nicht transparent genug. Viele laden einfach eine selten aktualisierte Binärdatei von http://www.nominatim.org/.

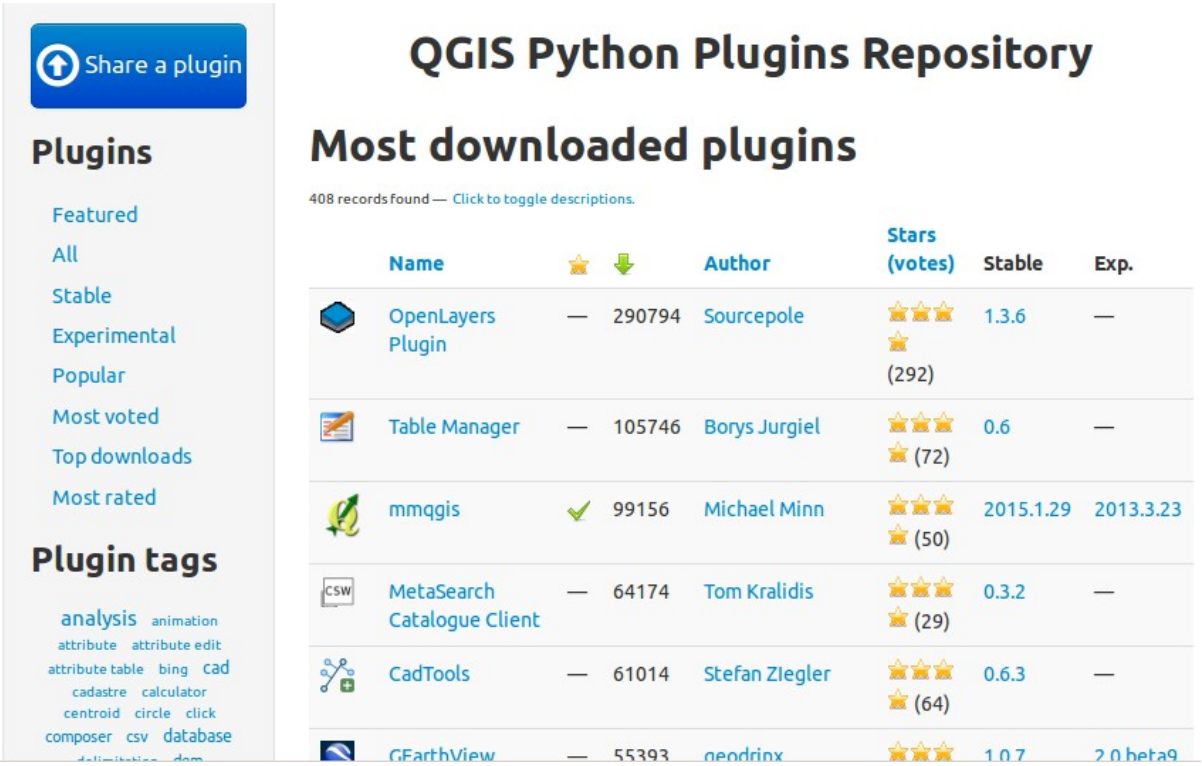
Ich habe mich damit intensiver beschäftigt und plane bis zur FOSSGIS die Scripte neu schreiben und neu dokumentieren. Selbst wenn nicht wird der Vortrag ein guter Überlick werden. Ich arbeite bei http://data.opencagedata.com/index.html#about-section und wir bieten einen Geocoder u.a. auf Basis von Nominatim an http://geocoder.opencagedata.com/. Ich arbeite seit 2006 Jahren mit geocodern mit underschiedlichen kommerziellen und offenen Daten.

# QGIS Plugins - Must-Haves, Fachlösungen und Geheimtipps

Pirmin Kalberer

Eine grosse Stärke von QGIS ist die einfache, aber umfassende Erweiterbarkeit mittels Python Plugins.

In kompakter Form wird eine Selektion aus der grossen Menge an öffentlich verfügbaren Plugins aus verschiedensten Bereichen vorgestellt [1]. Die Auswahl geht vom bestens bekannten Open Layers Plugin und weiteren "Must-Haves" über wenig bekannte Core-Plugins wie dem Offline Editing Plugin bis zu Insidertipps wie dem Remote Debugging Plugin. Neben nützlichen Helfern werden auch umfangreiche Fachlösungen kurz vorgestellt. Es werden Anwendungsbereiche von der Datenerfassung bis zur Plugin-Entwicklung abgedeckt, aber auch Tipps gegeben, wie man selbst ein passendes Plugin findet und evaluiert.




Kontakt zum Autor:

Pirmin Kalberer
Sourcepole AG
Weberstrasse 5
CH-8004 Zürich
www.sourcepole.ch
pka@sourcepole.ch
@PirminKalberer

Links:

[1] http://sourcepole.ch/qgis-plugins- fossgis2015

# GeoPortal.rlp

## Ein Erfolgsmodell für den Einsatz von FOSS in der öffentlichen Verwaltung

Armin Retterath

Dieser Vortrag illustriert anhand der Entstehungsgeschichte des rheinland-pfälzischen Geoportals, das mittlerweile auch im Saarland und in Hessen eingesetzt wird, welches immense Potential im Einsatz von FOSS-Software in der öffentlichen Verwaltung steckt. Der Vortrag diskutiert neben unerwarteten positiven Nebenwirkungen wie der verstärkten Zusammenarbeit der Verwaltungen auch Risiken und Probleme, die in der 9-jährigen Geschichte des Projekts zu bewältigen waren.

Das vollständig auf FOSS basierende GeoPortal.rlp wurde im Jahr 2006 konzipiert, um die GDI-RP aufzubauen. Rheinland-Pfalz hat sich damals für den Einsatz lizenzkostenfreier Software entschieden, um auch anderen Ländern bzw. Organisationen die Möglichkeit zu geben an dem Projekt mitzuwirken. Zum Zeitpunkt der Freischaltung des Portals im Jahr 2007 war die rheinland-pfälzische Lösung europaweit das einzige System seiner Art. Es basiert auf einem eigenständig entwickeltem Architekturkonzept (OWS-Service-Registry) und setzte zum damaligen Zeitpunkt auch weltweit Maßstäbe. Im Jahr 2009 wurde das System vom Saarland übernommen und es kam zu einer engen Kooperation der beiden Bundesländer, in deren Rahmen man sich die Weiterentwicklungskosten aufteilen konnte. Zur Umsetzung der EU-INSPIRE-Richtlinie wurden in der zugrundeliegenden Software Mapbender viele Funktionen implementiert, die den dezentralen Datenanbietern in Rheinland-Pfalz und im Saarland die Erfüllung der europäischen Anforderungen stark erleichtern. In Folge dessen haben diese beiden Bundesländer bezüglich der Datenbereitstellung derzeit europaweit eine Spitzenposition inne. Der Erfolg des Systems hat dazu geführt, dass auch Hessen sich im Jahr 2013 dazu entschieden hat sein aktuell noch auf proprietärer Software basierendes Geoportal durch die FOSS-Lösung aus Rheinland-Pfalz zu ersetzen. Im Januar 2015 wird der Prototyp des neuen Geoportals Hessen freigeschaltet. Anhand der Entwicklung Projektes in den letzten 9 Jahren kann gezeigt werden welches immense Potential im Einsatz von FOSS steckt. Es wird neben Darstellung der Vorteile aber auch auf die Risiken und die Probleme eingegangen, die während dieser Zeit auftauchten und zu bewältigen waren.

# Offene Geodaten: Lage von Orten im Vergleich

Thomas Brinkhoff

## Einführung

Neben amtlichen und kommerziellen Geodaten stellen seit einigen Jahren auch insbesondere durch *Crowdsourcing* erhobene nutzergenerierte Inhalte (user-generated content, UGC) eine wichtige Grundlage für raumbezogene Anwendungen dar. So ist OpenStreetMap (OSM) sicherlich derzeit das bekannteste Projekt für gemeinschaftliche Kartierungen (collaborative mapping). Sowohl durch solche Aktivitäten als auch durch die zunehmende Bereitstellung von amtlichen Daten unter Nutzung von freien Lizenzformen nimmt der Umfang von *offenen Geodaten* (open geospatial data) stetig zu [1]. Mit dieser Zunahme steigt aber auch die Heterogenität der Datenquellen und damit auch der Datenqualität. So kann amtlichen Daten (meist) eine hohe Lagequalität unterstellt werden, während diese bei anderen Herkunftsquellen stark variieren kann. Andererseits bietet die zunehmende Zahl von Datenquellen auch die Chance, die Daten untereinander zu vergleichen und Daten(quellen) hoher Qualität zu bestimmen und auszuzeichnen.

Eine andere wichtige Entwicklung, die man im Geoinformationsbereich beobachten kann, ist die zunehmende Verfügbarkeit einerseits funktional weit entwickelter und andererseits auch einfach zu nutzender Software unter freien Lizenzmodellen. Dies umfasst (u.a.) Software-Pakete und -Bibliotheken für Geodatenbanksysteme, Geodienste, Web Mapping, Geoinformationssysteme (GIS) und zur Programmierung raumbezogener Anwendungen [2]. Diese Entwicklung schlägt sich auch in den Lehrinhalten von Hochschulstudiengängen nieder. So wird im Bachelor-Studiengang „Geoinformatik“ an der Jade Hochschule in Oldenburg in den Modulen „Geoinformationssysteme I und II“ QGIS als eines von zwei Hauptsystemen gelehrt. Die Einführung in Geodatenbanken erfolgt mit PostgreSQL/PostGIS und die in Geodienste mit GeoServer. Die nachfolgenden Ausführungen beruhen u.a. auf einer Abschlussaufgabe für das Modul „GIS-Programmierung“. In diesem Modul wird in Entwicklung raumbezogener Anwendungen mittels Software-Bibliotheken hauptsächlich unter Nutzung von GeoTools und JTS (Java Topology Suite) eingeführt.

Aus den bisherigen Ausführungen leitet sich die Aufgabe ab, die nachfolgend betrachtet werden soll: eine Qualitätsuntersuchung von offenen Geodaten. Die zu betrachtenden Daten sind auf einen wichtigen Spezialfall beschränkt: die Lage von Orten (wie Städte, Gemeinden und Siedlungen) inklusive administrativer Gebiete (Bezirke, Gemeinden usw.). Solche Daten sind eine wichtige Grundlage für die Erstellung von Karten und für diverse Geoanwendungen. Die Untersuchung wird durch ein mit GeoTools/JTS entwickeltes Werkzeug unterstützt.

### Offene Daten zur Lage von Orten

Die Lage von Orten kann entweder über eine Punktkoordinate oder über Flächen (i.d.R. als Multipolygon) vorliegen. Wichtige Quellen für Lagedaten sind u.a. die GeoNames Geographical Database, OpenStreetMap und Wikipedia; sie weisen folgende Vorteile auf:

- weltweite Abdeckung
- zentrales Portal (OSM z.B. über http://download.geofabrik.de/)
- einheitliches Datenmodell
- (relativ) einfache Zugriffsmöglichkeiten auf Teile der Daten

**GeoNames** unterliegt der „Creative Commons Attribution 3.0 License“. Auf die Daten kann über Download und Dienste zugegriffen werden. Wichtige Attribute sind:

- geonameid: globale ID
- name: Name des Ortes
- alternatenames: Liste mit alternativen Namen und Schreibweisen

- latitude: geografische Breite in WGS‘84
- longitude: geografische Länge in WGS’84
- feature class: Grundtyp; „P“ für Orte (places), „A“ für administrative Einheiten
- feature code: Untertyp; z.B. „PPL“ für bewohnte Orte (populated place)
- country code: Staat mit ISO-3166 2-Code
- admin1 code, …, admin4 code: Administrative Ebene 1/2/3/4
- population: Einwohnerzahl (meist nicht belegt; ohne Referenzdatum)
- modification date: Datum der letzten Änderung

Als Defizite können genannt werden, dass keine Qualitätsindikatoren für die Lageinformation und keine nationalen Schlüssel in den Daten enthalten sind.

**OSM-Daten** werden (seit 2012) unter der „Open Database License“ (ODbL) veröffentlicht. Die Daten stehen (oftmals staatenweise) zum Download zur Verfügung. Administrative Einheiten liegen i.d.R. als flächenhafte Objekte vor, die ein „tag“-Element besitzen, dessen Schlüsselattribut „k“ den Wert „admin_level“ aufweist (nachfolgend werden solche Elemente verkürzt als „xxx“-Tag bezeichnet). Das Wertattribut „v“ des „admin_level“-Tags gibt die administrative Ebene an. Orte können als Punkt- oder Flächengeometrien vorliegen; sie sind über „place“-Tags kategorisiert. Oftmals sind Geoobjekte sowohl Ort als auch administratives Gebiet. Über „name“-Tags sind Namen und ggf. Sprachvarianten der Namen angegeben. Teilweise liegt (i.d.R. ohne Referenzdatum) die Einwohnerzahl über das „population“-Tag vor. Gliederungsinformationen können über „is_in“-Tags definiert sein. In Abhängigkeit vom Staat geben Tags (in Deutschland z.B. mit Schlüssel „de:amtlicher_gemeindeschluessel“) ggf. Auskunft über nationale Schlüssel.

Alle Elemente enthalten die Attribute „version“, „changeset“, „uid“, „user“ und „timestamp“ mit Informationen über die Herkunft und Aktualität. Importe sind ggf. über „source“-Tags gekennzeichnet. Quantitative Qualitätsindikatoren bezüglich Lage und Grenzverlauf existieren nicht. Auch variiert die tatsächliche Belegung der oben genannten Attribute. Durch geeignete Programme müssen die gewünschten Teildaten aus den vollständigen OSM-Dateien extrahiert werden, was aufgrund des recht komplexen Modells für Multipolygone, teilweise fehlender Geometrieteile und großen Dateien eine nicht immer triviale Aufgabe darstellt.

**Wikipedia** stellt zu vielen Artikeln eine (oder selten auch mehrere) WGS’84-Koordinaten zur Verfügung. Eine Möglichkeit des Zugriffs ist der kategorieweise KML-Export über den Tool Server:

http://toolserver.org/~para/cgi-bin/kmlexport?project=<*language*>&article=<*category*>

Optional kann zudem über den Parameter *l* die maximale Zahl von Rekursionen bezüglich der Kategorien angegeben werden. Das Resultat ist eine KML-Datei mit „Placemark“-Elementen, in denen eingebettet das „name“-Element den Artikelnamen, das „Point“-Element die Koordinaten und das „description“-Element einen Link auf den Wikipedia-Artikel enthält. Eine andere Möglichkeit stellt die Auswertung von Wikipedia-Dumps dar, was allerdings aufwändiger ist und – außer einer ID – i.d.R. keinen inhaltlichen Mehrwert liefert.

Eine administrative Zuordnung der Ortsangaben kann – wenn eine entsprechende Kategorisierung der Artikel vorliegt – durch den Aufruf des KML-Exports erzielt werden, z.B.

http://toolserver.org/~para/cgi-bin/kmlexport?project=de&article=Kategorie:Gemeinde in Niedersachsen

**Amtliche offene Geodaten** haben im Gegensatz zu den vorgenannten Datensätzen weder eine weltweite Abdeckung noch ein zentrales Zugangsportal. Sie liegen in keinem einheitlichen Datenmodell vor. Die Einfachheit des Zugriffs auf die Daten variiert; oftmals nutzen die Daten eine nationale Projektion. Positive Aspekte sind oftmals hohe Lagegenauigkeiten, die Nutzung nationaler Schlüssel und die vollständige Angabe von administrativen Zugehörigkeiten.

### Vollständigkeit und Eindeutigkeiten in GeoNames und OSM

Für die folgenden Untersuchungen werden vier Staaten betrachtet: Argentinien, China, Deutschland und Neuseeland.

Tabelle 1 zeigt für diese vier Staaten wesentliche Eigenschaften für GeoNames (Stand 5.2.2015). Die Belegung der administrativen Zugehörigkeiten ist teilweise sehr gering. Damit lassen sich insbesondere in China (33%), aber auch in Deutschland (12,6%) viele Orte nicht eindeutig über Angabe des Namens und der administrativen Zugehörigkeit geokodieren (d.h. mehrere Orte mit unterschiedlicher Lage werden bestimmt). Die Spalte „Letzte Änderung“ gibt den durchschnittlichen Zeitraum seit der letzten Modifikation in Jahren an.

| Land | Features gesamt | Feature Class "P" und "A" | Adm1 belegt | Adm2 belegt | Adm3 belegt | nicht eindeutig | Letzte Änderung |
|---|---|---|---|---|---|---|---|
| Argentinien | 47.150 | 7.458 | 99,96% | 16,73% | 0,08% | 0,46% | 5,70 |
| China | 447.884 | 414.124 | 99,93% | 0,12% | 0,70% | 32,95% | 3,21 |
| Deutschland | 182.058 | 91.450 | 99,99% | 39,78% | 53,96% | 12,60% | 3,21 |
| Neuseeland | 67.444 | 3.395 | 89,66% | 77,64% | 0,06% | 2,24% | 4,29 |

Tabelle 1: Eigenschaften von Orten und administrativen Einheiten in GeoNames

Tabelle 2 zeigt ähnliche Eigenschaften für OpenStreetMap-Daten (Stand 5.2.2015), wobei Deutschland sich auf Niedersachsen beschränkt. Ausgewählt wurden „node“-Elemente, die ein „place“-Tag beinhalteten. Die Spalte „Adm1 belegt“ bezieht sich auf die jeweils relevanten „is_in“-Tags. Auch hier ist nicht immer eine eindeutige Geokodierung möglich (China: 29%). Nationale Schlüssel sind kaum vorhanden.

| Land | Anzahl Places | Adm1 belegt | nicht eindeutig | nationaler Schlüssel | letzte Änderung |
|---|---|---|---|---|---|
| Argentinien | 11.344 | 57,11% | 7,47% | 0,00% | 1,50 |
| China | 59.059 | 2,48% | 29,08% | 0,00% | 1,67 |
| D: Niedersachsen | 9.238 | 59,68% | 2,18% | 0,06% | 3,05 |
| Neuseeland | 2.572 | 24,46% | 3,54% | 0,00% | 2,97 |

Tabelle 2: Eigenschaften von punktförmigen Orten (places) in OSM

Tabelle 3 zeigt die Eigenschaften von administrativen Einheiten in den OSM-Daten. Die Verfügbarkeit von Daten schwankt international sehr stark. Dies gilt auch für das Vorhandensein von nationalen Schlüsseln. Die Spalte „Features mit Geometrie“ gibt die Anzahl der Geoobjekte an, bei denen sich die Geometrie tatsächlich konstruieren lässt.

| Land | Typ | tatsächl. Anzahl | Features | Features mit Geometrie | nationaler Schlüssel |
|---|---|---|---|---|---|
| Argentinien | Bezirke | 513 | 290 | 290 | 0 |
| China | Präfekturen | 341 | 330 | 321 | 0 |
| China | Bezirke | 2.853 | 2.427 | 2.395 | 0 |
| China | Gemeinden | 43.575 | 615 | 608 | 0 |
| D: Niedersachsen | Gemeinden | 988 | 988 | 988 | 988 |
| Neuseeland | Bezirke | 67 | 61 | 59 | 0 |

Tabelle 3: Eigenschaften von administrativen Gebieten in OSM

### Werkzeug zur Untersuchung von Wikipedia-Koordinaten

Wie zuvor erwähnt, enthalten die Wikipedia-Daten die administrative Zugehörigkeit nur indirekt. Nationale Schlüssel lassen sich über den KML-Export nicht gewinnen. Damit sind für einen Lagevergleich mit amtlichen Geodaten folgende Funktionalitäten zweckmäßig:

- Laden der amtlichen Geodaten über Shapefiles.
- Einlesen der Wikipedia-Daten über den o.g. KML-Export:
  - Sprache und Kategorie können frei eingeben werden.
- Verwaltungsgebiete geometrisch bestimmen:
  - Es werden zwei Layer ausgewählt: Layer 1 mit Punktkoordinaten und Layer 2 mit Flächengeometrien. Bei Layer 2 wird zudem ein Namensattribut interaktiv bestimmt. Layer 1 wird um ein String-Attribut „admin“ ergänzt. Der Wert dieses Attributs wird aus dem Namensattribut des Features bestimmt, dessen Fläche den Punkt enthält bzw. schneidet.
- Qualitätsanalyse der Wikipedia-Georeferenzierung:
  - Der Benutzer wählt dazu das Wikipedia-Layer und das Layer mit den Referenzdaten. Folgende Fälle sind zu beachten: Das Wikipedia-Layer enthält das Attribut „name“ sowie ein oder kein „admin“-Attribut (Namenskonvention).
  - Das Referenz-Layer enthält immer ein Schlüsselattribut, ein Namensattribut sowie ein oder kein Attribut für das Verwaltungsgebiet. Schlüssel-, Namens- und optionales Attribut mit Angabe des Verwaltungsgebietes werden interaktiv ausgewählt (keine Namenskonvention).
  - Im ersten Schritt werden die zugehörigen Objektpaare aus Wikipedia- und Referenz-Layer bestimmt, indem eindeutige Paare aufgrund Gleichheit bezüglich Namen und (wenn gegeben) Verwaltungsgebiet bestimmt werden. Falls für den Namen eines Wikipedia-Objektes, das eine Ergänzung in Klammern besitzt (z.B. „Holzhausen (Wildeshausen)“) kein Partner im Referenz-Layer mit gleichem Namen gefunden wird, ist der Test auch ohne die Klammerergänzung (also z.B. mit „Holzhausen“) durchzuführen.
  - Es wird der horizontale und vertikale Abstand zwischen den minimalen und maximalen x- und y-Koordinaten im Referenz-Layer berechnet und davon der Mittelwert als Referenzabstand *r* genommen.
  - Für die Objektpaare wird der jeweilige Abstand *d* bestimmt. Falls $d=0$ ist die Abstandsklasse 0; falls $0 < d <$ r/100000 ist die Abstandsklasse 1; falls $r/100000 < d < r/10000$ ist die Abstandsklasse 2; falls $r/10000 < d < r/1000$ ist die Abstandsklasse 3; falls $r/1000 < d < r/100$ ist die Abstandsklasse 4; falls $r/100 < d < r/10$ ist die Abstandsklasse 5 und falls $d > r/10$ die Abstandsklasse 6.
  - Die Wikipedia-Objekte aus den Objektpaaren werden um die Attribute Abstand, Abstandsklasse und Kennnummer des Referenzpartners ergänzt und als neues Layer der Karte hinzugefügt. Die Gestaltung der Objekte erfolgt gemäß der Abstandsklasse.

Im Rahmen einer Abschlussaufgabe im Wintersemester 2013/14 für das Modul „GIS-Programmierung“ (5. Semester Bachelor-Studiengang Geoinformatik) sollten die Studierenden auf Basis von GeoTools

und JTS eine entsprechende Anwendung entwickeln. Zu Untersuchungszwecken wurden folgende amtlichen offenen Geodaten bereitgestellt (jeweils mit nationalem Schlüssel):

- Gemeinden und Kreise in Deutschland (12.744 bzw. 439 Flächenobjekte)
- Urban Areas und Rural Centers in Neuseeland (276 Flächen)
- Localidades (3.560 Punktobjekte) und Departamentos (526 Flächen) in Argentinien

Die nachfolgenden Abbildungen zeigen exemplarisch eine so entwickelte Anwendung:

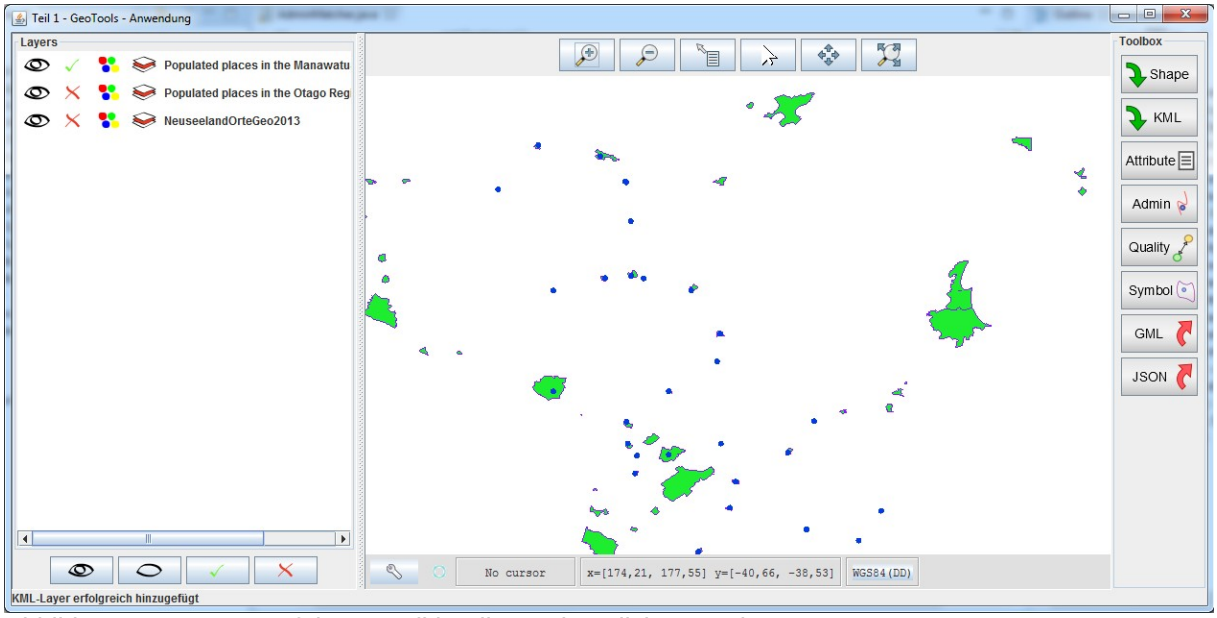



Abbildung 1: Zusammenführung Wikipedia- und amtliche Geodaten

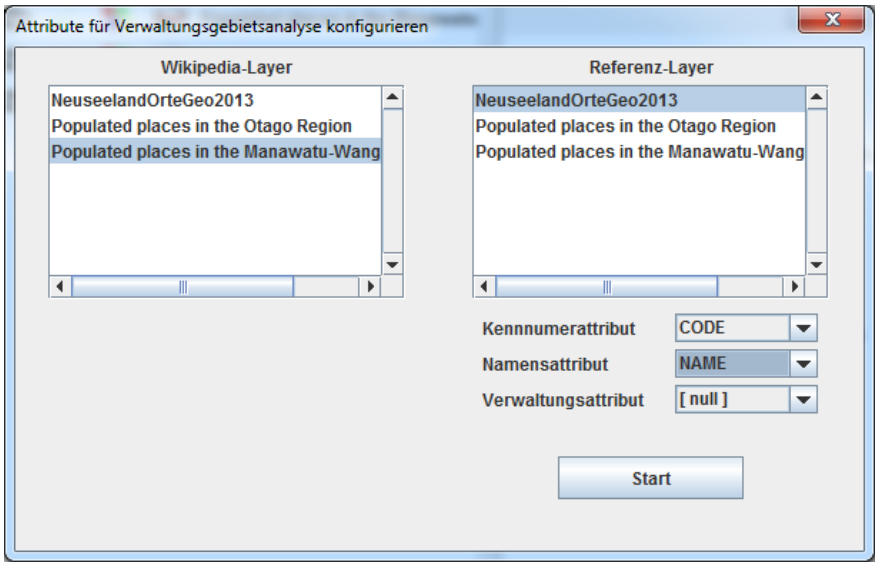



Abbildung 2: Konfiguration eines Vergleichs

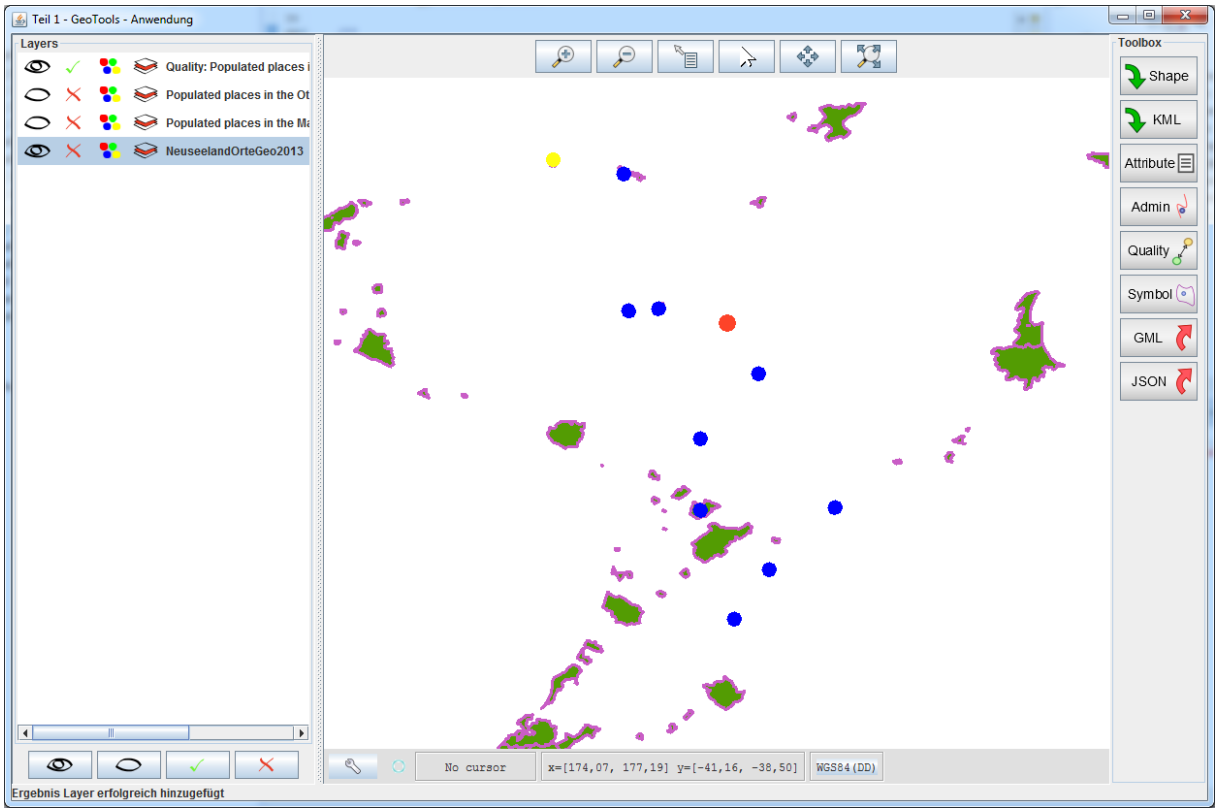



Abbildung 3: Resultat des Vergleichs (blau: Abstandsklasse 0, gelb: Abstandsklasse 1, rot: Abstandsklasse 3)

## Fazit

Die Verfügbarkeit und Qualität von offenen Geodaten für Orte und administrative Gebiete ist recht heterogen. Durch die Aufnahme offener amtlicher Geodaten in die Datenbestände von gemeinschaftlichen Kartierungen konnte teilweise eine deutliche Steigerung der Qualität erreicht werden (z.B. durch den Import der BKG-Daten in OpenStreetMap). Weltweit ist die Situation noch unbefriedigend. Die Verknüpfung von GeoNames- und Wikipedia-Daten beispielsweise mit OSM-Daten [3] birgt aufgrund der beobachteten Eigenschaften durchaus Risiken. Durch freie Software-Bibliotheken lassen sich aber mächtige Werkzeuge schaffen, um solche Geodaten trotzdem anwendungsorientiert nutzen zu können.

Kontakt zum Autor:

Prof. Dr. Thomas Brinkhoff
Jade Hochschule Wilhelmshaven/Oldenburg/Elsfleth
Institut für Angewandte Photogrammetrie und Geoinformatik
Ofener Str. 16/10
26122 Oldenburg
thomas.brinkhoff@jade-hs.de
http://iapg.jade-hs.de/personen/brinkhoff/

Literatur

[1] *Heipke, Christian*: „Crowdsourcing geospatial data“, ISPRS Journal of Photogrammetry and Remote Sensing 65(6), 2010, 550-557.

[2] *Open Source Geospatial Foundation (OSGeo)*: OSGeo Projects, http://www.osgeo.org/

[3] *Stadler, C., Lehmann, J., Höffner, K., Auer, S.*: „Linkedgeodata: A core for a web of spatial open data. Semantic Web“, 3(4), 2012, 333-354.

# Neues vom QGIS Print Composer und Atlas-Seriendruck

Andreas Neumann

Der Vortrag fasst die in den letzten QGIS-Versionen eingeführten Verbesserungen im Print Composer und im Atlas-Seriendruck zusammen. Pläne für die Entwicklung einer Reporting-Engine in einer zukünftigen QGIS-Version werden vorgestellt.

Die meisten bestehenden Elemente im QGIS print composer haben Detailverbesserungen erhalten. Ausserdem wurde der Umgang mit mehrseitigen Layouts verbessert. So können Tabellen und Rich-Text Inhalte (HTML-Frame) besser über mehrere Seiten fliessen. Viele Eigenschaften von Elementen im Kartenlayout (z.b. Dateipfade, Position, Grösse, HTML-Inhalte) können neu aufgrund von Datenspalten und Berechnungen (QGIS Expressions) definiert werden. Kartengitter können nun in anderen Koordinatensystem definiert werden und mehrere Kartengitter können übereinander gelegt werden (z.b. UTM/Meter und Grad/Minuten/Sekunden im WGS84 System). Fotos/Grafiken haben verschiedene Skalierungsmodi und Ankerpunkte erhalten - v.a. nützlich wenn diese in Seriendrucken verwendet werden. Ausserdem können Grafiken direkt aus dem Web eingebunden werden.

Der Atlas-Seriendruck hat verschiedene Verbesserungen erhalten: Geometrien welche im Zusammenhang mit dem jeweiligen Atlas-Feature stehen können unterschiedlich symbolisiert werden. Es bestehen verbesserte Filteroptionen für die Atlas-Feature-Liste, die Kartenansicht und Tabellen - ebenfalls im Zusammenhang mit dem jeweiligen Atlas-Feature. Seriendrucke können auch unterschiedliche Seitenformate aufweisen und die Masstäbe der Kartenansichten können auf optimale runde Massstabszahlen festgelegt werden.

Ein Ausblick erläutert die Pläne für die Einführung einer neuen Reporting Engine in den nächsten QGIS Versionen.

# Geonetzwerk metropoleRuhr

David Arndt

**Stellen Sie sich vor, alle raumbedeutsamen Daten in der Region sind mit einem Mausklick erreichbar. 53 Mausklicks, der 53 Gemeinden, werden durch Einen ersetzt!**

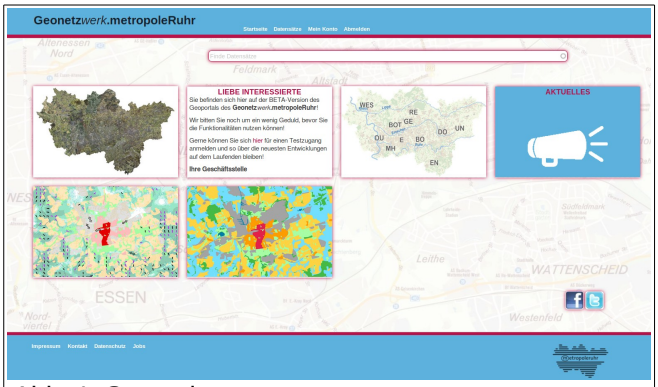


Abb. 1: Startseite

Daran arbeitet seit 2013 das Projekt **Geonetz*werk*.metropoleRuhr**. Das Kooperationsprojekt, verbindlich durch die Unterzeichnung einer Kooperationsvereinbarung im Dezember 2013 gestartet, hat das Ziel ein Netzwerk zu bilden. Dieses Netzwerk aus den 11 kreisfreien Städten, 4 Kreisen der Metropole Ruhr und dem Regionalverband Ruhr soll regional bedeutsame Geodaten bereitstellen und diese der Bevölkerung, der Wirtschaft und den Verwaltungen austausch- und nutzbar machen.

Durch gemeinsam initiierte Veranstaltungen, Workshops und Fortbildungen findet ein regelmäßiger Wissenstransfer über aktuelle Themen statt. Die Geschäftsstelle des Projektes, welche beim Regionalverband Ruhr angesiedelt ist, vernetzt die konkreten Arbeitsschritte mit den Zielvorgaben und gilt als Ansprechpartner in allen Fragen rund um Geodaten und Geoinformation. Weiterhin koordiniert und organisiert sie die internen Veranstaltungen, präsentiert die Arbeiten in Fachforen oder Kongressen und bündelt die Interessen der Mitglieder. Um die Geodaten einer breiten Öffentlichkeit erfahrbar und für Anwender nutzbar zu machen, wird ein Geoportal mit einem integrierten Mitgliederbereich aufgebaut.

Hier werden zukünftig kommunal und regional bedeutsame Geodaten für die Metropole gebündelt. Beispielsweise können regional flächendeckende Informationen über die rechtskräftigen Bebauungspläne in der Metropole Ruhr so zeitnah mit anderen hinterlegten Daten (Luftbilder, Fachdaten) verknüpft werden und so die (verwaltungsinternen) Erarbeitungs- und Beteiligungsprozesse langfristig unterstützen.

Welche Vorteile bieten digitale und frei verfügbare regionale Geodaten für die Stadt von morgen?

- Schaffung einer Plattform für einen demokratischen Wissensaustausch
- Veröffentlichung von Planungs- und Fachdaten mit einem Klick werden möglich
- Durch eine flächendeckende und regionale Bereitstellung der Daten werden interkommunale Abstimmungsprozesse erleichtert, da der Blick über die eigenen Kommunengrenzen hinaus ermöglicht wird
- Informationen über die Region können zu unterschiedlichen Themenschwerpunkten zentral eingesehen und miteinander kombiniert werden

- Flexible Bereitstellung von Information für Bürger, Nachbarkommunen, der Wirtschaft und damit Schaffung von Transparenz, Förderung von öffentlicher Akzeptanz für Verwaltungsentscheidungen
- Die stärkere Erlebbarkeit der Region führt zu einer verstärkten Identifikation mit der Metropole Ruhr
- Die hohe Standorttransparenz erleichtert den Wirtschaftsunternehmen in der Region und ansiedlungswilligen Unternehmen die Standortplanung

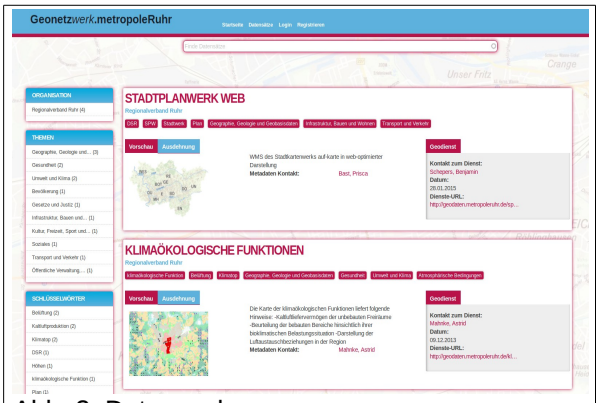


Abb. 2: Datensuche

Technische Basis des Portals bildet das Content Management System Drupal. Dieses bildet die Oberfläche zur Nutzung der einzelnen Komponenten:

- Metadateneditor
- WMS/WFS-Dienste Erstellung
- Veranstaltungsplanung
- Bereitstellung von aktuellen Informationen

Abb. 3: Funktionen

Ziel ist der Aufbau einer modularen Technik, die es ermöglicht, einzelne Komponenten auszutauschen oder wegzulassen. Hierbei kommen die verbreiteten OGC-Standards, WMS, WFS, GML, SLD, CSW usw. zum Einsatz, die eine weitgehende Unabhängigkeit von der einzusetzenden Software ermöglichen.

Um die Lizenzkosten gering zu halten, und den Mitgliedskommunen die Möglichkeit zu bieten die technischen Entwicklungen, die im Rahmen des **Geonetz*werk* metropoleRuhr** entwickelt werden, kostengünstig weiter zu nutzen, setzen wir überwiegend Open-Source Software ein.

Kontakt zum Autor:

David Arndt
Regionalverband Ruhr – Referat 9 – **Geonetz*werk* metropoleRuhr**
Kronprinzenstraße 35
45128 Essen

0201/2069-412
geonetzwerk@rvr-online.de

# OSM-Tagging in Wikidata

Tim Alder

Der Vortrag soll beschreiben wie das OSM-Taggingschemata in Wikidata kam, wozu das Ganze zukünftig gut sein könnte und welche Unterschiede es zwischen beiden Projekten gibt.

Wikidata ist die freie Datensammlung aus dem Wikipedia Umfeld.

Das Vorhaben das OSM-Tagging in Wikidata zu hinterlegen stellt eine Ergänzung zu dem Verlinken von OSM-Objekten mit Wikipedia/Wikidata-Tags dar, es wird vielmehr versucht die übergeordneten OSM-Objektklassen mit Wikidata zu verbinden. Dazu wurde in Wikidata ein neues Property für das OSM-Tagging angelegt (Property:P1282) und in diesem wird das Tagging als Key oder Tag hinterlegt. Beispiel für "Leuchtturm" "P1282" = "Tag:man_made=lighthouse". Vereinfacht gesagt werden von Wikidata aus Links zum OSM-Wiki angelegt.

Eine Anwendung könnte sein, die Wikipedia Artikel in den verschiedenen Sprachen als Übersetzungen mit anzuzeigen. Es gibt eine Vielzahl von Fällen, bei dennen es in der Wkipedia mehr als 20 Sprachversionen gibt, im OSM-Wiki aber vielleicht nur 5 Übersetzungen existieren. Gerade für OSM-Mitwirkenden ohne sehr gute Englischkenntnisse könnte es eine Hilfe sein, wenn er nach Begriffen in seiner Muttersprache suchen könnte.

Eine andere Anwendung stellt eine weitere Automatisierung des Matchings von OSM- und Wikipediaobjekten darstellen. Wenn wir also z.B. wissen, dass alle mit man_made=lighthouse getaggten Objekte in Wikidata zur Klasse "Leuchturm" gehören, können wir sie einfacher mit den Mitgliedern dieser Klasse verknüpfen. Dass das Arbeiten mit den Klassen von Wikidata sehr gut funktioniert, wurde bereits im Bereich der Wikipedia-Geoobjekte gezeigt. Über ein kleines Skript wurde bereits eine Anbindung and Tag-info für das Tagging in Wikidata realisiert. Da Wikidata eine Verbindung zu einer Reihe anderer Datenbanken darstellt (Freebase, GND, BNCF, ...), entstehen vielleicht noch nicht absehbare Projekte.

Der Vortrag wird ein paar Beispiele (Passstraße, Trafo, Tanzeinrichtung, Schuhverkäufer, Fahrradladen) aufführen, wo das Matching zwischen beiden Projekten schlecht funktioniert. Teilweise fehlen Wkipedia-Artikel. Eine 1:1-Umsetzung des OSM-Wikis kann der Links zur Wikipedia nicht sein, somit bleibt weiterhin das OSM-Wiki die verbindliche Beschreibungsseite des OSM-Taggings.

**Links**

- http://taginfo.openstreetmap.org/projects/wikidata_org#tags
- https://toolserver.org/~kolossos/wikidata/superclasses.php
- https://www.wikidata.org/wiki/Property:P1282

# QField for QGIS / Eine native Benutzerschnittstelle für mobile touch Geräten.

Marco Bernasocchi | Matthias Kuhn – OPENGIS.ch GmbH

Die Ubiquität von mobilen Geräten aller Grössen ist in den letzten Jahren enorm gestiegen. Mit weltweit mehr als 2 Milliarden verkauften Mobile Geräten bis 2014 [0] und einem stetig wachsenden Markt kommen folglich auch immer mehr solcher Geräte in der Arbeitswelt zum Einsatz.

Intuitive Bedienung, erhöte Handlichkeit, integriertes GPS und (relativ) geringe Anschaffungskosten machen Heute den Kauf mobiler Geräte schmackhafter und einfacher den je.

Dank seiner Multi-plattform Natur (Win, Mac, Linux und Android) und seinen breiten Features-Set (Desktop, Server, Web-Client), ist QGIS eine der verbreitesten Open-Source GIS Softwaren und wird bereits von vielen Institutionen benutzt. Die Ergänzung der QGIS Suite mit einer native touch User Interface bietet den Anwendern eine vollwertige mobile GIS Daten Verwaltungsinfrastruktur.

Durch die Erfahrungen bei der Entwicklung von QGIS für Android haben wir bei OPENGIS.ch herausgefunden was für eine mobile Anwendung notwendig ist und funktioniert. Vor allem haben wir Erkenntisse darüber gewonnen was absolut zu vermeiden ist: Komplexität, kleine UI Elemente und Projektdefinitionsarbeit auf mobilen Geräten.

Unser Motto bei der Entwicklung unserer neuen UI ist : “Less is more” . Dank vordefinierten Modi (Datenerhebung, Datenprüfung, Vermessungen, etc.) und klaren Bedienungselementen können sich somit die Anwender in jeder Situation auf ihre Aufgaben konzentrieren.

Um den Arbeitsablauf noch weiter zu vereinfachen entwickeln wir ein neues offline Synchronisationtool. Dieses ermöglicht einen nahtlose Datenaustausch zwischen dem mobilen Gerät und der vorhandenen Infrastruktur.

### Demos

http://qfield.opengis.ch/demos

### Entwicklungsleitfaden

- Für Feldarbeit optimiert
- GPS basiert
- Vollständig funktionell auch wenn offline
- Leichte Synchronisation möglich
- Einfache Projektvorbereitung auf dem Desktop
- Bedienungsfreundlich durch wenige und grosse Tasten

### Modale Paradigma

- Unterschiedliche Betriebmodi möglich
  - Anzeige, Digitalisierung, Messung, Inspektion, ...
- Permanente Tools
  - Pan, Zoom, Identifikation

- Massstabsleiste, GPS-Koordinaten
- Mitell Fadenkreuz mit Fangfunktion

### Installation und Update (http://qfield.opengis.ch/get)

QField ist im Playstore vorhandenen und kann wie alle übliche Apps installiert und aktualisiert werden. Updates werden automatisch vom Playstore vorgeschlagen.

- Beim ersten Start, auf Anfrage Ministro II (Qt Library manager) installieren
- Anschließend zurück zu QField.
- Alfällige Installation der Bibliotheken akzeptieren
- QField Startet

### Benutzung

- Ein langer Touch (etwa eine Sekunde) auf dem Bildschirm erzeugt eine Selektion.
- Datenspeicherung: Jedes Android Gerät unterstützt einen gemeinsamen "externen Speicher", auf dem Sie Dateien speichern können. Dies kann ein Wechseldatenträger (z. B. eine SD-Karte) oder ein interner Speicher sein.
- Die QGIS Projekte und Daten werden meistens auf dem "externen Speicher" gesichert. Diese können über den Storage Link (falls vorhanden) im „open project“ Dialog geöffnet werden.
- Falls nicht vorhanden kann der externe Speicher mit mehrmaligen Klicks auf den “nach oben” Pfeil abgerufen werden (bis auf /sdcard)

### Features, Bugs und Spenden

Für die Weiterentwicklung, Behebung von Bugs oder für die Entwicklung neuer Features sind wir auf Sponsoren, resp. Kunden angewiesen.

Bei Interesse oder Fragen können Sie uns gerne unter info@opengis.ch kontaktieren.

### Technologien

- Qt5 und QtQuick 2 (offiziell durch das Qt Projekt empfohlen und von Digia unterstützt)
- Hillft QGIS auf Qt5 zu wechseln
- QtQuick 2 Controls UI + Ein QWidget für QgsMapCanvas
- Synergien mit QGIS for android (Librarien und Skripten)
- ArmV7a minimum

**QField for QGIS / Eine native Benutzerschnittstelle für mobile touch Geräten.**

## Screenshots


Illustration 1: Vollbild Karte

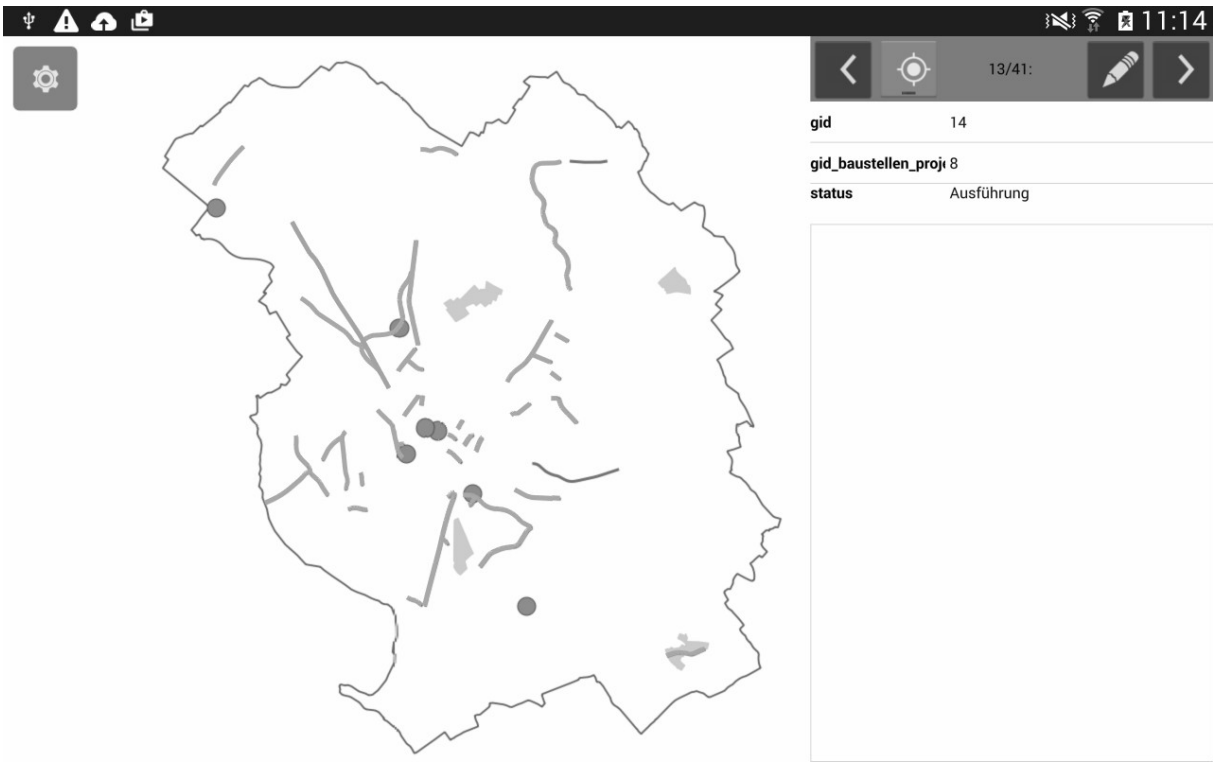



Illustration 2: Selektion

**QField for QGIS / Eine native Benutzerschnittstelle für mobile touch Geräten.**

**Kontakt zum Autor:**

Marco Bernasocchi
OPENGIS.ch GmbH
Via Cochetta 2
7152 Sagogn
Switzerland
+41794672470
info@opengis.ch

**Literatur**

[0] http://www.gartner.com/newsroom/id/2645115

# Spatial Index von Solr

## Erfahrungen und Einblicke im Umgang mit Solr Spatial Index am Beispiel des Geoportal.de

Tim Balschmiter

Seit der Version 4 bietet Solr Funktionen eines umfangreichen Spatial Indexes an. Am Beispiel des Geoportal.de sollen Erfahrungen und Einblicke in die Verwendung des Spatial Index mit Solr gegeben werden.

Die GDI-DE stellt Beschreibungen und Zugangsdaten zu Geodaten von Bund, Ländern und Kommunen bereit, welche über das Geoportal.de recherchiert werden können. Über eine Volltextsuche können derzeit ca. 130000 Datensätze gefunden werden. Für einen performanten Zugriff auf die Datensätze wird die Open Source Software Solr eingesetzt. Derzeit erfolgt die Suche ausschließlich über die themenbezogenen Inhalte der Metadaten. Hierfür finden u.a. die Attribute UUID, Titel, Abstract, Keywords und Kontaktdaten aus den Metadaten Berücksichtigung. Auf Basis einer individuellen Konfiguration erstellt Solr für jede Suchanfrage ein sogenanntes Scoring. Dieses ergibt sich aus der Anzahl gefundener Suchbegriffe und deren Position im Datensatz und bestimmt die Platzierung in der Ergebnisausgabe. Seit der Version 4 bietet Solr die Möglichkeit einer erweiterten geobasierte Suche an. Diese beinhaltet GIS-Funktionalitäten wie Distance, Contains, Intersects und Within zum Verarbeiten von Punkt-, Linien- und Flächengeometrien. Neben der Verwendung für das Scoring bietet es sich an die Suchergebnisse über diese Operationen gezielt vom Nutzer filtern und sortieren zu lassen. Um die GIS-Funktionen nutzen zu können werden die in den Metadaten angegebenen Bounding Boxen indexiert. Aktuell befinden sich diese Funktionen in der Testphase. Der Vortrag soll Einblicke und Erfahrungen im Umgang mit einem räumlichen Index auf Basis von Solr vermitteln.

# Ein Geowiki auf OSM-Basis

Klaus Stein, Christoph Schlieder

## Einleitung

Wikis erlauben uns, gemeinsam Informationen zu sammeln, Texte zu schreiben und zu verändern und diese online bereitzustellen. Dies geht vom internen Arbeitsgruppenwiki über das FossGis-Konferenzwiki und die Wookieepedia bis hin zur Wikipedia als allumfassendes Lexikon. Die OpenStreetMap-Datenbank leistet ähnliches für Geodaten: Nutzer erstellen kollaborativ Geoobjekte und versehen sie mit Zusatzinformationen.

Wir stellen hier ein Geowiki vor, das beide Aspekte, den text- und den kartenorientierten in einer Anwendung integriert.

## Was ist ein Geowiki?

Der Begriff Wiki (vgl. Leuf und Cunningham 2001) beschreibt zunächst sehr allgemein eine Webanwendung zum kollaborativen Erstellen und Bearbeiten von Inhalten. Daher wird von einigen Autoren (vgl. Boulos 2005, Chen et al 2006 und andere) jede Karte oder Geodatenbank, die Nutzerinhalte erlaubt, als Geowiki bezeichnet. Dies reicht von sehr stark eingeschränkten Anwendungen wie FixMyStreet[1] oder Wheelmap[2], bei denen der Nutzer nur Punkte oder Objekte auf der Karte markieren und mit Anmerkungen versehen kann bis zur OSM-Datenbank selbst mit ihren doch sehr komplexen Editiermöglichkeiten. Priedhorsky 2010 stellt konkretere Anforderungen an eine umfassendere Geowiki-Funktionalität auf, unter anderem freien Zugang und einfache Handhabbarkeit.

Den aufgeführten Projekten gemein ist der Fokus auf die Karte als primäre Anzeige der Daten, die Zusatzinformationen werden auf der Karte visualisiert oder in einem Tooltip oder einer Sidebar auf Klick hin angezeigt.

Umgekehrt gibt es Ansätze, klassische Wikis um Geoinformationen zu erweitern. Allein die Wikipedia bietet hier eine Vielzahl unterschiedlicher Möglichkeiten[3]. Zum einen existieren Koordinaten-Templates, um WGS84-Koordinaten als Punkte auf einer als Bild eingebundenen Karte einzuzeichnen und die Artikel selbst mit Geokoordinaten zu Taggen. Diese Informationen werden unter anderem vom Tool GeoHack[4] gecrawlt und visualisiert. Der in viele Seiten eingebundene WikiMiniAtlas[5] kann diese und weitere Informationen anzeigen. Zusätzlich erlaubt die deutschsprachige Wikipedia die direkte Einbindung einer OSM-Karte in die Seite. Auf dieser werden Links auf alle Wikipedia-Seiten angezeigt, die in der OSM-Datenbank entsprechend getagt[6] sind. Damit bietet die Wikipedia zwar eine relativ gute Integration von Geoinformationen, sie beschränkt sich aber auf die textuelle Angabe von Punktkoordinaten und den Durchlauf eines externen Tools oder erfordert externe Edits in OSM.

Somit firmieren unter dem Begriff Geowiki zwei unterschiedliche Konzepte: zum einen eine durch den Nutzer annotierbare oder auch veränderbare Karte, zum anderen ein klassisches Wiki, bei dem Kartendarstellungen in einzelne Seiten eingebunden werden.

## Ein integriertes Geowiki

Das hier vorgestellte Geowiki versucht beide Sichten zu vereinen. Abbildung 1 zeigt die Wiki-Seite der Mayerschen Gärtnerei in Bamberg. Der Text links liefert wie im klassischen Wiki Informationen zum Thema. Zudem enthält der Text jedoch Geolinks, die auf Objekte auf der Karte verweisen. Ein Klick

1 http://fixmystreet.ie
2 http://wheelmap.org/
3 http://de.wikipedia.org/wiki/Wikipedia:WikiProjekt_Georeferenzierung/Hauptseite
4 https://tools.wmflabs.org/geohack/geohack.php?params=51_57_31_N_7_37_26_E
5 http://meta.wikimedia.org/wiki/WikiMiniAtlas
6 z.B. <tag k="wikipedia" v="de:Bamberg"/>

auf so einen Link hebt das entsprechende Geoobjekt (Punkt, Polylinie oder Polygon oder eine Kombination aus diesen) auf der Karte graphisch hervor und zoomt und zentriert die Karte zudem auf Wunsch. Selbstverständlich kann der Nutzer die Karte jederzeit auch manuell verschieben und zoomen, wenn er dies wünscht. Umgekehrt zeigt ein Klick auf ein Geoobjekt auf der Karte in einem Tooltip sowohl nähere Informationen als auch Links auf alle Wiki-Seiten, auf denen dieses verlinkt ist. Diese enge Integration von Text und Karte bietet die Möglichkeit, einzelne Geoobjekte im Text ausführlicher und im Kontext eines größeren Textes zu beschreiben, was in Form von Tags nicht möglich wäre.

Da der Nutzer alle im Text beschriebenen Orte auf der Karte angezeigt bekommt, können Ortsbeschreibungen im Text entfallen, die Beschreibung wird übersichtlicher und informativer.

Auf Wunsch kann sich der Nutzer auf der Karte auch alle weiteren, nicht von dieser aber von anderen Wiki-Seiten verlinkten Geoobjekte anzeigen lassen. Durch die Links von Geoobjekten auf Wiki-Seiten bietet sich hiermit eine weitere Ebene der Navigation im Wiki auf Basis räumlicher Nähe.

Sowohl der Text als auch die Geoobjekte können direkt im Wiki editiert werden. Sie werden hierbei in einer eigenen Datenbank abgelegt und nicht direkt nach OSM gespeichert, wogegen ein Import von Objekten *aus* OSM vorgesehen ist. Wir haben aus mehreren Gründen auf einen Schreibzugriff auf die OSM-Datenbank verzichtet: Viele der eingezeichneten Objekte sind nur im Kontext des Wikis sinnvoll, da sie sich auf die Beschreibungen im Text beziehen. Sie haben somit in der OSM-Datenbank nichts verloren. Umgekehrt kann die Software so auch für interne Wikis mit privaten Daten genutzt werden. Und schließlich sinkt dadurch die Hemmschwelle, sich am Wiki zu beteiligen, wenn sie nicht fürchten müssen, dadurch etwas in OSM kaputtzumachen.

**Umsetzung**

Das Wiki wurde als Ruby on Rails[7] Anwendung realisiert und nutzt auf Clientseite Leaflet[8] zur Kartendarstellung. Auf Serverseite setzen wir für die Verarbeitung der Geodaten RGeo[9] mit GEOS ein, um die notwendigen Bereichsanfragen zur Darstellung der im Kartenausschnitt sichtbaren Geoobjekte durchführen zu können. Die Geodaten selbst werden als WKT als Attribut eines Rails-Models in einer von Rails unterstützten Datenbank hinterlegt. Wir hatten hier zunächst mit PostGIS experimentiert, konnten aber feststellen, dass für die zu erwartenden Anzahlen von wenigen tausend Geoobjekten ein DB-Geoindex keine Geschwindigkeitsvorteile bringt. Daher haben wir die die Bereichsbestimmung von der Datenbank in den Rails-Server verlagert, wodurch neben Postgres auch Mysql und Sqlite als Datenbank genutzt werden können. Für Anwendungen mit sehr vielen Geodaten ist eine PostGIS-Nutzung mit nur wenigen Anpassungen jederzeit möglich.

Der Datenaustausch mit dem Client erfolgt mittels GeoJSON, da dies auf Serverseite von RGeo und auf Clientseite von Leaflet direkt unterstützt wird. Dennoch war eine Vorverarbeitung notwendig, da Leaflet beim Umgang mit GeoJSON-Collections einige Einschränkungen aufweist.

Die Verwaltung eines komplexen Geoobjekts, z.B. eines Gebäudekomplexes mit mehreren Gebäuden, als eine Geometry Collection erwies sich als einfach und effizient. Allerdings können so Teile nicht gesondert ausgezeichnet werden, da Attribute wie Farbe oder Liniendicke sich immer auf das ganze Geoobjekt beziehen. Hierarchische Beziehungen, d.h. ein Geoobjekt aus mehreren Einzelobjekten, wären technisch nicht sehr schwierig umsetzbar, überfordern nach unserer Erfahrung allerdings häufig den Nutzer.

Die Texteingabe auf dem Client haben wir mittels TinyMCE[10] als WYSIWYG umgesetzt. Die Entwicklung eines eigenen Plugins erwies sich hier als unerwartet schwierig, da wir für die Einbindung der Links auf Geoobjekte einen eigenen Linktyp benötigten und (zumindest in Version 3) hierzu passende Hooks fehlen. Die Geoobjekte selbst werden vom Nutzer in einem auf Leaflet Draw aufsetzenden Karteneditor erstellt und bearbeitet, der direkt von TinyMCE aus aufgerufen werden kann.

Der Editor bietet die Möglichkeit, Geometrien direkt aus OSM zu importieren. Hierzu werden Daten mittels Overpass-API[11] abgerufen. Um Probleme mit der browserseitigen Same-Site-Policy zu vermei-

7 http://rubyonrails.org/
8 http://leafletjs.com/
9 https://github.com/rgeo/rgeo
10 http://www.tinymce.com/
11 http://overpass-api.de/

den stellt der Client die Anfrage hierbei an den Rails-Server, der sie in eine Overpass-Abfrage umwandelt und forwarded. Dies hat zudem den Vorteil, dass sich clientseitig die Abfragen nicht von denen lokal verwalteter Geoobjekte unterscheiden. Das vereinfacht den Client-Code und bietet dem Rails-Server zudem die Möglichkeit, Geodaten aus weiteren Quellen einfach bereitzustellen.

Unser Geowiki entstand im Rahmen des BMBF[12]-Projektes EMN-Moves als Teil eines größeren Softwareprojektes. Zur Zeit arbeiten wir an einer Stand-Alone-Version des Geowikis, die wir unter http://www.kinf.wiai.uni-bamberg.de/geowiki unter einer freien Lizenz veröffentlichen.

Kontakt zum Autor:

Dr. Klaus Stein
Prof. Dr. Christoph Schlieder
Otto-Friedrich-Universität Bamberg
An der Weberei 5
96045 Bamberg
+49 951 863-2845
klaus.stein@uni-bamberg.de
christoph.schlieder@uni-bamberg.de

**Literatur**

[1] *Leuf, B.; Cunningham, W*. The Wiki way: quick collaboration on the Web, 2001.

[2] *Boulos, M. N*. Web GIS in practice III: creating a simple interactive map of England's strategic Health Authorities using Google Maps API, Google Earth KML, and MSN Virtual Earth Map Control. International Journal of Health Geographics, 4(1), 22. 1–8, 2005.

[3] *Chen, W. Y.; Lee, B. N.; Chang, E. Y.* Fotowiki: distributedmap enhancement service. In Proceedings of the 14th Annual ACM international Conference on Multimedia, 803–804. 2006.

[4] *Priedhorsky, R. R*. The value of geographic wikis Doctoral dissertation, University of Minnesota. 2010.

12 Bundesministerium für Bildung und Forschung. Gefördert im Rahmen der Ausschreibung „Mobil bis ins hohe Alter – nahtlose Mobilitätsketten zur Beseitigung, Umgehung und Überwindung von Mobilitätsbarrieren“

# OpenLayers 3

## Stand, Neues & Zukünfiges

Andreas Hocevar, Bart ven den Eijnden, Marc Jansen

OpenLayers 3 wird vorgestellt, zahlreiche Beispiele zeigen die Verwendung. Unterschiede zur Vorgängerversion werden aufgezeigt und auch wo und warum OpenLayers 3 anders ist. Aktuelle Entwicklungen, wie etwa das OL3-Cesium Project, welches die dritte Dimension für OpenLayers zuänglich macht, werden präsentiert und ein Blick in die zukünftige Entwicklung gewagt.

OpenLayers 3 (der Nachfolger des weitverbreiteten OpenLayers 2) liegt nach längerer Entwicklungszeit seit Ende August 2014 in der Version 3.0.0 vor und bringt als hoch performante und vielfältige JavaScript Bibliothek alles mit, was man für moderne Kartenanwendungen im Web benötigt.

Der Vortrag wird OpenLayers 3 vorstellen und zahlreiche Beispiele der Verwendung zeigen. Auf Unterschiede zur Vorgängerversion wird eingegangen werden, und wir werden zeigen, wo und warum OpenLayers 3 anders ist. Wir werden aktuelle Entwicklungen -- wie etwa das OL3-Cesium Project, welches die dritte Dimension für OpenLayers zuänglich macht -- präsentieren und auch einen Blick in die zukünftige Entwicklung wagen.

# GRASS Funktionalität in QGIS nutzen

## Möglichkeiten der optimalen Interaktion von QGIS und GRASS

Otto Dassau, Sören Gebbert

Seit 2005 ist GRASS GIS in QGIS über das GRASS Plugin integriert und stellt hunderte Analysemethoden über QGIS bereit - ergänzend mittlerweile auch über das Processing und WPS Plugin. Wir vergleichen die drei Varianten und schauen in die Zukunft.

Seit 2005 ist GRASS GIS Funktionalität in QGIS über das GRASS Plugin integriert und bietet die Möglichkeit, hunderte, bewährte GRASS Module zur Datenanalyse und -modellierung über die benutzerfreundliche QGIS GUI anwenden zu können. Durch das Processing und das WPS Plugin ist diese Interaktion in den letzten Jahren durch zwei neue, völlig unterschiedliche Ansätze ergänzt und erweitert worden.

Wir starten in diesem Vortrag einen Vergleich der drei Möglichkeiten zur Interaktion von GRASS und QGIS. Anhand von Beispielen stellen wir die Vor- und Nachteile dar und versuchen aufzuzeigen, welche Variante in welcher Situation die beste Lösung darstellt. Wann ist es sinnvoll das Processing Plugin zu verwenden, wann sollte man auf das bewährte GRASS Plugin setzen und welche Stärken und Schwachstellen bietet die Integration von GRASS Funktionalität über das WPS Plugin.

Als Abschluss stellen wir die Aktuelle Entwicklung der Interaktion zwischen GRASS und QGIS vor und versuchen einen Blick in die zukünftige Entwicklung zu werfen.

# OpenStreetMap und die Kunst, die Welt zu ordnen

**Sprachliche und logische Aspekte des Taggingschemas**

Falk Zscheile

## 1 Erfassung von Geodaten

Einer der großen Unterschiede von OpenStreetMap zur klassischen Geoinformationserfassung ist es, dass kein hierarchisch gegliedertes System zur Erfassung geographischer Informationen von Objekten existiert.

### 1.1 Herangehensweise der klassischen Kartographie

Bei den topographischen Landkarten existiert mit dem *Musterblatt für die Topographische Landkarte 1:25000* eine Anleitung, wie geographische Informationen in der Karte darzustellen sind. Daraus ergibt sich gleichermaßen auch eine Strukturierung und Gliederung der Informationen. Durch die Zusammenstellung im Musterblatt wird gleichzeitig mitgeteilt, welche Informationen überhaupt in die Karte aufgenommen werden. Was nicht als Darstellungsform existiert, erscheint auch nicht in der Karte. Entsprechend werden auch nur die für die Darstellung notwendigen Informationen im Feld erhoben. Dies folgt bereits aus dem Aspekt der Wirtschaftlichkeit und Sparsamkeit der Aufgabenerfüllung durch eine Behörde, § 7 Abs. 1 Satz 1 BHO bzw. die entsprechenden Bestimmungen in den Landeshaushaltsordnungen, z.B. § 7 Abs. 1 SäHO.

Um das Auffinden der Darstellungsvorschriften für einzelne geographische Informationen zu erleichtern, wurde eine Gliederung vorgenommen, die sich über das Inhaltsverzeichnis oder das Sachregister erschließen lässt[6]. Eine vergleichbare Strukturierung geographischer Informationen mit Schwerpunkt Schifffahrt nimmt auch die vom Bundesamt für Seeschifffahrt und Hydrographie herausgegebene *Internationale Kartenserie Karte 1 (INT 1)* vor.

### 1.2 Herangehensweise von OpenStreetMap

Von der eben skizzierten klassischen Herangehensweise unterscheidet sich die Erfassung und Einteilung geographischer Informationen bei OpenStreetMap erheblich.

#### 1.2.1 Erfassung von Daten mittels Crowdsourcing

Bei OpenStreetMap dürfen im Prinzip alle Mitwirkenden ihre Informationen so erfassen, wie sie es für richtig halten. Der Mitwirkende entscheidet zunächst ad hoc, wie er ein geographisches Objekt in der Datenbank erfasst. Es existiert kein auf abstrakten Überlegungen und von der Aufgabe her determiniertes hierarchisch strukturiertes Kategorienschema. Dies ist die wichtigste Konsequenz aus dem Crowdsourcingprinzip[4], nach welchem OpenStreetMap arbeitet. Im Zusammenhang mit geographischen Informationen spricht man hier auch von „Volunteered Geographic Information“ (VGI).

Wie ein Objekt „richtig“ in der Datenbank erfasst wird, stellt sich erst nach und nach heraus. Dies hängt von der Frage ab, wie sich nachfolgende Mitwirkende bei der Eintragung eigener geographischer Objekte verhalten. Übernehmen andere Mitwirkende die ad hoc Entscheidung ihrer Vorgänger oder treffen sie eine eigene davon abweichende ad hoc Entscheidung bei der Eintragung? Werden also bereits getroffene Kategorien für geographische Objekte übernommen oder wird eine eigene alternative Kategorie gebildet und für die Eintragung des Objekts in die Datenbank verwendet? Im letzteren Fall wird also eine alternative Form der Erfassung gewählt und es stellt sich erneut die Frage, ob diese Form der Erfassung wiederum von anderen Mitwirkenden erfasst wird. Diesem Vorgehen liegt im Wesentlichen die Idee zu Grunde, wie sie sich in der inkrementellen Softwareentwicklung findet[vgl. 3, S. 314].

Einen gewissen korrigierenden Einfluss auf die Neuerfindung oder Verwendung bereits bestehender Objektkategorien kommen dabei dem OpenStreetMap-Wiki[7] zu, wo idealerweise die gängigsten Erfassungsvorschläge dokumentiert sind. Auch zentral eingebundene Karten, wie der Mapnik-Stil auf www.openstreetmap.org sorgen nicht unerheblich für eine Vereinheitlichung im Datenschema. Schließlich geben auch Anwendungen wie Taginfo[10] einen guten Anhaltspunkt über bereits etablierte Erfassungsmöglichkeiten.

### 1.2.2 Menschlicher Faktor und inkrementelle Datenerfassung

Anders als in der Softwareentwicklung spielt bei einem Crowdsourcingprojekt wie OpenStreetMap der menschliche Faktor eine bedeutende Rolle[13]. Bei Open Source Softwareprojekten besteht ein wesentliches Korrektiv darin, ob die Software funktioniert oder darin, ob eine realisierte Idee besonders gut ein Problem löst etc. Diese unmittelbare Rückkopplung fehlt bei OpenStreetMap als geographischer Datenbank. Da nicht klar ist, zu welchem konkreten Zweck die Informationen gesammelt werden, fehlt es an einer so unmittelbaren Rückbindung der Datenerfassung an den Nutzen, wie dies bei Software zwischen Programmierung und Ergebnis der Fall ist.

Einigkeit besteht bei OpenStreetMap darin, dass die Daten zur Darstellung einer Karte geeignet sein sollen. Auch dass die Möglichkeit zum Routing gegeben sein soll, ist mittlerweile Konsens. Doch bereits an dieser Stelle wird das Problem deutlich. So war es in der Anfangszeit keineswegs klar, dass die Straßen und Wege in der Datenbank über ein gemeinsames Node verbunden werden mussten, denn in der gerenderten Karte erschienen die Straßen und Wege verbunden, auch wenn sie es in der Datenbank tatsächlich nicht waren, sondern nur nahe beieinander oder unverbunden übereinander lagen.

Mit geographischen Daten lässt sich aber noch bedeutend mehr bewerkstelligen als routingfähige Karten. Je spezialisierter die Anwendungen werden, die mit den OpenStreetMap-Daten umgesetzt werden sollen, desto schwieriger ist der Umgang damit. OpenStreetMap-Daten weisen zwar unter Umständen eine größere Informationsvielfalt[12] auf, als dies die eingangs geschilderten amtlichen Karten/Daten tun. Auf der anderen Seite sind Daten bei OpenStreetMap mit einer großen Unschärfe in der Bedeutung versehen.

Unschärfen tun sich insbesondere unterhalb der Darstellungsmöglichkeiten etablierter Karten wie beispielsweise www.openstreetmap.com auf. Jenseits der Darstellung fehlt ein wesentliches Korrektiv beim Erfassen der Daten in der OpenStreetMap-Datenbank. Aber auch schon bei an sich noch in der Karte darstellbaren Informationen kommt es zu erheblichen Unschärfen. Eines der prominentesten Beispiele aus der OpenStreetMap-Welt ist die die Übertragung von Fuß- und Radwegen in das Datenschema von OpenStreetMap. Wie es zu diesen Problemen kommt, soll nachfolgend nähere erklärt werden.

## 2 Die Kunst, die Welt in Worte zu fassen

Neben dem bereits oben geschilderten Crowdsourcing bei OpenStreetMap gibt es noch eine weiteres Konzept, das weitreichende Folgen auf das Datenschema von OpenStreetMap hat. Es handelt sich dabei um die sog. „on the ground rule“. Diese Regel besagt im Wesentlichen, dass nur etwas erfasst werden soll, dass man im Feld auch tatsächlich als (geographische) Information vorgefunden hat. Eine der wichtigsten Ausnahme hiervon ist die Erfassung von postalischen und administrativen Grenzen, die im Folgenden aber nicht weiter betrachtet werden sollen.

Oben wurde bereits geschildert, dass es jedem Mitwirkendem bei OpenStreetMap grundsätzlich erlaubt ist, ein geographisches Objekt so einzutragen, wie er es für richtig hält. Außerdem entscheidet der Mitwirkende selbst, welche Informationen erfassenswert sind. Er entscheidet also über das „Was“ und „Wie“ der Eintragung selbst. Die Orientierung an der „on the ground rule“ führt dabei zu einigen bemerkenswerten Effekten, die bisher bei der Diskussion um das „Wie der ‚richtigen‘ Erfassung“ nicht

ausreichend berücksichtigt wurden. Die zahlreichen und oft ergebnislosen Diskussionen in Foren und Mailinglisten legen ein beredetes Zeugnis davon ab.

## 2.1 Erfassung durch die Mitwirkenden

Eine der wichtigsten Aufgaben, die ein Mitwirkender bei OpenStreetMap hat, ist es, seine Wahrnehmung, die er von einem geographischen Objekt hat, in ein Datenschema zu übertragen und in die OpenStreetMap-Datenbank einzutragen. Die „on the ground rule“ legt dem Mitwirkenden dabei nahe, sich so gut wie möglich an der Wirklichkeit zu orientieren. Dabei wird sich der Mitwirkende, wenn er mit dem vorhandenen Datenschema nicht gänzlich unzufrieden ist, an dem orientieren, was er als Datenschema bereits vorfindet. Das heißt, der Mitwirkende macht sich eine Vorstellung von der Welt und versucht, diese dann in ein Datenschema zu übertragen.

Auf der Auswertungsseite versucht ein menschgemachter Computeralgorithmus die in der OpenStreetMap-Datenbank erfassten Informationen auszuwerten und in eine Anwendung zu überführen. Der Auswertung liegt dann das Verständnis des Programmierers zugrunde.

Das verknüpfende Element zwischen vom Mitwirkenden erfasster Information und der vom Programmierer ausgewerteten Information ist die Definition, welche der Information zugrunde liegt. Hier liegt das Kernproblem von OpenStreetMap, denn oft sind gerade die Definitionen zu einzelnen geographischen Objekten im Wiki unklar, wage oder widersprüchlich. Werkzeuge wie Taginfo können hier auch keine Antwort geben, denn sie geben nur einen quantitativen Überblick. Eine Aussage zum eigentlichen Informationsgehalt, der sich hinter einem Datenschema verbirgt, wird von Taginfo nicht getroffen. Eine solche Aussage lässt sich dann ableiten, wenn man die quantitative Aussage mit der qualitativen Aussage der Wikidefinition in Bezug setzt. Damit gelangt man wieder an den Ausgangspunkt, weil die Definitionen im Wiki unklar, vage oder widersprüchlich sind und oft nur den kleinsten gemeinsamen Nenner widerspiegeln.

Dem OpenStreetMap-Wiki kann also oft nur die Aussage entnommen werden, dass sich hinter einer bestimmten `key=value` Kombination der Datenbank die Bedeutung b1, b2 oder b3 verbirgt. Hierzu sei auf das Beispiel von `highway=primary` verwiesen, für die das OpenStreetMap-Wiki zumindest zwei Varianten kennt – „Hauptverbindungsstraße“ oder „Straße mit übergeordneter Verkehrsbedeutung“[9].

## 2.2 Randscharfe und kernprägnante Begrifflichkeiten

Das sich hier zeigende Phänomen hat seine Ursache in der Methodik, mit der OpenStreetMap bzw. seine Mitwirkenden die Welt erfassen. Mit der „on the ground rule“ orientieren sich die Mitwirkenden am „natürlichen“ Sprachgebrauch und ordnen entsprechend ein geographisches Objekt in das Datenschema ein. Dabei ist aber die eigene Vorstellung des Mitwirkenden bei der Einordnung von entscheidender Bedeutung. Diese kann aber gerade in Grenzfällen deutlich von der Vorstellung anderer Mitwirkender abweichen, ohne dass diese Diskrepanz im Datenschema zu Tage tritt. Sie zeigt sich erst beim Streit um die Definition bzw. Bedeutung.

Folgendes Beispiel soll das Problem veranschaulichen: Ein Mitwirkender möchte eine auch für die Öffentlichkeit zugängliche Betriebskantine in die Datenbank eintragen. Er entscheidet sich dafür, kein eigenes Schema zu entwickeln, sondern auf bestehende Tags zurückzugreifen. Er wird bei seiner Suche auf `amenity=restaurant` und `amenity=fast_food` stoßen. Unterstellt, es gibt keine Definition für Betriebskantine im Wiki oder der Mitwirkende findet die entsprechende Definition nicht, so ist er bei der Einordnung darauf angewiesen, welche Bedeutung er selbst dem Begriff „Betriebskantine“ zuweist.

Bei amtlichen geographischen Informationen besitzt ein geographisches Objekt eine entsprechende feste Definition und kann so gut in einem Datenschema erfasst werden. Bei OpenStreetMap ist es faktisch genau umgekehrt. Es gibt ein mehr oder weniger etabliertes Datenschema, um dessen Definition im Nachhinein gerungen wird.

Auf der Auswertungsseite wird hingegen auch bei OpenStreetMap von einem genau abgrenzbaren und feststehendem Bedeutungsgehalt ausgegangenen bzw. ein solcher in der Regel erwartet, wie dies bei Behördendaten der Fall ist. Vom unerfahrenen OpenStreetMap-Nutzer wird ein eindeutiges Datenschema mit feststehenden Bedeutungen erwartet. Dass dies oft nicht zutrifft, wird erst bei näherer Auseinandersetzung mit den Diskussionen rund um die „richtige“ Erfassung einzelner geographischer Objekte in Foren oder auf Mailinglisten deutlich.

### 2.2.1 Kernprägnante Begriffsbildung

Diesem bei OpenStreetMap existierenden Phänomen lässt sich gut mit der in der Sprachwissenschaft verwendeten Unterscheidung zwischen der kernprägnanten und der randscharfen Bedeutung von Begriffen beschreiben. Von Kernprägnanz wird gesprochen, wenn sich der Sprecher hinsichtlich der Bedeutung eines Wortes an einem prototypischen Bedeutungskern orientiert[11, S. 5], also am alltäglichen „natürlichen“ Wortsinn. Der genaue Umfang der Bedeutung hat für den Verwender wenig Interesse[11, S. 5].

Für eine erste Veranschaulichung mag das Wort „Katze“ dienen[11, S. 5]. Bei Verwendung dieses Wortes, wird im alltäglichen Sprachgebrauch prototypisch davon ausgegangen, dass es sich um eine Hauskatze handele. Selbst das Geschlecht des Tieres spielt bei dieser Aussage keine Rolle. Es ist für das alltägliche Gespräch auch nicht wichtig, ob neben der Hauskatze auch Tiger oder Löwen Katzen sind[11, S. 5]. Der Sprachgebrauch und die Kommunikation leidet nicht unter dieser Unschärfe[11, S. 5].

Für das oben angeführte Beispiel, bei dem sich der Mitwirkende zwischen amenity=restaurant und amenity=fast_food entscheiden muss, bedeutet es Folgendes: Eine Betriebskantine passt weder in den begrifflichen Kernbereich eines klassischen Restaurants noch in jenen eines Fastfoodrestaurants. Er ist also gezwungen, sich zu überlegen, in welchen Randbereich sich sein Begriff „Betriebskantine“ besser einfügt.

Versteht er unter einem Restaurant eine Einrichtung, in der von Porzellantellern mit Besteck gegessen wird oder nimmt er ein klassisches Menü, zusammengesetzt aus Fleisch, Sättigungsbeilage und Gemüse in den Blick, dann wird er die Betriebskantine für sich als amenity=restaurant definieren. Ein Fastfoodrestaurant wäre dann etwas, wo man mit Händen aus Einwegverpackungen isst.

Stellt sich der Betreffende unter Restaurant demgegenüber etwas vor, wo man viel Zeit in nettem Ambiente verbringt, weil einem die Speisekarte an den Tisch gebracht wird und das Essen erst nach der Bestellung zubereitet wird, so wird er die Betriebskantine nicht als Restaurant einordnen. Vielmehr wäre eine Betriebskantine so etwas wie ein Fastfoodresturant (amenity=fast_food), denn man bekommt sein Essen unmittelbar mit der Bestellung und verbringt nicht mehr Zeit in der Einrichtung, als zur Essensaufnahme unbedingt notwendig.

Je nach der Vorstellungswelt lässt sich also das eine oder andere vertreten und darüber entsprechend in einschlägigen Foren und Mailinglisten streiten.

Ähnlich verhält es sich im alltäglichen Sprachgebrauch mit den Begriffen Fuß- oder Radweg. Jeder weiß, was gemeint ist. Wie der Fuß- oder Radweg ausgebaut, ob er durch Verkehrszeichen gekennzeichnet oder baulich getrennt von der Straße für Kraftfahrzeuge verläuft, spielt im alltäglichen Gespräch keine entscheidende Rolle. Für die aufgabenorientierte Auswertung einer Datenbank macht es aber einen großen Unterschied. Je genauer etwas definiert ist, desto vielseitiger sind die Anwendungs- und Auswertungsmöglichkeiten. Ziel sollte es unter diesem Aspekt sein, sich von Sammelbegriffen, die alles oder nichts bedeuten können, wegzuentwickeln und einzelne Merkmale oder Kategorien getrennt zu erfassen.

So lassen sich Ausbauzustand, Zugangsregeln und Beschilderung nach StVO bei Wegen sehr gut als eigenständige Merkmale erfassen. Eine vorangehende Einordnung als Fuß- oder Radweg (und der Streit darüber) ist also vermeidbar, wenn man nur geschickt Kategorien bildet (dazu später mehr).

Das macht letztendlich auch Definitionen einfacher, denn je weniger ein einzelner Begriff schon in der Alltagssprache bedeutet, desto kleiner ist auch der Begriffshof, der sich um die Kernbedeutung bildet. Eine Einigung auf eine feststehende Definition ist dann oft leichter möglich. Es bleibt weniger Raum für abweichende Vorstellungen.

### 2.2.2 Randscharfe Begriffsbildung

Mit einer bei der Umgangssprache (als Sprache mit kernprägnanter Bedeutung) auftretenden und ohne weiteres hinzunehmenden Unschärfe kann die Wissenschafts- und Fachsprache nicht leben. Sie benötigt eine randscharfe Bedeutung. Dies wird durch einen ausdrücklichen Benennungsakt sichergestellt (Definition), hierdurch wird klargestellt, wo die Bedeutungsgrenze eines Wortes liegen soll[11, S. 5 f.]. Halten sich die entsprechenden Kreise an die Definition, so ist aus einem kernprägnanten Wort ein randscharfes geworden[11, S. 6].

Auch die Geoinformatik, deren Gegenstand die Verarbeitung geographischer Daten ist, ist als Wissenschaft auf die Verwendung randscharfer Bedeutungen angewiesen. Ist dies, wie beispielsweise bei OpenStreetMap-Daten nicht gewährleistet, so reduziert sich der Nutzen der vorhandenen Daten lediglich auf Bereiche, in denen es im Wesentlichen auf die Kernbedeutung ankommt (z. B. einfacher Stadtplan), aber nicht auf die genauen Bedeutungsgrenzen. Damit wird vielen Verwendungen, die denkbar wären, wenn hinter dem in der Datenbank verwendeten Datenschemata randscharfe Begriffe stehen würden, von vornherein die Grundlage entzogen. Folgt man dann dem oben gemachten Vorschlag, komplexe Begriffe wie Fuß- oder Radweg in einzelne Kriterien aufzuspalten, so besteht die Chance zu schneller Einigung. Bei der Auffindung sinnvoller Einzelkriterien, die als Datenschema zur Beschreibung komplexer Gebilde verwendet werden können, kann die Logik einen wesentlichen Beitrag leisten.

## 2.3 Zwischenergebnis

Aufgrund der Unzulänglichkeiten bei den Definitionen als Folge der Orientierung an der kernprägnanten Gemeinsprache bei der Einordnung von geographischen Objekten in das Datenschema der OpenStreetMap-Datenbank weisen die OpenStreetMap-Daten nur einen begrenzten Nutzen auf. Dieser Nutzen bleibt deutlich hinter den Möglichkeiten zurück, die bei eindeutig randscharfer Begriffsverwendung im Datenschema möglich wären.

Zukünftig müssen Möglichkeiten entwickelt werden, die unter Berücksichtigung des Crowdsourcingcharakters von OpenStreetMap zu einer Konsolidierung der verwendeten Definitionen, also zu randscharfer Begriffsbildung, führen.

Ein erster Schritt dorthin sollte sein, dass sich die Mitwirkenden bei der Diskussion um Begriffsbedeutungen zunächst einmal vergegenwärtigen, ob sie sich gerade im Bereich einer kernprägnanten oder randscharfen Wortbedeutung bewegen.

Derartige sprachliche Einordnungsschwierigkeiten würden sich auch gut über die von Hartmann initiierte Umfrageplattform[2] visualisieren lassen. Anders als über Taginfo können hier auch die Auffassungen, die sich hinter dem `key=value` Paar verbergen, abgefragt und visualisiert werden. Durch gezieltes und abgestuftes Nachfragen könnten sich auch die von den einzelnen Mappern gewählten Kriterien gut identifizieren und visualisieren lassen. Die so gewonnenen Erkenntnisse könnten in verbesserte Definitionen münden und führen über diesen Weg auch zu einer Konsolidierung des Datenschemas. Die bisher im Wesentliche auf textlichen Austausch angewiesene Meinungsbildung könnte auf diese Weise zudem effektiver und übersichtlicher gestaltet werden.

Hat man einmal anhand von Umfragen strittige Merkmale in Definitionen identifiziert, kann ein weiteres Mittel zur Verbesserung der Begriffs- bzw. Definitionsbildung und damit letztlich des Datenschemas

die stärkere Betonung der Regeln der Logik sein. Dabei ist die Definition kein Allheilmittel, sondern ist in der Regel auch das Ergebnis des Versuchs einer systematischen Ordnung[11, S. 5].

## 3 Die Kunst, die Welt zu ordnen

Das oben beschriebene Bedürfnis nach feststehenden Definitionen ist eng mit den Anforderungen der Logik an eine Definition verbunden. Die Forderung nach der Verwendung von randscharfen Begrifflichkeiten findet sich auch im Begriff der Nominaldefinition wieder, die einen Begriff für die Wissenschaftssprache festlegt.

### 3.1 Definitions- und Kategoriebildung

Hier kann man auf die Prädikabililienlehre, wie sie schon Aristoteles verwendet hat, zurückgreifen. Danach gibt es fünf Prädikabilien[5, S. 34]:

1. Gattung (genus proximum)
2. Art (species)
3. artbildender Unterschied oder Differenz (differentia specifica)
4. eigentümliche Eigenschaft (proprium)
5. zufällige Eigenschaft (accidens).

Nach der klassischen Definitionslehre besteht die Definition aus dem Gattungsbegriff (genus proximum) und dem artbildenden Unterschied (differentia specifica).

Die Definition eines Quadrates lautet: gleichseitiges Rechteck. Anhand der fünf Prädikabilien lässt es sich wie folgt beschreiben:

1. Gattung: Rechteck
2. Art: Quadrat
3. artbildender Unterschied: gleichseitig

Damit sind die Merkmale zusammengefasst, die ein Quadrat ausmachen. Weitere eigentümliche oder zufällige Eigenschaften können hinzukommen. Ein eigentümliches Merkmal ist kein notwendiges, wesensbildenden Merkmal, aber typisch für die beschriebene Art, z.B. dass sich die Diagonalen im Quadrat halbieren und einen rechten Winkel zueinander bilden.

Mit dieser Art der Definition lassen sich zwar nur Eigenschaften und keine Beziehungen beschreiben[5, S. 28 f.], für ein Projekt, welches sich die Erfassung raumbezogener Sachinformationen zum Ziel gesetzt hat, erscheint diese Art des Definierens zunächst ausreichend. In welcher Beziehung Sachinformationen zueinander stehen, lässt sich in der Datenbank beispielsweise mit einer Relation abbilden.

Zudem sind auch noch die eigentümlichen und die zufälligen Eigenschaften neben der eigentlichen Definition verfügbar. Diese können mit zusätzlichen `key=value` Kombinationen an einem Datenbankobjekt eingetragen werden.

### 3.2 Kategoriebildung und OpenStreetMap

Unter dem Gesichtspunkt der Definitionsbildung zur eindeutigen Beschreibung geographischer Objekte macht auf den ersten Blick die key=value Paarung wenig Sinn. Die Beschreibung der Straßenbedeutung in der Datenbank ist eindeutig unabhängig davon, ob man als Attribut für das Objekt

`highway=primary` oder `primary`

eintragen würde.

Unterstellt man aber, dass `primary` grundsätzlich unterschiedliche geographische Objekte beschreiben kann, so erscheint es notwendig, den Begriff `primary` einer Gattung zuzuordnen, um den Begriff in der Datenbank nicht für mögliche andere Definitionen „zu verbrennen“. Diese Funktion der Gattungsbeschreibung übernimmt der key. Das `key=value` der Form `highway=primary` beschreibt also einen bestimmten Typ der Gattung „Straßen und Wege“[8]. Der Schlüssel (key) `highway` steht für die Gattung (genus proximum) und der Wert (value) `primary` steht für den artbildenden Unterschied (differentia specifica). Gemeinsam wird daraus die Definition für eine „Hauptverbindungsstraße“ oder ein „Straße mit übergeordneter Verkehrsbedeutung“[9]. Anders als bei `highway=value` ist bei anderen Gattungen aber bereits das Datenbankschema selbst höchst inkonsistent, wie beispielsweise das weite Feld von `amenity=value` zeigt.

Dies macht zwei voneinander zu unterscheidende Probleme deutlich. Einmal systematisch definitorische Probleme im Datenbankschema selbst, zum anderen definitorische Probleme der durch das Datenbankschema repräsentierten Begriffe, also die Definition der geographischen Objekte selbst.

### 3.2.1 Definitorische Probleme des Datenbankschemas

Wenn man das Datenschema in den Blick nimmt, so stellt sich die Frage, ob highway=primary, highway=path und highway=living_street tatsächlich zur gleichen Gattung gehören. Nach der Definition im OpenStreetMap-Wiki ist es der Fall, weil `highway` als „Straßen und Wege“[8] definiert wird. Betrachtet man aber die (vermeintliche) Gattung `amenity`, so ist es fast unmöglich zu definieren, was das verbindende Element sein soll. Nach der eben eingeführten Definition müsste ein Merkmal (differentia specifica) ausreichen, um eine Unterscheidung herbeizuführen. Dies ist mitnichten der Fall. Die `key=value`-Paarung bei amenity=value wird nicht im Sinne einer Definition bestehend aus Gattungsbegriff und artbildenden Unterschied verwendet. Bei allen `amenity=value` Verwendungen handelt es sich faktisch um ein loses Sammelsurium ohne erkennbaren gemeinsamen Gattungsbegriff.

### 3.2.2 Definitorische Probleme geographischer Objekte

Auf Basis eines einigermaßen konsistenten Datenbankschemas lassen sich definitorische Schwierigkeiten meistens gut identifizieren. Dies zeigt, dass oben bereits eingeführte Beispiel von `highway=primary`. Hier dreht sich der Streit im Wesentlichen nur noch darum, welches das (entscheidende) artbildende Unterscheidungsmerkmal (differentia specifica) ist und welche Merkmale lediglich eigentümliche Eigenschaft (proprium) beziehungsweise zufällige Eigenschaft (accidens) sind.

Beim Streit um die Definition von `highway=primary` als „Hauptverbindungsstraße“ oder „Straße mit übergeordneter Verkehrsbedeutung“[9] dreht sich der Streit darum, ob die Anzahl der (Kraft-) Fahrzeuge pro Tag artbildende Unterscheidungsmerkmal ist oder aber die Einordnung der Straße durch eine wie auch immer geartete Institution (beispielsweise eine Behörde) als Hauptverbindungsstraße. Welche Einordnung man wählt, ist in gewisser Weise beliebig, beide Kriterien sind tauglich um highway=motorway, primary, secondary und tertiary voneinander abzugrenzen. Hier würde letztlich eine Entscheidung per Umfrage Klarheit bringen. Sollte sich die Mehrheit für „Verkehrsbedeutung“ als artbildendes Unterscheidungsmerkmal entscheiden, so könnte die behördlich gewollte Verkehrsbedeutung als eigentümliche Eigenschaft oder zufällige Eigenschaft über ein zusätzlich zu bildendes (und noch zu schaffendes) Attribut in die Datenbank eintragen lassen.

Bei `highway=unclassified`, `track` und `residential` ist sowohl das Merkmal „Verkehrsbedeutung“ als auch „behördliche Einordnung“ problematisch. Im Bereich der Wege highway=footway, cycleway, steps, living_street und path versagt es ganz. Hier sind plötzlich völlig neue artbildende Unterscheidungsmerkmal relevant. An dieser Stelle geht die Unklarheit über artbildende Unterscheidungsmerkmale beim geographischen Objekt nahtlos in den Streit über die Abbildung im Datenschema über. Erinnert sei hier nur an die gesamte Problematik um Fuß- und Radwege – und ihre rechtlichen (StVO) und/oder tatsächlichen (Ausbauzustand) Kriterien. An dieser Stelle schließt sich auch der Kreis zum Ausgangspunkt, dem Problem von Kernprägnanz und Randschärfe.

## 4 Ergebnis

Das Datenschema und die Definitionen geographischer Objekte sind bei OpenStreetMap von Unklarheiten und Inkonsistenzen geprägt. Diese machen es den etablierten Mitwirkenden schwer, die Datenstruktur von OpenStreetMap im Sinne von präzisen und klaren Beschreibungen geographischer Objekte weiterzuentwickeln. Die Anfänger stehen vor einer riesigen Menge unlogischer Tags, die den Einstieg erschweren, zahlreiche Fallen und Stolpersteine bieten und so für reichlich Frust und Enttäuschung sorgen.

Kurzfristige Lösungen und Patentrezepte sind im Rahmen eines communitybasierten Projekts nicht möglich. Die Vergegenwärtigung der Probleme von kernprägnanter und randscharfer Begriffsbildung sowie ein Mindestmaß an logisch-definitorischer Reflexion ermöglichen es, Probleme zu erkennen und Lösungen zu entwickeln. Insbesondere wenn diese Prozesse durch Umfrage und Visualisierungswerkzeuge unterstützt werden.

Kontakt zum Autor:

Falk Zscheile
August-Bebel-Straße 4
09224 Grüna
falk.zscheile@gmail.com

Literatur

1] *Bundesamt für Seeschiffahrt und Hydrographie*, Hrsg. Internationale Kartenserie Karte 1 (INT 1). Zeichen, Abkürzungen, Begriffe in deutschen Seekarten. Hamburg, Rostock: Bundesamt für Seeschiffahrt und Hydrographie, 2005.

[2] *Harald Hartmann*. Umfrageplattform. 2015. url: osm.haraldhartmann.de/umfrage/ (besucht am 12. 02. 2015).

[3] *Christian Horn, Peter Forbrig* und *Immo O. Kerner*, Hrsg. Lehr- und Übungsbuch der Informatik. Grundlagen und Überblick. 3. Aufl. Bd. 1. Leipzig: Fachbuchverlag Leipzig, 2003.

[4] *Jeff Howe*. The Rise of Crowdsourcing. 2006. url: http://www.wired.-com/wired/archive/14.06/crowds.html (besucht am 08. 07. 2012).

[5] *Albert Keller*. Allgemeine Erkenntnistheorie. 2. Aufl. Stuttgart, Berlin, Köln: Verlag W. Kohlhammer, 1990.

[6] *Landesvermessungsamt Nordrhein-Westfahlen*, Hrsg. Musterblatt für die Topographische Landkarte 1:25000. 3. Aufl. Bad Godesberg, 1993.
url: http://www.bezreg- koeln.nrw.de/brk_internet/geobasis/sonstige/topographische_karten_aeltere/musterblatt.pdf.

[7] *OpenStreetMap-Wiki*. OpenStreetMap Wiki. 10. Juli 2014. url: http://wiki.openstreetmap.org/wiki/Main_Page (besucht am 08. 02. 2015).

[8] *OpenStreetMap-Wiki*. De:key:highway. 22. Jan. 2015. url: http://wiki.openstreetmap.org/wiki/DE:Key:highway (besucht am 12. 02. 2015).

[9] *OpenStreetMap-Wiki*. De:Tag:highway=primary. 13. Sep. 2014. url: http://wiki.openstreetmap.org/wiki/DE:Tag:highway=primary (besucht am 12. 02. 2015).

[10] *Jochen Topf*. Taginfo. url: https://taginfo.openstreetmap.org/ (besucht am 08. 02. 2015).

[11] *Harald Weinrich*. „Formen der Wissenschaftssprache“. In: Wissenschaftssprache und Sprachkultur. Hrsg. von Weinrich u. a. Tutzinger Materialien 61. Tutzing: Evangelische Akademie, 1989, S. 3–21.

[12] *Dennis Zielstra* und *Alexander Zipf.* A Comparative Study of Proprietary Geodata and Volunteered Geographic Information for Germany. 2010. url: http://www. agile-online.org/Conference_Paper/CDs/agile_2010/ShortPapers_PDF/ 142_DOC.pdf (besucht am 13. 02. 2015).

[13] *Falk Zscheile*. „Open Street Map in Freiheit erstarrt? Gedanken zur Fortentwicklung durch die Community“. In: FOSSGIS Konferenz 2013. Hrsg. von FOSSGIS e. V. Rapperswil, 2013, S. 91–96.

# Ein Duett: OpenLayers 3 im Zusammenspiel mit AngularJS

Dominik Helle & Kai Culemann

Durch die Kombination von AngularJS im Zusammenspiel mit OpenLayers3 ist es möglich, mit überschaubarem Aufwand, komplexe Kartenanwendungen zu erstellen.

AngularJS ist ein Open-Source-Framework mit der dynamische Webanwendungen nach dem MVC-Prinzip erstellt werden können. Eine besondere Stärke von AngularJS liegt hierbei in der Unterstützung von bidirektionalem Databinding. Das bedeutet, dass Benutzereingaben direkt Auswirkungen auf die Webanwendung haben, ohne dass eine manuelle Synchronisation zwischen Ein- und Ausgabe notwendig ist. Anwendungen können daher einfache als Single-Page-App entwickelt werden.

Entwickelt wurde das Javascript-Framework von Google. AngularJS wurde im Jahre 2009 als Open-Source-Software veröffentlicht und wird mittlerweile von einer aktiven Open-Source-Community weiterentwickelt.

OpenLayers ist wohl eine der bekanntesten Javascript-Bibliothek um moderne Kartenanwendungen im Web zu erstellen. „OpenLayers 3“ ist der Nachfolger der weiterverbreiteten Bibliothek „OpenLayers 2“ und steht seit Ende August 2014 in der Version 3.0 zur Verfügung.

Die Kombination der beiden Bibliotheken ermöglicht es neue Funktionen zu Anwendungen hinzuzufügen und so das beste aus beiden Welten zu verknüpfen. Durch das Zusammenspiel kann zum Beispiel eine perfekt ans Layout abgestimmten Layerauswahl integriert werden. Neben graphischen Elementen können aber auch zusätzliche komplexe Kartenfunktionen hinzugefügt werden.

In dem Projekt „Offene Regionalkarte Mecklenburg-Vorpommern (ORKa.MV)“, wurden beide Komponenten kombiniert eingesetzt und als Basis für die Entwicklung einer Web-Kartenanwendung verwendet. Die hierbei entstandenen Komponenten wurden als Open-Source-Software veröffentlicht.

Bereits während der Entwicklung lag ein besonderes Augenmerk auf der möglichen Wiederverwendbarkeit der im Projekt entstanden Softwarekomponenten. Aus diesem Grund wurden die Entwicklung in zwei Projekte aufgeteilt.

Unter dem Arbeitstitel AnOl [1] wurden die Basisbibliothek bereitgestellt. Projektspezifische Änderungen wurden durch Anpassung und Erweiterung der AnOl-Bibliothek in einem extra angelegten Projekt [2] durchgeführt und veröffentlicht.

Für die nahtlose Integration in das Layout von Webanwendungen wurden HTML-Templates verwendet. Diese ermöglichen eine schnelle und einfache Anpassung an das gewünschte Layout, ohne dass der eigentliche Quellcode angepasst werden muss. Zudem wird die  Lesbarkeit des Quellcodes gefördert.

Durch die Verwendung von AngularJS ist es möglich, dass die in der Anwendung benötigten Elemente als HTML-Tags definiert werden können.

Beispiel-Code-Integration: Erstellen einer Karte mit Layerauswahl:

```
<div anol-map>
   <div anol-layerswitcher></div>
</div>
```

Im Folgenden werden ausgewählte Komponenten, die während des Projektes entstanden sind, kurz erläutert.

### Kartenauswahl

Eine typische Funktion einer Kartenanwendung ist die Kartenauswahl. Diese wird benötigt um zwischen verschiedenen Hintergrundkarten hin und her zu wechseln oder um bestimmte Point of Interests (POIs) ein- und ausblenden zu können.

Der Einsatz einer Kartenauswahl ist auch immer abhängig vom Layout und den hierbei eingesetzten Komponenten. Als Kartenbibliothek bietet OpenLayers in der Version 3.0 standardmässig keine eigene mitgelieferte Kartenauswahl.

AnOl stellt diese Möglichkeit zur Verfügung und arbeitet hierbei auch als Vermittler zwischen Oberfläche und der Kartenanwendung. Über ein HTML-Template kann die Kartenauswahl an das gewünschte Layout angepasst werden.

Die Verwendung der Kartenauswahl ist denkbar einfach. Es muss einfach der entsprechende HTML-Code-Snippet zur Anwendung hinzugefügt werden:

Beispiel-Code-Integration:

```
<div anol-layerswitcher></div>
```

### Permalink

Will man einem Kollegen oder einem Freund einen bestimmten Kartenausschnitt zeigen ist es oft mühsam ihn genau zu dem gleichen Ausschnitt zu navigieren. Damit der Austausch von Kartenausschnitten vereinfacht wird, haben sich so genannte Permalinks etabliert. Diese Links enthalten alle notwendigen Informationen um eine bestimmte Karte an einem bestimmten Punkt zu laden.

AnOl stellt diese Funktionalität bereit und aktualisiert hierfür bei jeder Karteninteraktion die URL der Anwendung. So werden alle notwendigen Informationen übertragen, die dann beim Laden der Anwendung wieder ausgelesen werden. Ein Austausch von Kartenausschnitten ist somit schnell und einfach möglich.

Zusätzlich zur Karte wird auch das Zusammenspiel der weiteren Kartenkomponenten beim Laden eines Permalinks von AnOl übernommen. So wird zum Beispiel auch die Layerauswahl auf die aktuell aktive Karte umgestellt.

### Controller-Elemente

Um die Anwendung einheitlich konfigurieren zu können bietet AnOl zudem einige Wrapper für typische Kartenfunktionen von OpenLayers an. Über diese Wrapper lassen sich vorhandene Funktionen von OpenLayers einfach als HTML-Tags in die Anwendung integrieren. Dies erspart die sonst notwendige Konfiguration.

Beispiel-Code-Integration:

```
<div anol-mouse-position></div>
<div anol-scale-line></div>
```

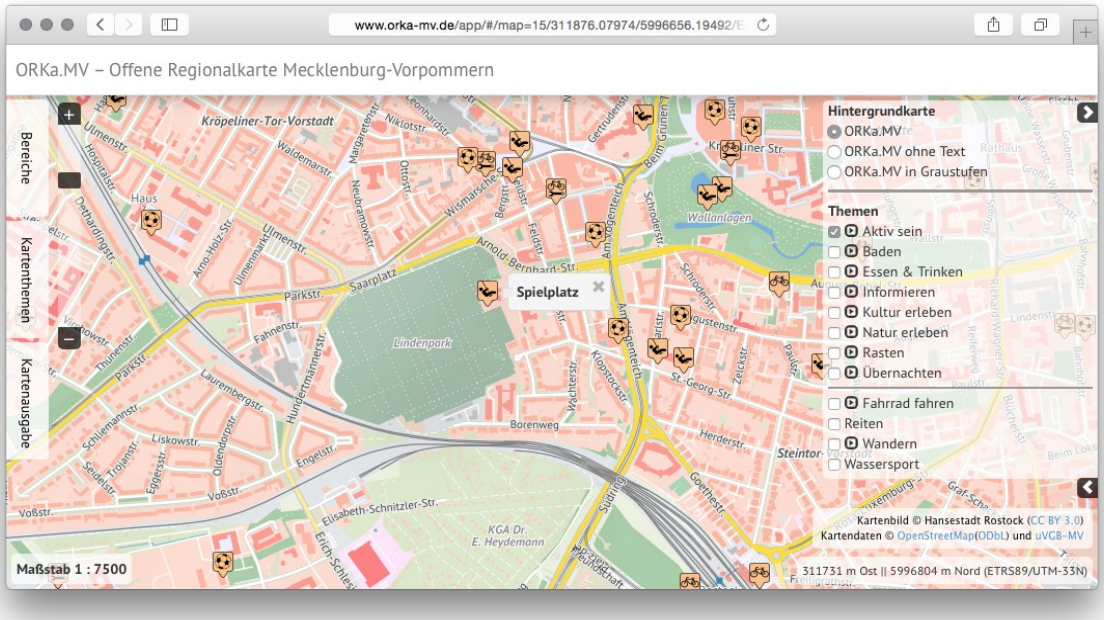



Abb 1: Screenshot – ORKa.MV: Anwendung mit AngularJS und OpenLayers 3

Die im ORKa.MV-Projekt erstellte Kartenanwendung verfügt noch über weitaus mehr Funktionen als die hier ausgewählten Komponenten.

So wurde in der Anwendung u.a. eine Abfrage von POIs oder auch das Erstellen eines Kartenexports realisiert. Die volle Funktionalität der Anwendung kann unter http://www.orka-mv.de/app aufgerufen und getestet werden.

Die im Projekt entwickelten Komponenten der Kartenanwendung wurden unter der MIT Lizenz veröffentlicht und stehen auf Github zum Download zur Verfügung.

Kontakt zu den Autoren:

Dominik Helle
Omniscale GmbH & Co. KG
Nadorster Straße 60
26123 Oldenburg
E-Mail: helle@omniscale.de
Homepage: http://omniscale.de

Kai Culemann
Omniscale GmbH & Co. KG
Nadorster Straße 60
26123 Oldenburg
E-Mail: culemann@omniscale.de
Homepage: http://omniscale.de

Literatur

[1] Anol: https://github.com/omniscale/anol

[2] Orka-App: https://github.com/omniscale/orka-app

[3] Offene Regionalkarte MV: http://www.orka-mv.de/app

[4] Angular JS: http://angularjs.org

[5] OpenLayers3: http://openlayers.org

# Zeitreihenanalyse mit GRASS GIS

## GRASS das temporale Open Source GIS

Sören Gebbert

Durch die Integration der Zeit als neue Dimension in GRASS GIS stehen nun über 45 Werkzeuge zur Zeitreihenalayse bereit. Eine Auswahl an Werkzeugen für die Analyse, Verarbeitung und Visualisierung von Raster- und Vektor-Zeitreihen wird vorgestellt.

Seit den 90iger Jahren wurde GRASS GIS für die Analyse von Satellitenbild-Zeitreihen und numerischen Modellen verwendet. Seit der GRASS GIS Version 7 steht nun jedoch eine neue Dimension von Werkzeugen zur Zeitreihenanalyse zur Verfügung.

GRASS GIS ist das erste Open Source GIS in das die Zeit als eine weiterer Dimension integriert wurde. Dabei wurde Wert auf Effizienz und parallele Verarbeitung von großen Zeitreihen gelegt. Dadurch ist GRASS GIS in der Lage Zeitreihen mit zehntausenden Raster, 3D Raster- oder Vektorlayern zu verarbeiten.

Es stehen über 45 Werkzeuge für Management, Analyse, Verarbeiten und Visualisierung von Raster-, 3D Raster und Vektorzeitreihen bereit. In diesem Vortrag wird eine Auswahl der verschiedenen Werkzeuge vorgestellt.

Diese umfassen die:

- Effiziente Handhabung von tausenden Raster- und Vektor-Layern in so genannten Space Time Datasets
- Kombinierte räumliche und zeitliche Abfragen mittels SQL und algebraischen Ausdrücken
- Zeitreihenverabreitung mittels einer Raum-Zeit Kartenalgebra
- Zeitliche Aggregation sowie Akkumulation von Rasterzeitreihen
- Visualisierung von verschiedenen Raster- und Vektorzeitreihen mittels Animation
- Darstellung von temporalen Beziehungen zwischen Zeitreihen

**Links**

- Wissenschaftlicher Artikel über Zeit in GRASS GIS:
  http://ifgi.uni-muenster.de/~epebe_01/tgrass.pdf
- Wissenschaftlicher Artikel über das GRASS GIS Temporal Framework (im Review Prozess):
  http://www.josis.org/index.php/josis/article/view/206

# OSM auf Rädern

Nicole Beringer

Sind die OSM Karten soweit, um den Wettbewerb mit kommerziellen Karten im Autonavigationsbereich aufzunehmen?

Die OSM Karten basieren auf dem direkten User Feedback und verbessern kontinuierlich die Kartenqualität. Die Software-Unternehmen aus dem automotiven Bereich verfolgen die OSM Entwicklung mit größer Aufmerksamkeit und stehen auf der Spitze der Implementierung dieser neuen Technologie. Elektrobit Automotive ist ein globaler Automobil Software-Unternehmen und bringt folgende Themen in die Diskussion:

- Die heutige Beurteilung der OSM Karten aus Sicht der professionellen Fahrzeugnavigationssysteme
- Potenzial von OSM Karten in der Automobilindustrie und mögliche Auswirkungen von dem Projekt auf die Navigationsindustrie
- Langfristigen Aussichten für die OSM in dem automotiven-Kontext

Außerhalb der Abstract Veröffentlichung im Web, für die FOSSGIS Komission: Elektrobit wird während des Vortrages zum ersten Mal eine praktische Implementierung von OSM Karten in einer neuen, kostenlosen Navigation App mit dem professionellen automotive- Kernel präsentieren.

# GeoExt

## Ein ExtJS basiertes Rich-Client-WebGIS-Framework in Zeiten von AngularJS und Konsorten

Christian Meier, Marc Janssen

Der Vortrag stellt die neueste Version von GeoExt vor und zeigt auf, welches Handwerkszeug dem Entwickler hier bereitsgestellt wird. Unterschiede zwischen ExtJS und anderen Bibliotheken werden benannt, dies kann als Diskussionsgrundlage für die Wahl einer Bibliothek dienen. Schwerpunkt ist die Betrachtung der zukünftigen Entwicklung von GeoExt.

GeoExt [1] ist eine auf den JavaScript-Bibliotheken OpenLayers (für interaktive Karten im Web und Verarbeitung einer Fülle von OGC-konformen Formaten, [2]) und ExtJS (Framework zur Erstellung von Desktop-ähnlichen Webanwendungen mit nativem Look and Feel, [3]) aufbauende OpenSource JavaScript-Bibliothek, die es vereinfacht, Kartenmaterial in ansprechenden und komplexen Oberflächen zu präsentieren, so genannte "Rich Webmapping Applications".

Neben ExtJS bietet der Markt eine Vielzahl weiterer JavaScript-Frameworks und Bibliotheken an, die sich ebenfalls der Herausforderung angenommen haben, die Entwicklung von webbasierten JavaScript Clients zu vereinfachen und zu harmonisieren. Hier sind -- und das ist nur eine willkürliche Auswahl -- etwa AngularJS ([4]) und EmberJS ([5]) zu nennen. Eben jene Frameworks sind derzeit in der Entwicklergemeinschaft sehr beliebt, es werden viele klare Vorzüge dieser modernen Frameworks gelobt und die Art und Weise der Problemlösung spricht viele Developer an.

Der Vortrag wird die neueste Version von GeoExt vorstellen und aufzeigen, welches Handwerkszeug dem Entwickler hier bereitsgestellt wird. Wir werden Unterschiede zwischen ExtJS und den vorgenannten Bibliotheken benennen und Diskussionsgrundlage für die Wahl einer Bibliothek geben. Hierbei können wir als Kernentwickler von GeoExt nie vollständig neutral vorgehen, wir wollen jedoch versuchen jeweilige Vor- und Nachteile der jeweiligen Bibliotheken herauszustellen.

Zum Zeitpunkt der Abstract-Einreichung wird an GeoExt massiv weiterentwickelt: Es stehen die Unterstützung von ExtJS 5 und (später) OpenLayers 3 an. Ein weiterer Schwerpunkt wird dementsprechend auf der Betrachtung dieser und der zukünftigen Entwicklung von GeoExt liegen.

### Links

- Version 4.2.1 [3]: http://www.sencha.com/products/extjs/
- Version 2.0.2 [1]: http://geoext.github.io/geoext2/
- Version 2.13.1 [2]: http://openlayers.org/two/
- [5]: http://emberjs.com/
- [4]: https://angularjs.org/

# Kreisbogen in QGIS

Stefan Ziegler

Neben den gängigen Vektorelementen (Punkt, Linie, Polygon) unterstützt QGIS neu Kreisbogengeometrien. Es ist jetzt möglich solche Geometrietypen (ST*CircularString, ST*CompoundCurve, ST_CurvePolygon etc.) aus einer Postgis-Datenbank zu lesen, anzuzeigen, zu editieren und wieder zu speichern.

Gängige Vektorelemente eines Geografischen Informationssystems sind Punkt, Linie und Polygon. In der amtlichen Vermessung (amtlicher Liegenschaftskataster) und anderen Themenkreisen ist zusätzlich der Kreisbogen zum Definieren und Verwalten von Objekten erlaubt. Proprietäre geografische Informationssysteme zur Erfassung und Verwaltung der amtlichen Vermessung erlauben, im Gegensatz zu den gängigen FOSSGIS Desktop Lösungen, den Umgang mit Kreisbogen. FOSSGIS Lösungen segmentieren die Kreisbogen beim Import in das Zielsystem. Dieser pragmatische Ansatz hat neben einigen Vorteilen aber leider den Nachteil, dass man das Wissen über die Lage und die Definition eines Kreisbogenelementes verliert.

Für die Verifikation verschiedener Datensätze wurden Entwicklungen im Kern von QGIS hinsichtlich der Verwaltung und Bearbeitung von Kreisbogen gemacht. Neu können Kreisbogengeometrien (ST*CircularString, ST*CompoundCurve, ST_CurvePolygon etc.) aus einer Postgis-Datenbank gelesen, in QGIS angezeigt und editiert werden und anschliessend wieder gespeichert werden. Dafür wurde die Geometrieklasse von QGIS neu entwickelt. Zusätzlich zu den Kreisbögen kann durch das Redesign korrekt mit M- und Z-Werten von Koordinaten umgegangen werden. Ebenfalls denkbar sind zukünftig weitere Geometrietypen wie Splines etc.

Neben den technischen Erweiterungen und den daraus resultierenden Herausforderungen wird gezeigt aus welchem Grund nicht alle Arbeitsschritte den Umgang mit Kreisbogen beherrschen. So können zum Beispiel die meisten Geometrieoperatieon (Verschnitte, Differenzen etc.) noch nicht mit Kreisbogengeometrien umgehen. Für diese Arbeitsschritte wird die Geometrie on-the-fly segmentiert.

Neben dem eigentlichen Auslöser - der Verifikation von Geodaten - darf auch der Vorteil bei der Erfassung nicht unterschätzt werden. So sind in verschiedenen Themenkreisen Daten viel schneller erfasst, wenn sie mittels Kreisbogen (drei Punkte) definiert werden können.

Aktuelle Entwicklungen in anderen Projekten (ogr2ogr, GeoServer) zeigen, dass die Thematik “Kreisbogen” endgültig in der Open Source GIS Welt angekommen ist.

# Automatisierte Aufbereitung und Analyse von LKW-Mautstrecken in Deutschland am Beispiel von OpenStreetMap-Daten

Robert Klemm

Zusammenfassung der Bachelorarbeit

**Hintergrund:**

In Europa werden verschiedene Mautsysteme verwendet, die sich in ihren Erfassungs- oder Abrechnungsmethoden unterscheiden. Diese Methoden werden in der Bachelorarbeit kurz aufgegriffen und mit den EU-Ländern Deutschland, Österreich und der Schweiz auf ihre Besonderheiten verglichen. Da die Berechnung der Maut in einigen EU-Staaten speziellen Randbedingungen unterliegt (streckenbezogene oder gewichtsbezogene Abgabe), müssen diese Aspekte auch im Schema der Datenbank berücksichtigt werden. Hinzu kommen zusätzliche Informationen, die für die Visualisierung mautpflichtiger Strecken wichtig sind, wie beispielsweise Informationen über die betroffenen Fahrzugtypen oder über den Betreiber der Strecke.

Mit der öffentlichen Debatte zur Ausweitung der LKW-Maut auf ausgewählten Bundesstraßen und zur Einführung einer PKW-Maut in Deutschland rücken Geodaten, speziell die Verkehrsdaten, wieder in den Fokus der Nutzer. Die Straßennutzung durch LKWs wird in Deutschland auf Grundlage des Mautgesetzes [1] des Bundes mit einer Abgabe, der LKW-Maut, belegt. Die Berechnung der Maut richtet sich in Deutschland nach mautpflichtigen Abschnitten und nicht nach der gesamten, gefahrenen Strecke. Derzeit ist es schwierig die Mautdaten mit den Verkehrsdaten zu vereinigen. Die Mautabschnitte besitzen nur eine BASt-interne Bezeichnung und sind nur über die räumliche Geometrie oder das lokale Wissen editierbar.

**Bachelorarbeit:**

Das in dieser Arbeit beschriebene Tool und die damit erzeugten Mautdaten erleichtern dem Nutzer (OSM-Mapper) die Arbeit. Für ihn validiert das Tool durch die Dijkstra -Routinganalyse die OSM-Daten, basierend auf dem Tagging-Schema mit mautbezogenen Routingparametern, wie Fahrzeug-, Achs-, Gewichtsklasse und Betreiber. Die erzeugten Mautdaten helfen auch Nutzern, die mit Hilfe von OSM die mautbezogenen Routingparameter interpretieren oder auswerten wollen. Beispielsweise lässt sich so die schnellste und günstigste Strecke, bezogen auf die zu entrichtende Straßenmaut, ermitteln und grafisch anzeigen.

Im Rahmen der Bachelorarbeit ging es um die Aggregierung von verkehrsbezogenen Daten, die von der Bundesanstalt für Straßenwesen (BASt) [2] und OpenStreetMap (OSM) zur Verfügung gestellt werden. Projektziel war es, ein Vorgehen zu entwickeln, das mautbezogene Verkehrsdaten ausliest und diese einheitlich, im OSM-Tagging-Format, in einer OSM-Datenbank ablegt. Dem OSM-Mapper soll eine WMS-Schnittstelle oder eine Korrekturliste zur Verfügung gestellt werden, die das Erfassen und Validieren von Mautdaten erleichtern sollen. Für den allgemeinen Nutzer lassen sich, mit Hilfe einer Applikation, die Daten als mautpflichtige Route anzeigen. Zudem sollen die Daten grafisch ansprechend in einer Lagekarte visualisiert werden. Für das zu entwickelnde System werden die ursprünglichen Daten der BASt und OSM nach PostgreSQL importiert. Danach werden die Daten mit verschiedenen Analysen und Verarbeitungsverfahren mit Hilfe der PostGis- und PgRouting-Bibilothek aufbereitet.

Das Gesamtprojekt besteht aus drei Elementen:

1. Erzeugen einer Korrekturliste und einen WMS-Dienst für den OSM-Mapper
2. einem Maut-Informationsportal

3. Maut-Routing für den LKW-Fahrer

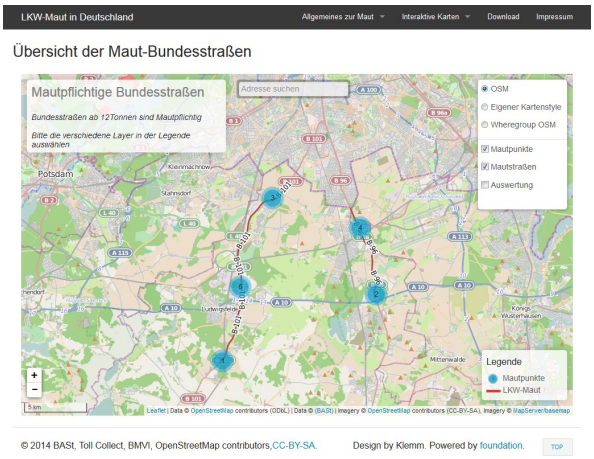



Abb. 1 : Maut-Informationsportal

## Maut-Informationsportal

In dieser Applikation werden die Geometrien auf eine Kartenbasis von OpenStreetMap gelegt, um eine Orientierung für Nutzer zu gewährleisten. Bildhafte Signaturen helfen dem Nutzer eine schnelle Übersicht über die wichtigsten Räumlichkeiten und die dazugehörigen Mautinformationen zu erhalten.

## Maut-Routing für den LKW-Fahrer

Die Routing-Applikation kann das Ergebnis der Routing-Abfrage individuell anhand der vorhandenen Daten darstellen. Als Basis für die Routing-Applikation wurde das OpenSource-Programm PgRouting [3] benutzt. Für die grafische Oberfläche dient Open Source Routing Machine (OSRM) [4], das im Rahmen des Projektes angepasst und weiterentwickelt wurde.

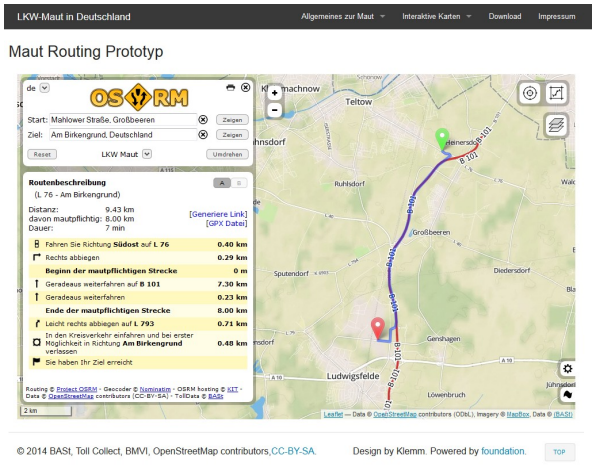



Abb. 2 : Maut-Routing für den LKW-Fahrer

## Technische Umsetzung:

Das gesamte Projekt wurde mit den OpenSource-Programmen auf einem Ubuntu-Server umgesetzt. Als Basis dient ein objektrelationales Datenbankmanagementsystem (ORDBMS) „PostgreSQL“ [5]. Die Datenbank wird mit der SQL-Datenbanksprache bedient und unterstützt die gängigen Datentypen, Operatoren, Funktionen und Aggregate. Die Erweiterungen PostGIS [6] und PgRouting unterstützen die räumlichen Daten in einer PostgreSQL-Datenbank. In PostGIS werden neben der Speicherung und Bearbeitung von räumlichen Daten auch Analysefunktionen benutzt. PgRouting bietet Funktionen für “Kürzeste-Wege”- Berechnung und andere netzorientierte Algorithmen an.

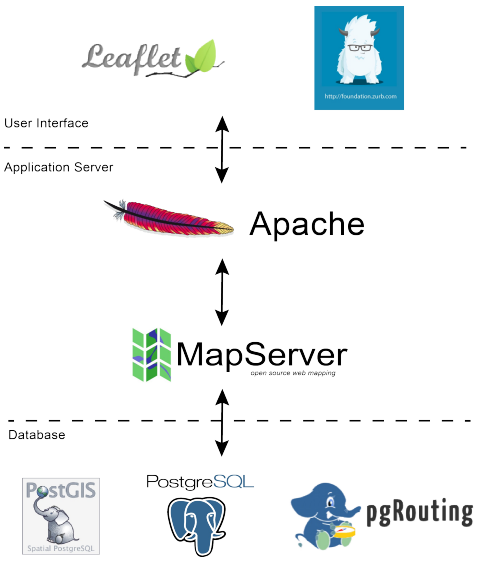



Abbildung 1: Technische Umsetzung

Ein weiteres wichtiges Element ist ein Applikation Server, wie Apache-Server. Der Apache-Server nimmt die Anfragen vom User-Interface entgegen und leitet sie an den MapServer weiter. Der verwendete MapServer [7] ist ein Karten-Server, der es ermöglicht, Geodienste gemäß den Spezifikationen des Open Geospatial Consortium (OGC) zur Verfügung zu stellen.

Das User-Interface besteht aus einem CSS -Framework und einer Leaflet-Applikation [8]. Unter Leaflet versteht man eine auf JavaScript basierende Bibliothek. Ziel ist es, die Geodaten mit einer grafische Benutzeroberfläche zu kombinieren, wie z. B. die Zoomfunktion oder die Schnittstellenabfrage Web Map Service (WMS). Die Foundation von der Firma Zurb [9] ist ein freies CSS-Framework, mit eigener JavaScript-Bibliothek zum Erstellen von Webseiten auf mobile- oder Desktop-Plattformen. Die Foundation unterstützt die gängigen HTML-, CSS-, CSS(Sass)-Format. Außerdem

**Zusammenfassung:**

In dieser Bachelorarbeit wurde ein Vorgehen entwickelt, bei dem mautbezogene Verkehrsdaten erfasst und umgewandelt werden, sodass sie in OSM abgelegt werden können. In diesem Zusammenhang wird eine neue Kartierungsvorschrift vorgestellt, die als Tagging-Schema für das OSM-Datenmodell dient.

Der Umgang mit dem Script hilft Nutzern, wenn sie die Überprüfung der Daten automatisieren wollen. Das Script erzeugt Vorlagen für das Maut-Tagging mit Hilfe eines Routing Algorithmus, der Eigenschaften wie Fahrzeug-, Achs- oder Gewichtsklasse und Betreiber berücksichtigt. Damit lässt sich zukünftig die schnellste und günstigste Route, bezogen auf die zu entrichtende Straßenmaut, erzeugen.

Das Vorgehen und das Script sind jetzt schon eine kostengünstige Alternative zu kommerziellen Lösungen. Die Ergebnisse haben eine gute Qualität, da sie auf Grundlage von aktuellen und weltweit erfassten Daten beruhen.

Eine Demoversion zur Bachelorarbeit kann man sich unter: http://maut.mapwebbing.eu/ anschauen.

Kontakt zum Autor:

Robert Klemm
robert.klemm1988@gmail.com

Literatur

[1] BMVI. (2015). Das Mautgesetz. (B. f. Infrastruktur, Hrsg.) Abgerufen am 05.02.2015 von http://www.bmvi.de/SharedDocs/DE/Artikel/UI/lkw-maut-gesetze-und-verordnungen.html

[2] BASt. (2015). Die Mauttabelle. (B. f. Straßenwesen, Hrsg.) Abgerufen am 05.02.2015 von http://mauttabelle.de/

[3] pgRouting. (2015). pgRouting Project. Abgerufen am 05.02.2015 von http://pgrouting.org/index.html

[4] Projekt OSRM. (2015). Open Source Routing Machine. Abgerufen am 05.02.2015 von http://project-osrm.org/

[5] PostgreSQL. (2015). PostgreSQL Project. Abgerufen am 05.02.2015 von http://www.postgresql.org/

[6] PostGIS Community. (2015). Handbuch PostGIS. Abgerufen am 05.02.2015 von http://postgis.net

[7] MapServer. (2015). MapServer Project. Abgerufen am 05.02.2015 von http://www.mapserver.org/

[8] Agafonkin, V. (2015). Leaflet. Abgerufen am 05.02.2015 von http://leafletjs.com/

[9] Zurb Foundation. (2015) Zurb Foundation (Framework). Abgerufen am 05.02.2015 von http://foundation.zurb.com/

# MapServer Pro-Tipps

Jörg Thomsen, MapMedia GmbH

MapServer ist bekannt für seine Zuverlässigkeit, Performance und Stabilität. Gerne wird aber auch über die Konfiguration mit den Mapfiles gestöhnt. Dabei bietet gerade diese Art der Konfiguration Möglichkeiten, die tief in der Dokumentation versteckt sind und die bei kniffligen Aufgaben wirklich hilfreich sein können. Die meisten Anwender kennen nur die Basis-Konfigurationen, aber wer weiß schon, dass es einen eingebauten OpenLayers-Client gibt, der es ermöglicht nur mit einem Mapfile eine interaktive Karte zu veröffentlichen, wie man Linien mit Pfeilspitzen versieht, Geometrien glättet oder den Zugriff auf MapServer-WMS direkt im Mapfile auf bestimmte IP-Adressen beschränken kann? Mit einfachen PHP- (oder anderen) Scripten eröffnen sich darüber hinaus vielfältige Möglichkeiten, den MapServer um weitere Funktionen zu erweitern oder sogar auszutricksen.

Die behandelten Themen haben sich größtenteils aus meiner praktischen Arbeit mit dem MapServer ergeben, dass heißt die hier gezeigten Tipps entstammen alle aus der eigenen Projektpraxis. Einige 'Tricks' stehen in der Dokumentation und werden regelmäßig in den Mailinglisten wiederholt, trotzdem werden sie nach meiner eigenen Erfahrung ignoriert, bis man selbst vor dem passenden Problem steht. Andere Tipps stammen aus den Tiefen der Entwickler-Mailinglisten und Request for Comments (RFC)[1] sowie dem msautotest[2]. Obwohl der Mapserver hervorragend dokumentiert ist, finden sich bei intensiverer Suche immer wieder Funktionen die Ihren Weg (noch) nicht in die Dokumentation gefunden haben. Die meisten Anforderungen, die uns über den Weg laufen sind nicht neu. Das ist zum einen schade, weil wir weniger innovativ sind, als wir annehmen, zum anderen ist das gut, weil es irgendwo bereits eine Lösung gibt. An dieser Stelle möchte ich gleich den ersten Stolperstein aus dem Weg räumen, bevor ich in medias res gehe: Vergesst die deutsche Dokumentation, die englische Versionen ist an vielen Stellen deutlich aktueller und zeigt Möglichkeiten auf, die in den letzten Monaten, wenn nicht Jahren, seit der Übersetzung hinzugekommen sind.

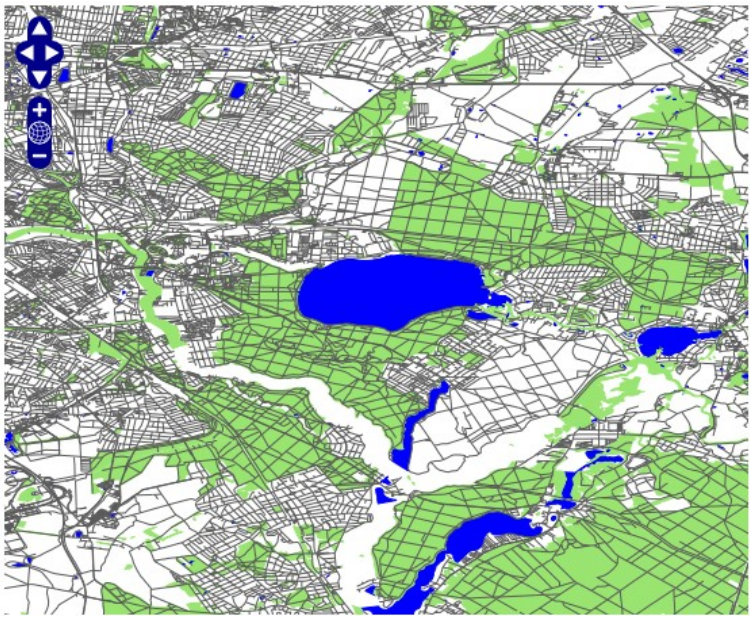
## Eingebauter Openlayers-Client

Man kennt das: Ein Mapfile ist soweit erstellt, dass man sehen möchte wie die Karte aussieht oder man möchte seinem Kunden zeigen, wie die Karte im Browser aussieht. Aber der Client ist nicht nicht konfiguriert, ein GIS als WMS-Client gerade nicht zur Hand und den WMS GetMap-Aufruf von Hand zusammenschreiben würde auch lange dauern bis Boundingbox und extent zusammen passen (von verschiedenen Zoomstufen ganz zu schweigen). Wir können hier auf den guten alten Browse-Mode des MapServers zurückgreifen, das Template müssen wir nicht selbst erstellen, wenn wir einfach OpenLayers als Template angeben:

```
http://www.osmct.de/cgi-bin/mapserv?map=/data/msprotipps/osm.map&mode=brow-
se&template=openLayers&layer=natural&layer=roads
```

liefert uns im Browser die Karte inklusive Mapping-Client:

1 http://mapserver.org/development/index.html
2 http://mapserver.org/development/tests/index.html und http://svn.osgeo.org/mapserver/trunk/msautotest/



Dabei sind ein paar kleine Dinge zu beachten:

- es wird der SIZE-Parameter im Mapfile benötigt, er bestimmt auch die Größe des OpenLayers-Clients im Browser
- der im Header gesetzte EXTEND bestimmt die Anfangs- sowie die maximale Ausdehnung der Karte
- mehrere Layer werden, wie immer im Browse-Mode, durch layer=x,layer=y, layer=z angegeben (nicht durch layers=x,y,z wie im WMS-Modus)
- der Clientrechner braucht eine Internetverbindung, weil er sich das OpenLayers-JavaScript vom MapServer-Server holen muss.

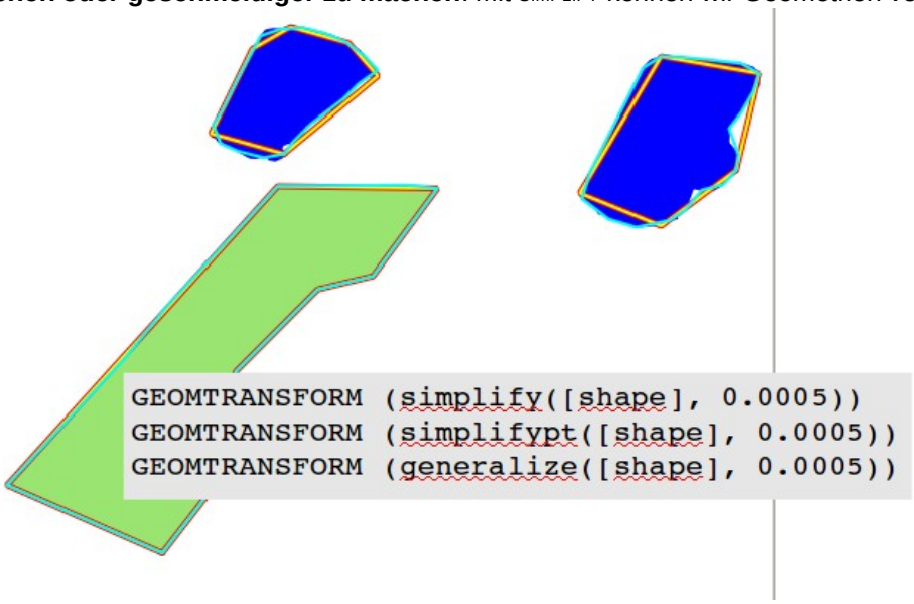
Das funktioniert auch mit WMS-Requests, hier muss der FORMAT-Parameter auf FORMAT=application/openlayers gesetzt werden. Und zu guter Letzt ist es sogar möglich ein eigenes OpanLayers-JavaScript zu benutzen, man muss das dem MapServer nur per Umgebungsvariable mitteilen. Weitere Informationen hierzu unter http://mapserver.org/cgi/openlayers.html.

## Schönere Karten dank Geomtransform

Wir haben nun einen funktionierenden Mapfile und können uns die Karte wie oben beschrieben auf einfache Weise im Browser ansehen und sogar in ihr navigieren. Dabei fallen womöglich Kleinigkeiten in der Kartendarstellung auf, die uns oder unserem Kunden nicht gefallen. Einige Beschriftungen sollen durch Umrahmungen hervorgehoben werden, die grobschlächtige Digitalisierung der Flüsse lässt diese zu eckig erscheinen und die Einbahnstraßen sollen ein 'Durchfahrt verboten' Schild an dem Ende erhalten, an dem man nicht in sie hinein fahren darf.
**Beginnen wir bei der Gestaltung der Label**. Bis zur Version 6 von MapServer hatten wir die Möglichkeit die Schlüssel BACKGROUNDCOLOR und BACKGROUNDSHADOWCOLOR zu benutzen, um einen Rahmen um eine Beschriftung zu zeichnen. Diese Methode ist nun seit einiger Zeit veraltet, die neue Methode bedeutet ein wenig mehr Aufwand, bietet aber bedeutend mehr Möglichkeiten bis hin zur Nutzung von Javascript in MapServer 7.
Wie erhalten wir also unsere Umrandung? Indem wir mit geomtransform ein Rechteck um unser Label zeichnen, das folgende Beispiel zeichnet eine Box mit rotem Hintergrund und grauem Schatten um unsere Beschriftung:

```
LABEL
   STYLE
      GEOMTRANSFORM 'labelpoly'
      COLOR 153 153 153
      OFFSET 3 2
   END
   STYLE
      GEOMTRANSFORM 'labelpoly'
      COLOR 255 0 0
   END
END
```


Der Wert 'labelpoly' ist dabei ein festgelegter Begriff, es wird immer ein Rechteck gezeichnet, das genau das Label umschließt. Die Ausgestaltung des Rechtecks erfolgt in einem Style-Block, wir haben also zur Gestaltung alle Möglichkeiten, die uns üblicher Weise in einem Style-Block zur Verfügung stehen und da das deutlich mehr ist als nur die Farbe anzugeben, hatte ich gewagt zu behaupten, dass uns mit etwas mehr Aufwand deutlich mehr Möglichkeiten zur Verfügung stehen.
Aber geomtransform wurde nicht nur eingeführt, um die Gestaltung der Label aufwendiger zu machen, es lässt sich auch auf alle anderen Geometrien anwenden und bietet z.B. die Möglichkeit **Geometrien zu vereinfachen oder geschmeidiger zu machen**. Mit SIMPLIFY können wir Geometrien vereinfachen:



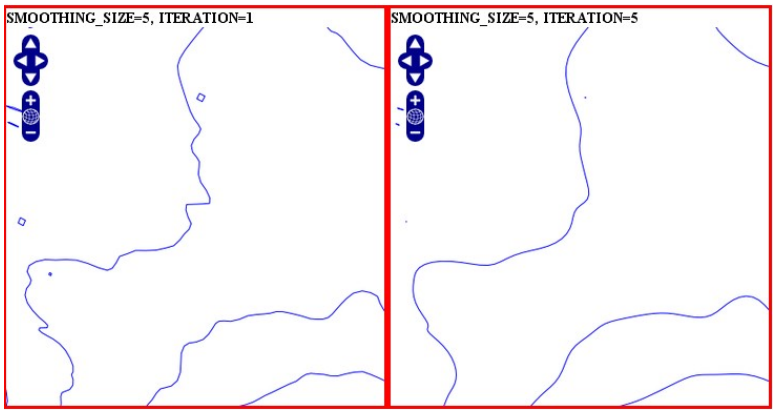

```
LAYER
   NAME 'naturalsymplify'
   TYPE POLYGON
   DATA './shapes/natural.shp'
   GEOMTRANSFORM (simplify([shape], 0.0005)) # Einheiten der Geodaten
```

Neben simplify gibt es weitere Methoden, für die ich hier auf die offizielle Dokumentation verweisen möchte, die auch zahlreiche weitere bebilderte Beispiele bietet[3].

Über die Vereinfachung von Geometrien hinaus besteht auch die Möglichkeit sie geschmeidiger zu machen oder zu generalisieren. Im Ergebnis sieht das so aus wie im folgenden Beispiel, das ich der Online-Dokumentation entnommen habe:

3 http://mapserver.org/mapfile/geomtransform.html#generalize-shape-tolerance

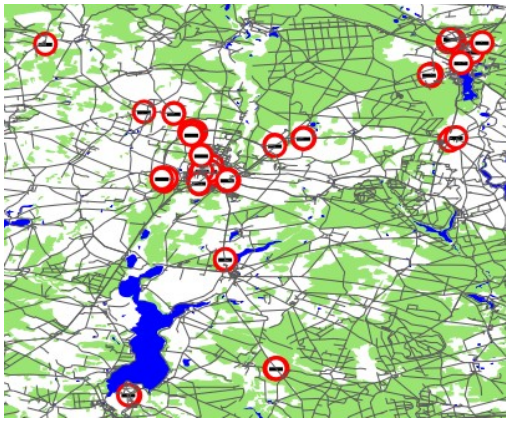
```
LAYER NAME "my_layer"
   TYPE LINE
   STATUS DEFAULT
   DATA roads.shp
   GEOMTRANSFORM (smoothsia([shape], 3, 1, 'angle'))
   CLASS
      STYLE
         WIDTH 2
         COLOR 255 0 0
      END
   END
END
```




Die dritte Anforderung war, dass die Einbahnstraßen **ein 'Durchfahrt verboten' Schild am Ende** erhalten. Eine Möglichkeit wäre nun sicherlich die Enden der betroffenen Straßen als Punkt-Shape-Datei zu erfassen und diese Datei als Punktlayer einzubinden. Das ist offensichtlich sehr aufwendig, zumal jedes mal, wenn die Straßen aktualisiert werden, auch die Punkte aktualisiert werden müssten. Zum Glück kann uns auch hier GEOMTRANSFORM weiter helfen. Mit *GEOMTRANSFORM 'end'* können wir auf den letzten Punkt einer Linie zugreifen und diesen einen Stützpunkt mit einer Punktsignatur versehen. Alternativ steht auch *GEOMTRANSFORM 'start'* zur Verfügung; beides setzt natürlich voraus, dass die Einbahnstraßen einheitlich in oder entgegen der Fahrtrichtung digitalisiert wurden. Im Mapfile sieht es vereinfacht wie folgt aus:

```
CLASS
   STYLE
      GEOMTRANSFORM "end"
      SYMBOL "circle"
      COLOR 255 0 0
      SIZE 20
   END
END
```

Das einfache Beispiel zeigt natürlich nur wie ein roter Kreis ans Ende der Linie gezeichnet wird. Die weiteren Style-Blöcke, die den weißen inneren Kreis und den schwarzen Balken zeichnen sind analog aufgebaut und können uns hier im Detail erspart bleiben. Das Ergebnis sieht jedenfalls so aus:



## Zugriffsbeschränkung über Client-Ips

Oft sollen WebMapServices oder bestimmte Layer nur einem eingeschränkten Nutzerkreis zur Verfügung stehen. Das kann während der Entwicklungsphase der Fall sein, wenn wir einen WMS konfigurieren und nur der Kunde als Externer die Möglichkeit haben soll sich den WMS anzusehen. Es ist aber auch vorstellbar, dass es in einem öffentliche WMS einzelne Layer gibt, die nur für einen eingeschränkten Nutzerkreis verfügbar sein sollen. Man kann diese Dienste nun mit einem OWS-Proxy oder anderen zusätzlichen Schichten in der Architektur absichern oder auf die MapServer-Immanente Möglichkeit der Zugriffsbeschränkung auf bestimmte IP-Adressen oder Adressbereiche zurück greifen. Sicher ist das in Produktivsystemen nicht immer die erste und beste Wahl, aber es bietet eine schnelle und einfache Möglichkeit der Zugangsbeschränkung.

Seit 2013 gibt es zwei neue Parameter für die Metadaten-Sektion, wie so oft können sie im Layer und/oder im Header stehen:

- o**ws_allowed_ip_list**: eine Liste der IPs oder IP-Ranges die auf den WMS zugreifen dürfen
- **ows_denied_ip_list**: eine Liste der IPs oder IP-Ranges die nicht auf den WMS zugreifen dürfen

```
LAYER
   METADATA
      "ows_allowed_ip_list" "123.45.67.89"
   END
END
```

Die IPs können auch in einer gesonderten Datei geführt werden:

```
WEB
   METADATA
      "ows_allowed_ip_list"  "file:/path/to/list_of_ips.txt"
   END
END
```

Es müssen nicht notwendigerweise Einzel-IPs aufgeführt werden, auch IP-Ranges per CIDR Notation sind erlaubt: “192.168.1.0/24”. Weitere Informationen online unter http://mapserver.org/development/rfc/ms-rfc-90.html.

## Zwischen cgi und MapScript: PHP-Fassaden

In den vorangegangenen Abschnitten konnten wir sehen, dass der MapServer eine Menge an Möglichkeiten zu bieten hat, die auf den ersten Blick nicht so offensichtlich sind. Sei es, dass die Informationen in der ausführlichen Online-Dokumentation unter gehen oder sei es, dass die Konfigurationsmöglichkeit ihren Weg noch gar nicht in die Dokumentation gefunden hat, wie bei der Zugriffsbeschränkung über die Client-IPs. Was aber wenn eine wirklich neue Anforderung auftritt? Klar: Wir arbeiten mit freier Software. Ich nehme mir den Quellcode, baue meine Funktion ein, kompiliere das Ganze auf meinem Server und fertig. Wenn ich meine Installation dann upgraden muss, baue ich meinen Code wieder ein, muss ihn vielleicht anpassen, damit er mit dem neuen MapServer funktioniert. Nein, dann doch lieber den offiziellen Weg gehen, meine Anpassung einreichen, gut argumentieren und darauf bauen, dass sie ihren Weg in den MapServer findet. Wenn ich nicht selbst programmieren kann, muss ich jemanden beauftragen, der all das für mich macht. Beide Wege sind möglich. Auf jeden Fall dauern sie eine Weile unter Umständen eine zu lange Weile für den Abschluss meines Projekts und die Beauftragung eines guten Entwicklers kostet natürlich auch mehr als ursprünglich eingeplant. Nicht immer, aber häufig gibt es in diesen Fällen einen Mittelweg indem ich das bisschen Code, das ich brauche um den MapServer herum schreibe[4]. Ich entwickle ein PHP-Script (oder ein Script in einer beliebigen anderen Sprache, die ich beherrsche), das einen WMS-Request entgegen nimmt, ihn vielleicht verändert und dann standard-konform an den MapServer weitergibt. Vom MapServer erhält mein Script das Abfrageergebnis, das Kartenbild oder ein XML-Dokument, und kann dieses bei Bedarf wieder modifizieren bevor das Ergebnis an den Client zurückgegeben wird. Zur Veranschaulichung ein Beispiel: Der http-Request http://isk.geobasis-bb.de/ows/dnm.php? liefert mir die Capabilities des WMS Digitales Navigationsmodell der Landesvermessung und Geobasisinformation Brandenburg (LGB). Hänge ich einen GetMap-Aufruf hinten ran, erhalte ich ein Kartenbild. Es steckt also ein echter WMS dahinter; wie kann mir aber ein WMS seine Capabilites schicken, wenn ich sie gar nicht ordentlich anfrage? Die Antwort steht ein paar Zeilen weiter oben. Das PHP-Script dnm.php prüft ob beim Aufruf Parameter übergeben wurden (z.B. 'REQUEST=GetMap&BBOX=....'). Ist das nicht nicht der Fall, stellt das Script einen GetCapbilites-Request an den WMS namens DNM und gibt die Antwort / das XML an den anfragenden Client zurück. Werden hingegen Parameter an das PHP-Script übergeben, werden diese Parameter auch den WMS weiter gegeben. Ich stelle mir das dnm.php ungefähr so vor:

```
<?php
$url = "http://[mapserver der lgb]/cgi-bin/mapserv?";

IF ( empty($_GET) ) {
$url .= "map=/d/osm.map&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetCapabilities";
$answer = file_get_contents($url);
$answer = str_replace("cgi-bin/mapserv?", "[mapserver der lgb]/dnm.php?",
       $answer );
ECHO $answer;
}
```

Hier ist schon der erste Teil zu Ende. Wenn das Script ohne Parameter aufgerufen wurde ( `IF ( empty($_GET) )` ), holt es vom WMS die Capabilities (`file_get_contents($url)`) und gibt sie mit `echo` wieder aus. Alles Weitere behandelt den Aufruf von dnm.php mit Parametern ich habe das hier stark verkürzt:

4 Und hier geht es mir wirklich um die Fälle, in denen eine schnelle einmalige Lösung her muss. Haben wir Zeit und ist abzusehen, dass wir eine langfristige Lösung brauchen, ist der Weg über die Implementierung in den meisten Fällen sinnvoller.

```
WHILE( list ( $key, $val ) = each ( $_GET ) )
{
      $url .= $key.'='.urlencode($val).'&';
}

ELSEIF ($_GET['REQUEST'] == "GetFeatureInfo" or $_GET['request'] == "Get-
FeatureInfo") {
   $answer = file_get_contents($url);
   $answer = utf8_decode($answer);
   ECHO $answer;
}
// GetMap, GetCapabilities und GetLegendGraphic lasse ich hier aus
// Platzgründen weg
?>
```

Das komplette Script mit Kommentaren ist in den Materialien zum Workshop zu diesem Vortrag zu finden[5]. Sicher sieht das Script der LGB nur so ähnlich aus, vor allem wird es dort deutlich mehr Routinen zur Absicherung und Fehlerbehandlung geben, das Beispiel soll auch nur das Grundprinzip verdeutlichen.

Ich kann das jetzt alles nach Belieben weiter ausbauen. Zum Beispiel kann ich wie im folgenden Beispiel mit meinem PHP-Script die Ergebniskarte verändern bevor ich sie an den Client weiter reiche indem ich die Kontraste erhöhe, die Karte entsättige oder irgendetwas anderes damit anstelle. Die Beispiele können unter wmsfilter.osmct.de im Browser angesehen werden.

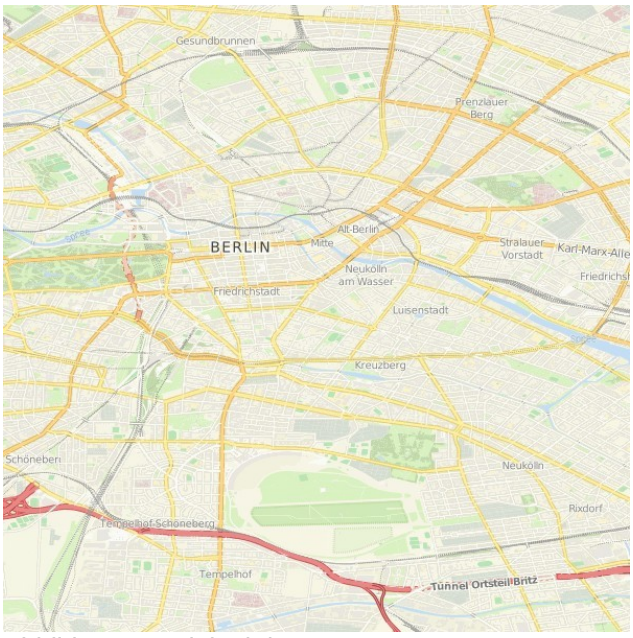


Abbildung 1: Original des WMS

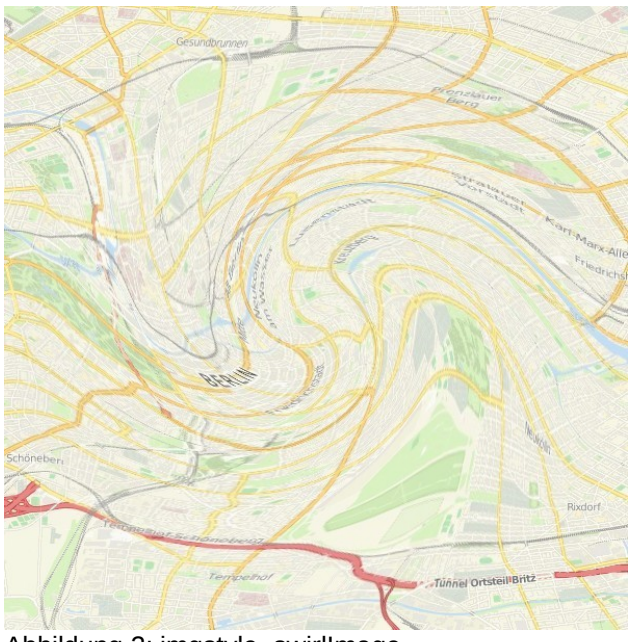


Abbildung 2: imgstyle=swirlImage

5 http://mapmedia.de/downloads/viewcategory/5-vortraege

Abbildung 3: imstyle=solarizeImage

```
http://wmsfilter.osmct.de/imagick_wms.php?
imstyle=solarizeImage&LAYERS=osm&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&S
TYLES=&FORMAT=image%2Fjpeg&SRS=EPSG
%3A4326&BBOX=13.34990092041,52.453745290526,13.452897746582,52.556742116698
&WIDTH=600&HEIGHT=600
```

Wir sehen, dass zusätzlich zu dem GetMap-Parametern der Parameter IMSTYLE übergeben wird. Das Script imagick_wms.php wertet diesen Parameter aus. Alle anderen Paamter werde 1:1 an den dahinter liegenden WMS weiter gereicht. Das Kartenbild des WMS wird vom Script entgegen genommen und gemäß des Werts von IMGSTYLE verändert, bevor das Script das Bild an den Client ausliefert.

Nun muss man nicht unbedingt das Kartenbild ändern. Denkbar – und vorgekommen – ist auch, dass alle Aufrufe durchgereicht werden und lediglich beim GetLegendGraphic-Request ein Legendenbild ausgeliefert wird, das nicht vom MapServer kommt sondern anderweitig generiert wird (seien wir ehrlich, so toll sind die Legendbildchen vom MapServer nicht).

## Weiter führende Informationen

Es gibt zahlreiche weitere Tipps zur Arbeit mit dem MapServer die man als Pro-Tipps weiter geben könnte, sicher ist die Auswahl immer persönlich was man täglich anwendet ist keinen besonderen Hinweis Wert, was einem besonders gut gefällt und in einer schwierigen Situation geholfen hat rutscht in der persönlichen Bewertungsskala nach oben. Wichtig ist sich nicht auf der Grundlage des eigenen Kenntnisstands mit einem „das geht nicht“ abzufinden, sondern nach Lösungen zu suchen, vielleicht auch in den Ergebnissen der Suchmaschinen die zunächst wenig hilfreich erscheinen und natürlich andere Nutzer und Entwickler auf der Mailingliste zu fragen.

Einige weitere Kniffe habe ich für die Workshops er FOSSGIS zusammen gefasst, die Unterlagen sind auf http://mapmedia.de/downloads/viewcategory/5-vortraege zu finden.

Darüber hinaus hat mir auch der gleichnamige Vortrag von Michael Smith und Jeff McKenna auf der FOSS4G 2014 sehr gut gefallen. Dort finden sich auch ein paar gute Tipps, die weniger technisch sind

wie „beachte die Namenskonventionen“ oder „nutze Syntax-Highlighting[6]“. Der Vortrag ist Video und als Foliensatz online:

http://de.slideshare.net/gatewaygeomatics.com/mapserver-protips

https://vimeo.com/106867779

Kontakt zum Autor:

Jörg Thomsen
Trainer der FOSS-Academy und Lehrbeauftragter der Beuth-Hoochschule Berlin

MapMedia GmbH
Gillweg 3, 14193 Berlin
+49 30 890 682-70
jt@mapmedia.de

6 Dazu möchte ich in meiner letzten Fußnote anmerken, dass man nicht unbedingt ein spezielles MapServer-Syntax-Highlighting benötigt, die Auswahl des Editors wird dadurch unnötig eingeschränkt. Ich nutze in geany die Hervorhebung für SQL-Code, aber auch die Hervorhebung für C und seine Derivate ist einen Blick wert.

# Geospatial Ruby

Mila Frerichs

Der Vortrag **»Geospatial Ruby«** soll einen Überblick darüber geben, was mit Ruby im Geo Bereich möglich ist.

**Zum Hintergrund**

Auf der letzen FOSS4G in Portland (2014) trafen sich einige Ruby Entwickler die im Geo Bereich aktiv sind um sich auszutauschen. Diese kleine Gruppe hat sich zum Ziel gesetzt Ruby der Geospatial Community näher zu bringen. Ich war Teil dieser kleinen Gruppe und will meinen Beitrag mit diesem Talk leisten.

**Ruby**

Ruby ist eine dynamische, freie Programmiersprache, die sich einfach anwenden und produktiv einsetzen lässt. Sie hat eine elegante Syntax, die man leicht lesen und schreiben kann und eine sehr aktive und Test-freudige Community hat.

Ruby ist eine Sprache der Balance. Ihr Schöpfer Yukihiro "Matz" Matsumoto kombinierte Teile seiner Lieblingssprachen (Perl, Smalltalk, Eiffel, Ada und Lisp) und formte daraus eine neue Programmiersprache, in der funktionale und imperative Programmierung ausbalanciert sind.

Viele große erfolgreiche Webprojekte sind mit Ruby und dem dazugehörigen Webframework Rails umgesetzt worden.

**Geospatial Ruby**

Der Fokus von „Geospatial Ruby" liegt auf drei wichtigen Bibliotheken. Alle drei vereinfachen den Umgang mit Geo Daten in einer sehr erfolgreichen und angenehmen zu schreibenden Sprache.

*Terraformer*

Die erste Bibliothek ist das **terraformer.rb** von ESRI (Washington & Portland), welches eine Portierung der Javascript Bibliothek **terraformer.js** ist.

Terraformer hat Tools zum arbeiten mit und transformen von GeoJSON. Es bietet eine leichtgewichtige API um geographische Daten zu speichern und abzufragen.

Es ermöglicht die Konvertierung von ArcGIS Geometrie Objekten zu GeoJSON oder Terraformer Objekten. Darüber hinaus ist es mittels des WKT Parsers möglich WKT Objekte in GeoJSON oder Terraformer Objekts zu konvertieren.

*rgeo*

Die zweite Bibliothek ist **rgeo**. rgeo ist die Allzweckwaffe für alles was Geo in Ruby betrifft. Sie vereinfacht das Schreiben von Location-Aware Applikationen.

In seinem Kern ist es eine Implementierung dees Industriestandard „OGC Simple Features Spezifikation" Dies macht es ideal für die Modellierung von Geolokalisierungsdaten.

### SimpleTile

Die dritte Bibliothek ist **SimpleTile** von propublica. Mit dieser Bibliothek ist es sehr einfach einen Tile-Server aufzusetzen. Es können sowohl einfache Raster-Tiles als auch komplexere Vektor-Tiles ausgeliefert werden. SimpleTile ist ein Wrapper für die C-Bibliothek SimplerTiles.

Kontakt zum Autor:

Mila Frerichs
Civic Vision
mila.frerichs@civicvision.de

# Daten aus OSM extrahieren und in QGIS weiterverarbeiten

## Einführung

Die OSM-Daten enthalten viele umweltrelevante Informationen, die auf den veröffentlichten Webkarten nicht offensichtlich erkennbar sind. Mit Hilfe von QGIS lassen sich diese Informationen auswerten und in neue aussagekräftigen Karten umsetzen.

Der Vortrag erläutert die Fragestellung am Beispiel der Verteilung von Windkraftanlagen in Deutschland. Angefangen vom Import der deutschlandweiten OSM-Daten in eine PostGis-Datenbank, über die Abfrage der WKA-Standorte bis hin zur Präsentation der Verteilung in einer farbigen Flächendichtekarte, die aus einer interpolierten Rasteroberfläche erzeugt worden ist, welche die Anzahl der Windanlagen im Umkreis von 20km darstellt. Dabei kamen Abfrage- und Geoverarbeitungswerkzeuge aus QGIS sowie das GRASS-Modul zur Spline-Interpolation (v.surf.rst) zur Anwendung.

## OSM als Datenquelle

Es gibt aktuell keinen einfach zu beschaffenden bundesweiten Datensatz mit den Standorten sämtlicher Windanlagen. In diesem Kapitel wird am Beispiel der Windanlagen exemplarisch dargestellt, wie die Datenbank des Crowdsourcing Projektes *OpenStreetmap* (OSM) zur Gewinnung naturschutzfachlich relevanter Daten genutzt werden kann.[1] Ziele ist es, eine Karte zur Verteilung der Windkraftanlagen in Deutschland zu erstellen, um Regionen zu erkennen, in denen bisher nur eine geringe Dichte von Anlagen vorhanden ist. Diese resultierenden Daten lassen sich dann für bundesweite Auswertungen z.B. im Bezug zum Vogelschutz nutzen.

Der *Bundesverband Windenergie* gibt die bundesweiten Standortdaten nicht als Geodaten heraus, führt aber eine Statistik, in der für Dezember 2013 insgesamt 23.654 Windanlagen verzeichnet waren.[2] Das ATKIS-Basis-DLM umfasst nach Koldrack et. al. im Jahr 2012 insgesamt 21.137 Windanlagen. Eine weitere Datenquelle ist das *EEG-Anlagenregister* mit insgesamt 23.013 Anlagen, dessen Daten jedoch nur über eine Adresscodierung und keine exakten geographischen Verortungen verfügen. Die OSM-Datenbank hatte 2013 insgesamt 21.300 Windanlagen erfasst (90 %).

Es mag eine überraschende Idee sein, in der Datenbank von OpenStreetmap nach WKA-Standorten zu suchen. Doch fällt schon beim Betrachten der OSM-Webkarte auf, dass Windanlagen in großer Zahl verzeichnet sind. OSM begann mit dem Ziel, eine frei verfügbare Straßenkarte der Welt auf Grundlage selbst erfasster GPS-Geodaten zu erschaffen, erfasst jedoch inzwischen jede Art von räumlichen Daten und gewinnt in vielen Gebieten den Charakter einer topographischen Karte. Die Karte des Projektes OpenTopoMap zeigt, dass sich eine der TK 25 ähnliche Darstellung mit OSM-Daten umsetzen lässt.[3] Flächennutzung und topographische Einzelobjekte werden in einzelnen Regionen so präzise kartiert, dass die Qualität in Genauigkeit und Fehlerfreiheit in manchen Gegenden die neue ATKIS-DLM-TK 25 hinter sich gelassen hat. Dies lässt sich exemplarisch für den Bereich westlich Kassels sehr gut erkennen. Auch wenn die Flächennutzung im landwirtschaftlichen Bereich noch sehr uneinheitlich erfasst wird, ist das topographische Potential von Karten aus OSM deutlich zu erkennen..

## Datenbeschaffung und Datenimport

Viele Objekte, die in der OSM-Datenbank erfasst werden, sind nicht zwangsläufig in der über das Web veröffentlichten Karte sichtbar. Jedoch lassen sich die öffentlich verfügbaren Rohdaten mit Hilfe üblicher GIS-Software auswerten und visualisieren. Die zu verarbeitenden Datenmengen sind sehr groß und strukturell sperrig. OSM-Daten von Deutschland werden tagesaktuell über den Server der Geofa-

1 http://www.openstreetmap.de/
2 http://www.wind-energie.de/infocenter/statistiken/deutschland/windenergieanlagen-deutschland
3 http://opentopomap.org

brik als Shapefile-Extrakt und als Rohdaten angeboten.[4] Die dort aufbereiteten Shapefiles enthalten nur einen Teil der Daten und sind für viele Fragestellungen deshalb nicht geeignet. Für die Analyse der WKA werden die Originaldaten im PBF-Format benötigt. Die ca. 2 GB große Datei *germany-latest.osm.pbf* enthält die Gesamtdatenbank für OSM in Deutschland.[5] Dieser Datenbestand ist nur handhabbar, wenn er in eine performante Geodatenbank importiert wird, so dass die PBF-Datei in einem ersten Schritt mit Hilfe des Programms *ogr2ogr* in eine PostGis-Datenbank geladen worden ist.[6] Das

```
ogr2ogr -f PostgreSQL PG:"dbname='OSM' host='localhost' port='5432' user='postgres' password='passwort'"
-gt 65536 -s_srs EPSG:4326 -t_srs EPSG:25832 -skipfailures germany-latest.osm.pbf
```

Kommando zum Import in die Datenbank OSM sieht folgendermaßen aus:

Die resultierende Datenbank war Anfang 2014 ca. 12 GB groß und enthält einen Polygonlayer, einen Linienlayer mit dem Verkehrs- und Gewässernetz sowie einen Punktlayer mit insgesamt 5.270.054 Punkten.

## Windanlagen auswählen

In jedem GIS-Programm lassen sich die Windanlagen mit einer SQL-Where-Abfrage über den Abfageeditor auswählen und anschließend als eigenen Datensatz speichern. Auf der Internetseite des OSM-Projektes findet man die Attributierungsregeln für Windanlagen.[7] Die Attributierung, das sogenannte *Tagging*, lautet für Windanlagen: „`generator:source`=*wind*“ und findet sich nach dem PostGis-Import in der Spalte *other_tags.* Mit der Abfrage *"other_tags" like '%wind%' AND "other_tags" like '%generator%'* lassen sich die Windanlagen auswählen. Zum Zeitpunkt der Auswertung waren 21.301 Windanlagen in der Datenbank erfasst, dies entspricht 90 % der vom Bundesverband für Windkraft registrierten Anlagen. Auf einer solchen Grundlage lässt sich eine räumliche Analyse zur Verteilung der Windanlagen in Deutschland umsetzen.

## Geodatenverarbeitung

Abbildung 1 stellt das Verteilungsmuster der Windanlagen in Deutschland da. Die grundlegende Verteilung ist gut zu erkennen. Ziel der Analyse ist jedoch eine kategorisierte Zonierung der Verteilung, die zu anderen räumlichen Datensätzen mit Polygon-Geometrien in Bezug gesetzt werden kann.

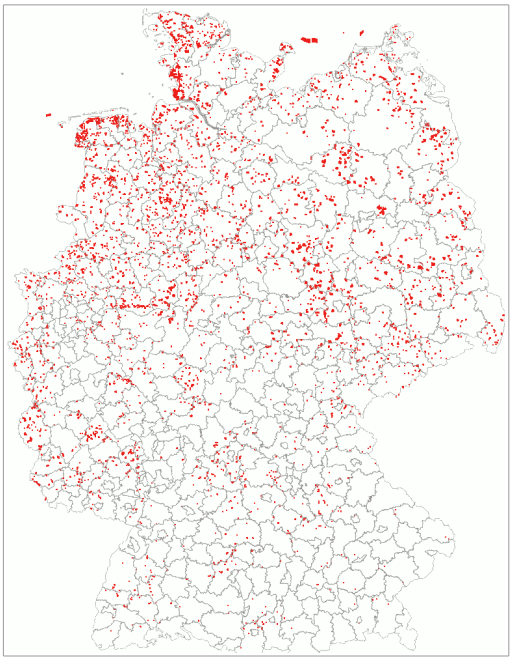


**Abbildung 1: Verteilungsmuster WKA**

Wenn die gesamte Fläche Deutschlands im Bezug zur Dichte der Windanlagen kategorisiert werden soll, muss zunächst eine Methodik ausgewählt werden, mit der die Verteilung zu analysieren ist. So ließe sich ein quadratisches Raster mit 10 km Zellenausdehnung über Deutschland legen, um anschließend die Anzahl der Windanlagen in den Zellen ermitteln zu lassen.

Interessanter ist es jedoch, eine kontinuierliche Oberfläche mir 1 qkm Pixel-Auflösung zu generieren, die über eine farbige Signatur symbolisiert, wie viele WKA sich im Umkreis von 20 km um jeden Pixel befinden. Dies lässt sich mit Hilfe von Interpolationsverfahren erreichen, wenn die Windanlagenpunkte mit Werten über die Anzahl der benachbarten Windanlagen versehen werden. Jede WKA bekommt als

4 http://download.geofabrik.de/europe/germany.html
5 http://download.geofabrik.de/europe/germany-latest.osm.pbf
6 Dies Programm ist teils der GDAL/OGR-Software die beispielsweise mit QGIS ausgeliefert wird
7 *http://wiki.openstreetmap.org/wiki/DE:Map_Features*
Direkt zum Thema: http://wiki.openstreetmap.org/wiki/DE:Tag:generator:source%3Dwind

Attributwert die Anzahl der im Umkreis von 20 km stehenden Windanlagen. Dies wird umgesetzt, in dem

- um die Windanlagen 20 km-Pufferpolygone mit der ID der Ursprungspunkte gelegt werden,
- anschließend die Anzahl der im Pufferbereich liegenden WKA-Punkte ermittelt und als Attribut auf die Pufferfläche übertragen werden (Werkzeug: *SpatialJoin*), um anschließend
- die Werte über *AttributJoin* anhand der ID auf die WKA-Punkte zurückzuübertragen.

Anschließend ist für jeden WKA-Punkt die Anzahl der in 20 km befindlichen Windanlagen als Attribut verzeichnet. Es gibt allerdings Bereiche, in denen sich im Umkreis von 20 km keine einzige Windanlage befindet. Da jedoch nur die WKA zum Sammeln der Punkt-Dichtedaten verwendet worden sind, gibt es keine Punkte mit dem Wert 0, so dass bei einer anschließenden Interpolation auch keine Bereiche ermittelt werden können, in denen sich keine Windanlagen im Umkreis von 20 km befinden. Aus diesem Grunde muss ein Hilfs-Punktraster erstellt werden, um in Bereichen ohne Windanlagen Datenpunkte mit dem Wert 0 für die Interpolation zu erzeugen.

Zu diesem Zweck wird eine regelmäßige Punktmenge mit 20 km-Abstand generiert, auf die im vorher beschriebenen Verfahren die Anzahl der benachbarten Windanlagen übertragen wird. Die resultierende Punktmenge wird mit den WKA Punkten vereinigt, so dass eine Punktmenge entsteht, die auch Punkte mit dem Attributwert 0 enthält. Abschließend wird aus dem Punktshapefile mit den Dichtewerten eine Rasteroberfläche mit 1 km² Auflösung im Spline-Verfahren (GRASS v.surf.rst) interpoliert. Das Ergebnis ist in Abbildung 2 zu sehen.

Die resultierende Oberfläche stellt ein nachvollziehbares Bild der Verteilung von Windanlagen in Deutschland da. Aus der Rasteroberfläche lassen sich in einem GIS-Rasterrechner gewünschte Wertebereiche abfragen und in einzelne Zonen-Shapefiles für konkrete bundesweite Fragestellungen exportieren. So ist es möglich, extrahierte Zonen besonders hoher Anlagendichte mit dem Vorkommen spezifischer Arten, mit Vogelzugrouten oder mit bestimmten Schutzgebieten zu verschneiden.

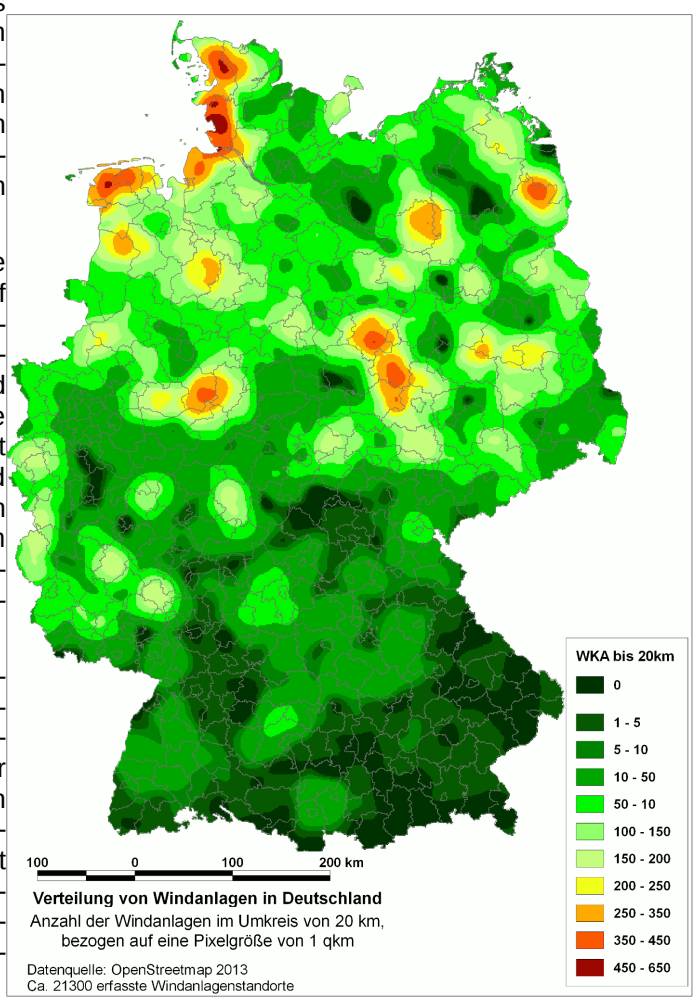



**Abbildung 2: Windanlagendichte in Deutschland**

## Fazit

Wo es keine offiziellen Daten gibt, lohnt es sich in öffentlich zugänglichen Datenbanken nachzusehen. *OpenStreetmap* kann dabei eine ertragreiche Quelle sein. Es Bedarf in jedem Einzelfall einer sorgfältigen Prüfung, ob die Daten für den jeweiligen Zweck nutzbar sind. Im Bereich Verkehrswege, erneuerbarer Energien und Stromleitungstrassen ist OSM eine interessante Quelle. Anzumerken ist, dass in OSM auch Daten zu finden sind, für die es ansonsten gar keine Quellen gibt. So werden viele inoffizielle MountainBike-DownHill-Abfahrten in OSM eingetragen.

## OpenStreetmap als Datenquelle für besondere Verkehrsarten

Das Straßennetz war von Anfang an Schwerpunkt der Datenerfassung im OSM-Projekt. Die Vollständigkeit des in OSM erfassten Straßennetzes erreicht bis zur Ebene der Kreisstraßen flächendeckend das Niveau des ATKIS-DLM. Auf der Ebene von Gemeindestraßen und Wegen ist die Vollständigkeit regional sehr unterschiedlich. In vielen Gegenden sind kleinste Pfade und Fußwege erfasst, in anderen fehlen selbst einzelne asphaltierte Feldwege.

Die Attributierung ist auf Kategorien des amtlichen ATKIS-DLM-Straßennetzes übertragbar, wie die Modellierung beider Netze in Abbildung 3, S. 74 bei einer Gegenüberstellung für den Bereich der Stadt Kassel zeigt.

**Straßennetz Kassel aus OSM und ATKIS-DLM**

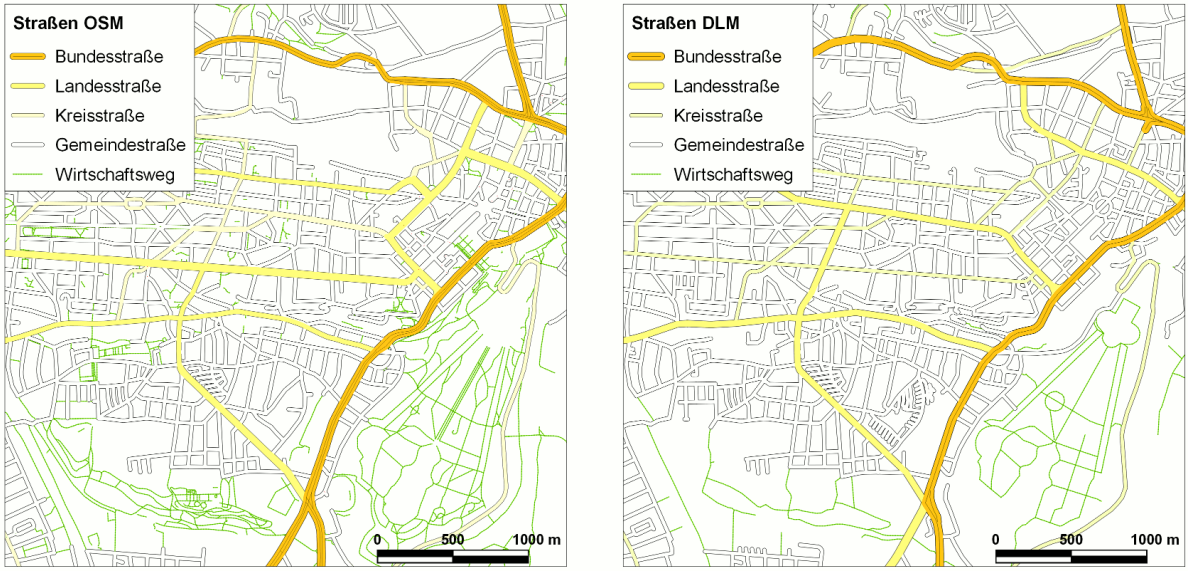



**Abbildung 3: Vergleich des Straßennetzes Kassel auf Basis von ATKIS und OSM**

OSM-Daten sind als Datenquelle brauchbar, bieten jedoch keine wesentlichen Vorteile bei Analysen überregionaler Straßennetze, wenn der Zugriff auf amtlichen Daten vorhanden ist. Die Aktualität der OSM-Daten ist jedoch in vielen Gebieten höher als die der amtlichen Daten.

Bei der Suche nach ganz speziellen Daten ist OSM eine erstaunlich ergiebige Quelle. Im folgenden Beispiel werden Mountain-Bike-Strecken auf unbefestigten Wegen deutschland-weit ausgewählt und visualisiert. Stellt man das Ergebnis naturschutzfachlich bedeutenden Waldlandschaften gegenüber, ist es möglich Regionen einzugrenzen, in denen Konflikte zwischen Naturschutz und Mountainbikern potentiell häufig auftreten können.

Mountainbike-Routen sind in der OSM-Datenbank im Linien-Layer über eine Suche nach dem *tag* ***mtb*** zu finden Gleichzeitig wird die Suche auf unbefestigtes Wege (*highway* = *path)* eingegrenzt.

"other_tags" LIKE '%mtb%' AND „highway“ = 'path'

visualisiert die Verteilung der in der OSM-Datenbank auf vollständig unbefestigten Wegen erfassten Mountainbike-Strecken in zwei verschiedenen Darstellungsarten. Es ist zu erkennen, dass Harz und Pfälzerwald eine auffällige Dichte von Mountainbikestrecken aufweisen.
Die OSM-Daten im Originalformat und als informationsreduzierte Shapefiles finden sich auf. http://download.geofabrik.de/europe/germany.html. Das Straßenetz ist auch in den Shapefiles vollständig und attributiert enthalten.

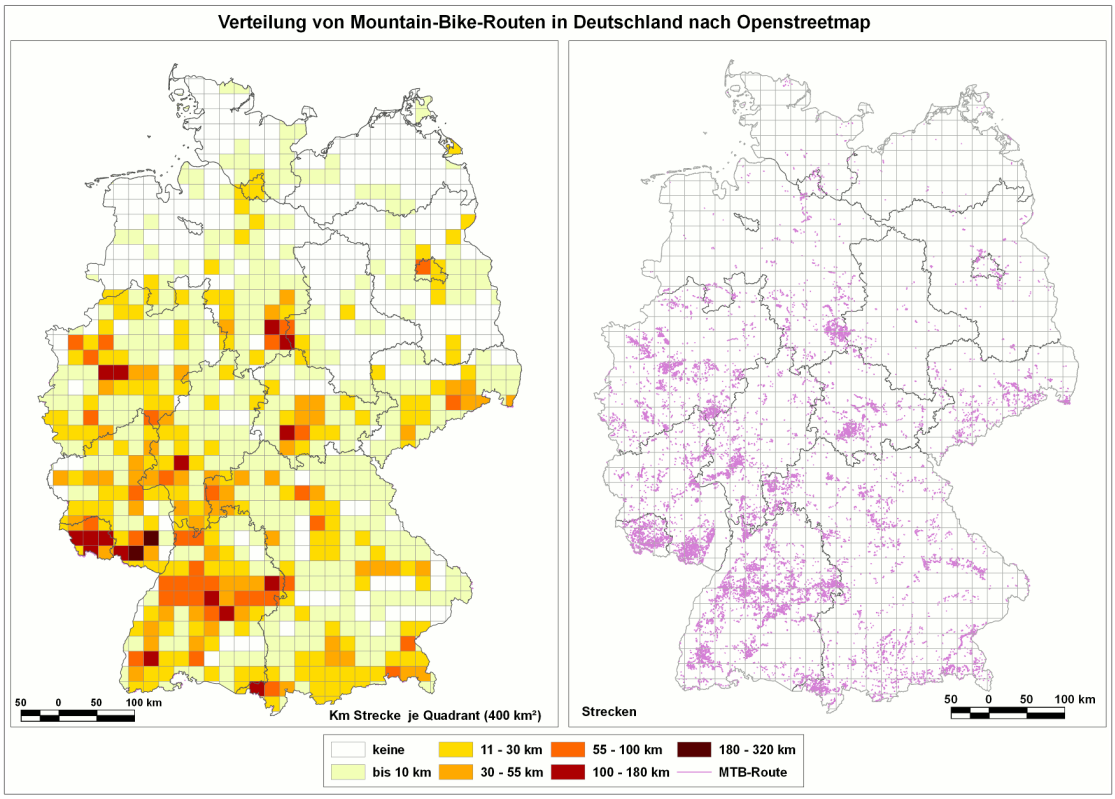



**Abbildung 4: Verteilung der von OSM auf unbefestigten Wegen erfassten Mountainbikestrecken**

## Stromleitungen aus OSM

Auch das Stromleitungsnetz ist in den OSM-Daten nahezu vollständig erfasst. Ein Vergleich des Stromleitungsnetzes aus dem ATKIS-DLM Hessen mit den OSM-Daten zeigt, dass sämtliche im ATKIS-DLM erfassten Stromleitungen dieses Gebietes auch von den OSM-Daten repräsentiert werden. Es handelt sich um Leitungen mit mindestens 110.000 Volt Spannung. Niedrigspannige Freileitungen werden nicht im ATKIS-DLM aber in den OSM-Daten aufgeführt. Zudem sind die Leitungen im OSM-Datensatz zusätzlich nach KV-Spannung attributiert.

Der Vergleich bezieht sich auf die Objektklasse „*DLM_51005_AX_Leitung (1100 = Freileitung)*“ des ATKIS-Basis-DLM und auf die mit „*power=line*“ sowie „*voltage*“ attributierten **ways** (Linien) des OSM-Datensatzes, die über eine entsprechende Attributabfrage extrahiert werden können.

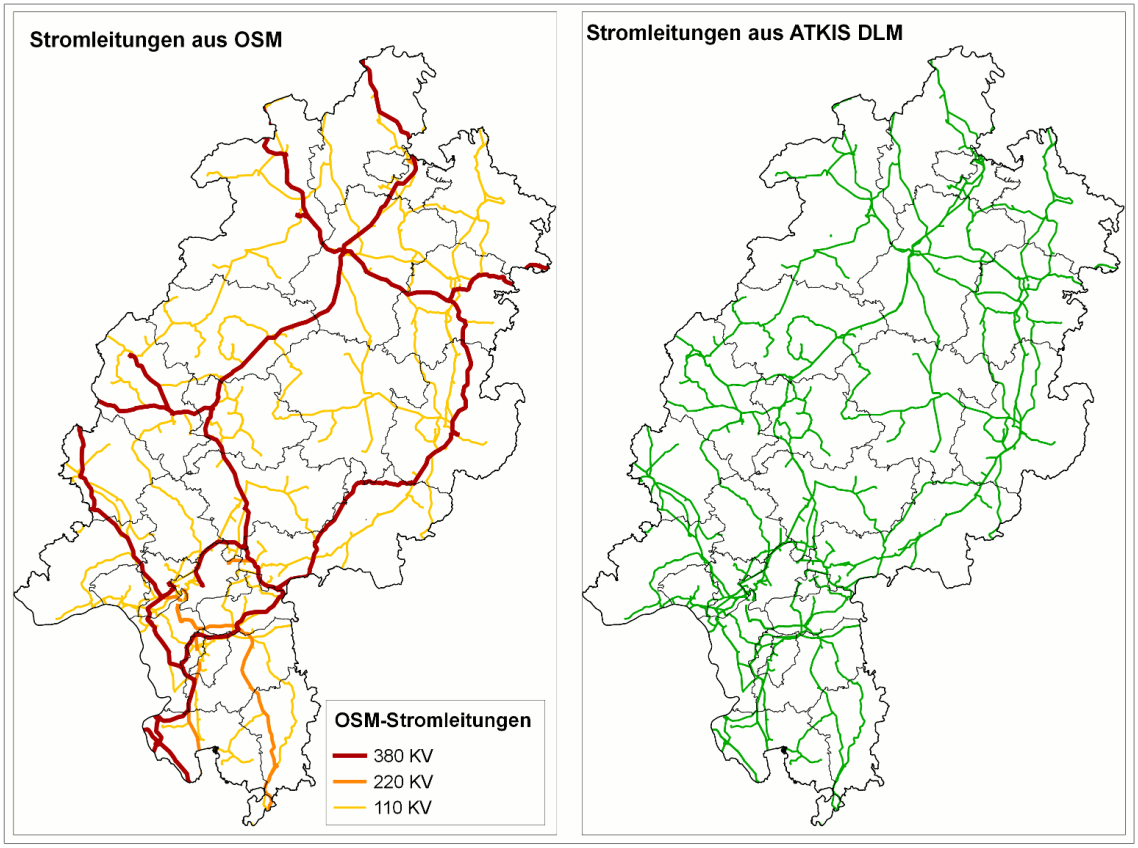



**Abbildung 5: Hochspannungsleitungen in Hessen: DLM und OSM im Vergleich**

## Mapnik oder MapServer?

Mit Mapnik und MapServer stehen zwei OpenSource Kartenrenderer zur Verfügung, die in Geschwindigkeit, Funktionsumfang und Bildqualität kaum Wünsche übrig lassen. Aber welche Software nehme ich für mein Projekt?

Mit Mapnik und MapServer stehen zwei OpenSource Kartenrenderer zur Verfügung, die in Geschwindigkeit, Funktionsumfang und Bildqualität kaum Wünsche übrig lassen. Aber welche Software nehme ich für mein Projekt?

Der Vortrag geht auf die kleinen und großen Unterschiede zwischen Mapnik und MapServer ein. Für welche Einsatzzwecke ist MapServer besser geeignet? Was kann Mapnik besonders gut? Wie können die Renderer in Anwendungen und Server integriert werden? Gibt es überhaupt nennenswerte Unterschiede?

Der Vortrag zeigt ausserdem anhand von Beispielen, welche Software das bessere Kartenbild liefert und welche Verbesserungen von den zukünftigen Versionen Mapnik 3 und MapServer 7 zu erwarten sind.

**Links**

- http://mapnik.org/
- http://mapserver.org/

# Jsonix: mit den OGC Services in JSON sprechen

Alexey Valikov

JSON hat wahrscheinlich XML schon längst als "lingua franca" ersetzt. Es ist viel leichtgewichtiger und einfacher zu verwenden als XML, vor allem in den JavaScript-basierten Web-Anwendungen.

Das Web-Mapping-Umfeld wird von JavaScript-Bibliotheken wie OpenLayers und Leaflet dominiert. Für die gehört JSON buchstäblich zur Muttersprache.

Wie sieht es nun mit den Standards aus? Open Geospatial Consortium (OGC®) definiert mehr als 50 Standards mit insgesamt über 100 einzelnen Versionen. Technisch sind die allerdings fast alle XML-basiert und durch XML-Schemata spezifiziert. Das sind de-jure und de-facto Standards, sehr weit verbreitet und gut unterstützt.

Das heißt, man muss auch in JavaScript-basierten Web Mapping Apps in XML sprechen können.

Es ist keine Rocket Science, aber fast immer lästig. Ein OL3 KML-Parser [2] ist über 2.5KLoc. Auch ein sehr einfaches WMS GetCapabilities Format [3] ist knapp 1KLoc. Und das meiste davon ist reines XML-Parsing, nur ein geringer Teil ist Payload-Verarbeitung.

Wäre es nicht schön, wenn man mit den OGC Web Services direkt in JSON sprechen und sich den Aufwand (und „Spaß“) des XML-Parsens sparen könnte?

## Einführung in Jsonix

Jsonix [1] ist eine Open-Source JavaScript Bibliothek, die es ermöglicht, zwischen XML und JSON zu konvertieren. Mit Jsonix kann man XML in JavaScript-Objekte parsen (dieses Prozess heißt *Unmarshalling*). Oder umgekehrt - JavaScript-Objekte in XML-Form serialisieren (das nennt sich *Marshalling*).

In wie weit hilft es bei der Kommunikation mit den OGC Diensten?

Nehmen wir zum Beispiel den berühmten OGC-Standard WMS [4], und zwar die `GetCapabilities`-Operation. Ergebnis dieser Operation ist ein XML-Dokument, das die vom WMS-Server angebotenen Layers, Styles, Formate usw. beschreibt (siehe [5]).

Wie sieht der Parsing-Code für dieses XML aus (und wie aufwändig er ist), haben wir auf dem Beispiel des OpenLayers WMS GetCapabilities Format gesehen. Anbei die Jsonix-Variante (siehe auch Live-Beispiel in JSFiddle [6]):

```
// Zuerst müssen wir einen Jsonix Context mit WMS und Xlink Mappings instanziieren
var context = new Jsonix.Context([XLink_1_0, WMS_1_3_0]);

// Dann können wir einen Unmarshaller (Parser) erzeugen
var unmarshaller = context.createUnmarshaller();

// Mit diesem Unmarshaller können wir XML von URL (Text, DOM, File, ...) parsen
unmarshaller.unmarshalURL(getCapabilitiesUrl, function(result) {

	// Und das Ergebnis bekommen wir als JavaScript-Objekt zurück
	$('#log').html(JSON.stringify(result, null, 2));
});
```

## Jsonix Mappings

Die Magie von Jsonix steckt in den Mappings - wie `XLink_1_0` und `WMS_1_3_0` in dem Beispiel oben. Diese Mappings sind JavaScript-Objekte, die eine Zuordnung zwischen XML-Konstrukten (Elementen, Attributen, textuellen Inhalten) und Eigenschaften von JavaScript-Objekten deklarativ beschreiben. Siehe [7] als Beispiel des XLink 1.0 Jsonix Mappings.

Das Beste ist, dass man diese Mappings aus XML-Schemata generieren kann. Die `XLink_1_0` und `WMS_1_3_0` Mappings wurden vollautomatisch aus XML-Schemata erzeugt. Ein folgender Command-Line-Befehl spart mehr als tausend LoC:

```
java -jar jsonix-schema-compiler.jar -catalog schemas/catalog.cat
      schemas/ogc/wms/1.3.0 -b schemas -d mappings
```

Das was wir eben mit dem Parsen gesehen haben, funktioniert auch in die andere Richtung. Jsonix ist vollständig bidirektional: Was geparst werden kann, kann auch serialisiert werden und umgekehrt.

Nehmen wir als Beispiel die `GetRecords`-Operation aus der OGC Catalog Service Web (CSW) Schnittstelle. Die `GetRecords`-Operation erlaubt es, die Datensätze (wie z.B. Geometadaten nach Dublin Core oder ISO 19139) abzufragen.

CSW verwendet weitere OGC (und nicht nur OGC) Standards. Schon eine relativ einfache Anfrage (“Bounding Box“-Suche nach Datensätzen mit **“**`WATER DEPTH`**“** im Titel) muss CSW mit Filter Encoding und GML kombinieren (siehe [8]).

So ein XML-Dokument korrekt und valide im JavaScript-Code zu erzeugen ist nicht unmöglich aber auch nicht leicht. Mit Jsonix kann man diese `GetRecords`-Anfrage stattdessen in Form eines JavaScript-Objects zusammenbauen – und dann als XML serialisieren ([9]):

```
{
      "csw:GetRecords" : {
            "service" : "CSW", "version" : "2.0.2",
            "maxRecords" : 10, "resultType" : "results",
            "query" : { /* … */ }
      }
}
```

Das serialisierte XML-Dokument entspricht automatisch den ursprünglichen XML-Schemata.

## Jsonix Features

Da es schon so viele XML-JS-Werkzeuge gibt, wäre es logisch zu fragen - warum noch ein Tool?

Jsonix bietet eine einzigartige Kombination von Features:

- Jsonix ist bidirektional, kann sowohl XML→JS wie auch JS→XML umwandeln;
- Jsonix ist struktursicher;
- Jsonix ist typsicher;
- Jsonix verwendet deklarative Mappings, die entweder manuell geschrieben oder automatisch aus XML-Schema generiert werden können.

Die Umwandlungen XML → JS und JS → XML haben wir auf den Beispielen von WMS GetCapabilities und CSW GetRecords bereits gesehen.

Struktur- und Typsicherheit benötigen einer näheren Betrachtung.

### Struktursicherheit

Was heißt Struktursicherheit und warum ist sie wichtig?

Betrachten wir zum Beispiel folgendes XML-Fragment:

```
<GetMap>
       <Format>image/png</Format>
</GetMap>
```

Soll `getMap.format` nun ein String sein? Oder ein Array von Strings? Oder ist das Format eventuell ein komplexeres Objekt?

Aus dem XML-Fragment alleine kann man keine verlässliche Struktur ableiten. Für den JavaScript-Code würde es bedeuten, es kann mal `getMap.format` oder `getMap.format[0]` oder `getMap.format[0].value` sein. Je nach dem, was in XML steht.

Alle diese Möglichkeiten in JavaScript-Code zu berücksichtigen ist sehr aufwändig. Vernünftigerweise braucht man eine Struktur, auf die man sich wirklich verlassen kann. Im Falle von OGC-Standards soll es eine Struktur sein, die direkt vom XML-Schema des Standards abgeleitet ist.

Genauso wichtig ist die Struktursicherheit bei der Serialisierung JS → XML. Die OGC Web Services (oder Web Services im Allgemeinen) erwarten XML nach konkretem XML-Schema, sonst werden die Anfragen nicht verstanden oder abgelehnt.

In Jsonix wird die XML↔JS-Konvertierung in Mappings deklariert. Das heißt, man bekommt definierte, vorhersagbare, verlässliche Strukturen sowohl auf JavaScript wie auch auf XML-Seite. Mit Jsonix wird `getMap.format` immer genau das, was im Mapping steht (z.B. ein Array von Strings). Auch das XML wird stabil. Und wenn das Mapping aus XML-Schema generiert wurde, wird das XML diesem XML-Schema entsprechen.

### Typsicherheit

Ähnlich ist es mit der Typsicherheit. Nehmen wir folgendes Beispiel:

```
<Layer queryable="1" .../>
```

Welchen Typ soll `queryable` haben? Eine Zahl? Eine Ganzzahl? Ein String?

In der Tat ist es ein Boolean, aber das kann man ohne Schema nicht sagen. Auch hier wird eine Art Konfiguration benötigt, um die Typen der Eigenschaften sicher zu deklarieren.

In Jsonix steht diese Konfiguration in Mappings. Das WMS Mapping für Jsonix definiert eine Attribut-Eigenschaft `queryable` mit dem Typ `Boolean`:

```
{     name: 'queryable',  typeInfo: 'Boolean', type: 'attribute' }
```

So wird `layer.queryable === true` statt Zahl 1 oder String “1”.

### Declarative Mappings

Struktur- und Typsicherheit erfordern allerdings, dass die Definitionen des Schemas auch im Runtime verfügbar sind.

Oft wird es *imperativ* gemacht. Z.B. in OpenLayers-Formaten ist das Mapping XML→JS händisch ausprogrammiert.

Jsonix Mappings dagegen sind *deklarativ*. Die Jsonix Mappings beschreiben lediglich die Strukturen und überlassen es dem `Jsonix.Context`, einer generischen Runtime-Engine, XML in JS oder umgekehrt JS in XML umzuwandeln.

Die Vorteile der deklarativen Mappings sind erheblich:

- Deklarative Mappings können leichter aus XML-Schemas generiert werden;
- Sie haben keine programmatischen Abhängigkeiten und können sehr einfach wiederverwendet werden;

- Sie können in JSON-Form gespeichert werden, was Integration in den JavaScript-Code trivial macht;
- Die Mappings können sogar in der Laufzeit zusammengebaut werden.

Der letzte Punkt lässt sich gut mit einem WFS DescribeFeatureType-Beispiel illustrieren.

Die WFS DescribeFeatureType-Operation kann für den gewählten Feature-Typ eine XML-Schema-Definition liefern. Am häufigsten wird dafür GML Simple Features Profile [10] verwendet. Hier kann man Jsonix wie folgt einsetzen:

- Da XML-Schema des Feature-Profils auch XML ist und ein eigenes Schema [11] hat, kann man das mit Jsonix parsen.
- Dann kann man für das Feature-Profil ein Jsonix-Mapping zur Laufzeit dynamisch zusammenbauern.
- Schließlich kann man dieses Mapping verwenden, um XML von Features nach diesem Feature-Profil zu parsen.

So kann man mit Jsonix die GML Features nach Simple Features Profile struktur- und typsicher parsen – ohne das Schema im vorab zu kennen (siehe Beispiel in JSFiddle [12]).

### Generierung der Mappings

Jsonix Mappings können aus XML-Schemata automatisch generiert werden. Dazu bietet Jsonix ein Werkzeug namens Jsonix Schema Compiler [13] an.

Jsonix Schema Compiler kann über Command-Line sowie über Ant, Maven oder NPM aufgerufen werden, um auf der Basis von eingehenden XML-Schemata die Jsonix-Mappings zu erzeugen. Allerdings wird Java 1.7 oder höher vorausgesetzt.

Jsonix Schema Compiler hat viele Konifugurationsparameter. Man kann die Mappings “verbose” oder “kompakt” generieren, mehrere Mappings in ein Modul packen oder sehr genau steuern, was im XML-Schema von Interesse ist oder in den Mappings weggelassen werden soll.

Eine wirklich fortgeschrittene Funktion ist die Abhängigkeitsanalyse. Z.B. kann man ein sehr schlankes GML 3.1.1-Mapping generieren mit nur dem, was für Filter 1.1.0 benötigt wird [14].

## Das inoffizielle „OGC Schemas Compilation Project“

Allerdings ist es nicht leicht die Schemata zu kompilieren: Der Benutzer muss sich mit dem Werkzeug gut auskennen um die Probleme wie Namenskollisionen oder Auflösung der Schema-Locations lösen zu können.

Um es einfacher zu machen, haben wir das inoffizielle „OGC Schemas Compilation Projekt“ [15] auf die Beine gestellt. Das Projekt bietet vorgenerierte Jsonix-Mappings für mittlerweile mehr als 20 am meisten verbreiteter OGC Schemata mit über 50 einzelnen Versionen.

Das heißt, man muss die OGC-Schemata im Regelfall gar nicht selber kompilieren, sondern kann einfach die vorgenerierten Mappings nehmen.

## ows.js

Jsonix ist nur ein XML↔JS-Konverter, der für die OGC XML-Schemata sehr gut passt. Was Jsonix allerdings nicht im Scope hat, sind konkrete Operationen von OGC-Schnittstellen. Um das plakativ zu machen: Jsonix kann WMS GetCapabilities XML parsen, wird aber selber den WMS GetCapabilities-Aufruf nicht machen. Jsonix alleine ist noch kein Client für die OGC Web Services.

Das ist im Fokus des Projekts ows.js [16]. ows.js ist ein neues OSGeo-Projekt, mit dem Ziel, eine JavaScript-Bibliothek für die OWS-Clients (Clients für die verschiedene OGC Web Services) zu implementieren.

Die ersten Tests haben gezeigt, dass Jsonix dabei sehr hilfreich sein kann. Daher ist die aktuelle Architektur-Idee, dass Jsonix die XML-JS-Konvertierung übernimmt und ows.js auf dieser Basis die APIs für die einzelnen OGC Services und Standards (z.B. CSW, WFS, WMS, …) baut.

## Einsatz von Jsonix im Boundless WPS GUI

Ein prakisches Beispiel für den Einsatz von Jsonix ist der Boundless WPS GUI [17].

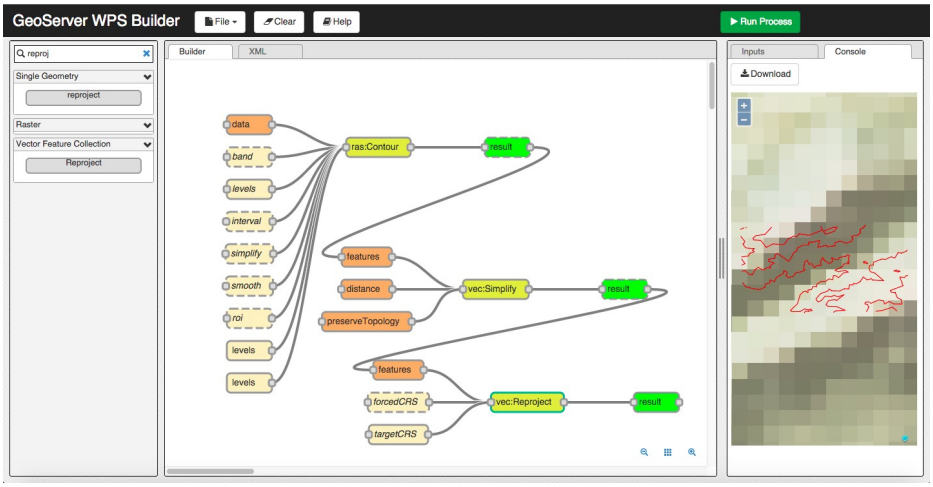



Abb. 1: Boundless WPS GUI

In diesem Projekt übernimmt Jsonix viele Aufgaben der XML-Verarbeitung. Boundless WPS GUI verwendet Jsonix für folgende Operationen:

- WPS GetCapabilities-Response zu parsen um die angebotenen WPS Prozesse aufzulisten;
- WPS DescribeProcess-Response zu parsen um die Ein- bzw. Ausgangsparameter eines ausgewählten WPS Prozesses rauszufinden;
- WFS GetCapabilities-Response zu parsen um rauszufinden, welche vektorielle Layers als Eingangsparameter für die WPS Prozesse verwendet werden können;
- WCS GetCapabilities-Response zu parsen um rauszufinden, welche Raster-Layers als Eingangsparameter benutzt werden können;
- Und um schließlich einen WPS Execute-Request zu bauen und in XML-Form zu serialisieren.

## Future Work

Jsonix bietet ein umfangreiches Set von Features und ist bereit für den produktiven Einsatz. Die zukünftige Entwicklung hat folgende Schwerpunkte:

- Vereinfachung der Schema-Kompilierung;
- Optimierung der Bibliothek-Größe (z.B. durch den Einsatz von Google Closure Compiler im Advanced Mode);
- Unterstützung von Adaptoren (um z.B. GML ↔ OL Geometrien direkt zu konvetieren);
- Unterstützung von Konvertierung ohne (oder nur mit partiellen) Mappings für die einfachen Anwendungsfälle;
- vorgenerierte Mappings für weitere OGC-Schemata;
- und natürlich weiterhin ein fanatischer Support für die Projekte, die Jsonix einsetzen.

## Fazit

Unter dem Strich reduziert Jsonix die Entwicklungsaufwände für XML-JS-Konvertierung drastisch.

**Jsonix: mit den OGC Services in JSON sprechen**

Jsonix ist ein mächtiges XML-JS-Konvertierungswerkzeug. Die Verwendung von XML-Mappings erlaubt es, die Struktur- und Typsicherheit bei der Konvertierung zu garantieren, was für die Verarbeitung von XML nach OGC-Standards sehr wichtig ist. Die Mappings kann man aus XML-Schemata automatisch generieren – oder die bereits vorgenerierten Mappings verwenden.

So kann Jsonix helfen, mit den OGC-Diensten in JSON statt XML zu sprechen.

Kontakt zum Autor:

Alexey Valikov, PhD
Senior Architekt, DB Systel GmbH
Jürgen-Ponto-Platz 1, 60329 Frankfurt am Main
+49 69 265 50000
valikov@gmx.net
http://xing.to/va

Literatur

[1] *Valikov, Alexey*: Jsonix. https://github.com/highsource/jsonix

[2] *OpenLayers 3*: KML Format.

https://github.com/openlayers/ol3/blob/master/src/ol/format/kmlformat.js

[3] O*penLayers 3*: WMS GetCapabilities Format.

https://github.com/openlayers/ol3/blob/master/src/ol/format/wmscapabilitiesformat.js

[4] *OGC*: Web Map Service Interface Standard. http://www.opengeospatial.org/standards/wms

[5] *OpenGeo Demo Web Map Service*: WMS GetCapabilities Response.

http://demo.opengeo.org/geoserver/osm/wms?service=WMS&request=GetCapabilities

[6] *JSFiddle*: Unmarshalling WMS GetCapabilities mit Jsonix. http://bit.do/jsonix-001

[7] XLink 1.0 Mapping für Jsonix.

https://github.com/highsource/w3c-schemas/blob/master/scripts/lib/XLink_1_0.js

[8] CSW GetRecords Example.

https://rawgit.com/highsource/jsonix/master/fiddles/yq8x0oqf/GetRecords.xml

[9] *JSFiddle*: Marshalling CSW GetRecords mit Jsonix. http://bit.do/jsonix-003

[10] *OGC*: GML Simple Features Profile. http://www.ogcnetwork.net/gml-sf

[11] *W3C*: XML Schema for XML Schemas 1.0. http://www.w3.org/2001/XMLSchema.xsd

[12] *JSFiddle*: Dynamic WFS DescribeFeatureType/GetRecords Example. http://bit.do/jsonix-004

[13] *Valikov, Alexey*: Jsonix Schema Compiler. https://github.com/highsource/jsonix-schema-compiler

[14] Konfigurationsdatei GML 3.1.1 für Filter 1.1.0. http://bit.do/jsonix-005

[15] *Valikov, Alexey*: Inoffizielles OGC Schemas Compilation Project.

https://github.com/highsource/ogc-schemas

[16] *Kralidis, Tom et al.*: ows.js. https://github.com/osgeo/ows.js

[17] *Boundless*: WPS GUI. https://github.com/boundlessgeo/wps-gui

**Disclaimer:** OGC®, OGC Web Services™, OWS™ sind registrierte Trademarks von Open GeoSpatial Consortium, Inc.®. Die Zeichen ® und ™ wurden im Text aus Lesbarkeitsgründen weggelassen. Das “OGC Schemas Compilation Project” ist kein OGC® Projekt und wird daher explizit als *inoffiziell* bezeichnet. Das “OGC Schemas Compilation Project” implementiert eine Reihe der OGC® Standards, wird allerdings explizit **nicht** als “OGC Certified Compliant” bezeichnet weil das OGC Compliance Testing nicht durchgeführt wurde.

# taginfo und wie es die Welt sah

Jochen Topf

Das OpenStreetMap-Projekt organisiert seine Daten über ein offenes Tagging-Schema. Jeder kann neue Objekte und Tags erfinden und in der OSM-Datenbank speichern. Dieser Vortrag stellt den Web-Dienst "taginfo" vor, der Informationen zu Tags aus verschiedenen Quellen darstellt und Mappern so hilft, die Übersicht zu behalten. Der Vortrag geht sowohl auf die alltägliche Nutzung als auch auf weniger bekannte Seiten des Dienstes ein.

Das OpenStreetMap-Projekt organisiert seine Daten über ein offenes Tagging-Schema. Jeder kann neue Objekte und Tags erfinden und in der OSM-Datenbank speichern. Damit man dabei nicht die Übersicht verliert, gibt es taginfo, ein Webdienst, der Informationen zu Tags aus verschiedenen Quellen sammelt und darstellt. Taginfo ist für viele Mapper inzwischen zu einem unverzichtbaren Tool geworden. Dieser Vortrag stöbert in einige Ecken des Dienstes, die nicht jeder kennt und zeigt, wie taginfo bei der täglichen Arbeit helfen kann. Daneben wollen wir mit taginfo einen Blick darauf werfen, was in OpenStreetMap so alles zu finden ist.

### Links

- http://taginfo.openstreetmap.org/

# GeoServer in action

## Fortgeschrittene Möglichkeiten beim Einsatz des GeoServers

Nils Bühner

GeoServer in action - Fortgeschrittene Möglichkeiten beim Einsatz des GeoServers. Es geht um Kompilieren, Schnittstellenverwendung, Extensions, Performance-Tuning, GeoWebCache-Einsatz, Troubleshooting und Stolperfallen.

Der GeoServer ist ein weithin bekannter und mächtiger OpenSource Kartenserver. Sofern man die Umgebung eingerichtet hat, ist sowohl die Installation als auch die Konfiguration von ersten WMS- und WFS-Layern sehr einfach. In diesem Vortrag wird auf die typischen Anforderungen eines "GeoServers in action" eingegangen. Der Vortrag fokussiert sich also weniger auf die "Out-of-the-box"-Verwendung des GeoServers, sondern beleuchtet vielmehr fortgeschrittene Möglichkeiten beim Einsatz dieser Software.

In diesem Rahmen werden u.a. folgende Themen behandelt:

- GeoServer auf Basis des Source-Codes selber kompilieren
- Verwendung der REST-Schnittstelle
- Einsatz des GeoWebCache (GWC)
- GeoServer Extensions
- Performance-Tuning für den Produktiveinsatz auf verschiedenen Ebenen
- Typische Stolperfallen und Troubleshooting

# Serverseitiges JavaScript und GIS

## Mit Node.js und npm die Welt beherrschen

Marc Jansen, Christian Mayer

Der Vortrag gibt eine allgemeine Einführung in die Node.js-Welt und zeigt wie mit Node.js geo-relevante Probleme, wie das Einlesen von GIS-Datenformaten sowie die Verarbeitung von Geodaten möglich wird. Ebenso wird eine Übersicht von nützlichen Node.js-Modulen aus dem GIS-Bereich sowie Beispielanwendungen vorgestellt.

Node.js [1] ist eine recht junge Open Source Plattform für serverseitige Netzwerkanwendungen, die erstmalig in 2009 für Linux veröffentlicht wurde. Die Besonderheit ist dabei, dass die Programme in JavaScript geschrieben werden, was bislang nur für Client-Anwendungen im Browser gängig war. Durch die Event-getriebene Architektur sowie einen asynchronen, nicht-blockierenden IO-Mechanismus kann performante und ressourcenschonende Software entwickelt werden. Node.js liegt aktuell (November 2014) in der Version 0.10.33 vor und ist unter der offenen MIT-Lizenz [2] veröffentlicht.

Da JavaScript die führende Programmiersprache für browserbasierte Webmapping-Anwendungen ist, was etablierte Bibliotheken wie OpenLayers, Leaflet oder GeoExt zeigen, wird verdeutlicht welche Möglichkeiten sich JavaScript-GIS-Entwicklern nun mit der serverseitigen Prozessierung eröffnen: Mit etablierten Werkzeugen können Anwendungen und Webservices zur Geoprozessierung erstellt werden ohne die gewohnte Programmiersprache verlassen zu müssen. Des Weiteren liegt mit npm ("Node Packaged Modules", dem Node.js Paket Manager) [3] eine Software vor, die es eminent vereinfacht modular und wiederverwertbar zu entwickeln und auf der Arbeit anderer Entwickler eigene Software aufzubauen.

Der Vortrag gibt eine allgemeine Einführung in die Node.js-Welt und zeigt wie mit Node.js geo-relevante Probleme, wie das Einlesen von GIS-Datenformaten sowie die Verarbeitung von Geodaten möglich wird. Ebenso wird eine Übersicht von nützlichen Node.js-Modulen aus dem GIS-Bereich sowie Beispielanwendungen vorgestellt.

### Links

- [1]: https://nodejs.org/
- [2]: http://opensource.org/licenses/mit-license.php
- [3]: https://www.npmjs.org/

# Crowd-Sourced Elevation

## DIY DEM

Peter Barth

Dieser Vortrag zeigt einen neuen Ansatz auf, wie mit handelsüblichen Smartphones ein hochgenaues digitales Höhenmodell mit Hilfe von Crowd-Sourcing erzeugt werden kann, das sowohl hinsichtlich Auflösung als auch Genauigkeit SRTM weit überlegen ist.

Digitale Höhenmodelle können für eine Vielzahl von Anwendungen im OpenStreetMap-Umfeld verwendet werden. In Renderstilen werden sie z.B. für Schummerungen oder Höhenlinien verwendet. Aber auch bei Routinganwendungen gibt es verschiedenste Gründe, das Geländemodell mit zu beachten: Fahrradfahrer, Rollstuhlfahrer aber z.B. auch Energiesparen bei Elektromobilität.

Mit SRTM steht ein genaues Höhenmodell mit nahezu globaler Abdeckung gemeinfrei zur Verfügung und erst kürzlich wurde auch die Version mit einer Auflösung von einer Bogensekunde freigegeben. Aber reicht das?

Dieser Vortrag beginnt mit einem Überblick zum Thema Höhenmodelle und beschreibt den Umfang, Auflösung und Genauigkeit verschiedener Datenquellen. Er zeigt anhand von Experimenten und deren Auswertung, wie man mit einem handelsüblichen Smartphone selbst ein Höhenmodell erstellen kann. Kann man mit diesen Methoden mit Hilfe von Crowd-Sourcing ein umfangreiches digitales Höhenmodell erzeugen? Ist die zu erwartende Auflösung als auch Genauigkeit SRTM wirklich weit überlegen? Dieser Vortrag wird es euch anhand von Experimenten und Auswertungen beantworten.

### Links

- http://wiki.openstreetmap.org/wiki/ASTER
- http://wiki.openstreetmap.org/wiki/SRTM
- http://www.opendem.info/

# Routing in der Datenbank

Daniel Kastl

pgRouting erweitert eine PostGIS/PostgreSQL Geo-Datenbank um Funktionen für “Kürzeste-Wege” Berechnung (Routing) und andere netzorientierte Algorithmen.

Die Vorteile des Datenbank-Routing Konzepts sind:

- Daten und Attribute können mittels JDBC, ODBC oder direkt durch Pl/pgSQL von zahlreichen Anwendungen modifiziert werden, wie etwa von QGIS und uDig. Solche Anwendungen können sowohl auf PCs als auch auf mobilen Geräten laufen.
- Änderungen an den Daten wirken sich unmittelbar auf das Routingergebnis aus. Es besteht keine Notwendigkeit, Daten vorauszuberechnen.
- Der “Kosten” Parameter kann dynamisch mittels SQL berechnet werden und aus verschiedenen Attributen aus unterschiedlichen Tabellen zusammengesetzt sein.

Zu den bekannten "Shortest Path" Suchalgorithmen, Einzugsbereichsermittlung oder "Travelling Salesperson Problem" (TSP) Optimierung, sind in naher Zukunft neue Funktionen verfügbar, wie der "Vehicle Routing Problem" (VRP) Algorithmus und ähnliche Werkzeuge zur Planung und Optimierung von Touren.

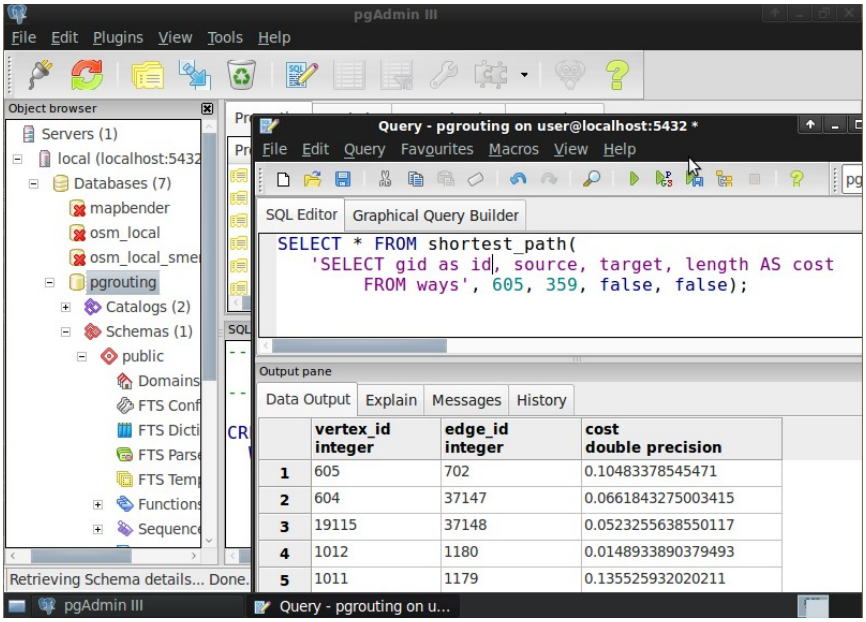


Abb. 1: pgRouting SQL Abfrage

Die aktuelle pgRouting  Version stellt folgende Funktionen bereit:

- All Pairs Shortest Path, Johnson’s Algorithm
- All Pairs Shortest Path, Floyd-Warshall Algorithm
- “Kürzeste-Wege” Berechnung mit A* Algorithmus
- Bi-directional Dijkstra Shortest Path
- Bi-directional A* Shortest Path

**Routing in der Datenbank**

- “Kürzeste-Wege” Berechnung mit Dijkstra Algorithmus
- Einzugsbereichsberechnung (Isolinien)
- K-Shortest Path, Multiple Alternative Paths
- K-Dijkstra, One to Many Shortest Path
- Problem des Handlungsreisenden (Traveling Salesperson Problem, TSP)
- Turn Restriction Shortest Path (TRSP)

pgRouting steht unter GPLv2 Lizens und wird von einer wachsenden Zahl von Nutzern, Organisationen und Unternehmen unterstützt.

Weitere Details zu pgRouting finden sich auf der Projektseite [1] und auf der Github-Seite [2] des Projekts.

Kontakt zum Autor:

Daniel Kastl
Georepublic (Deutschland)
Salzmannstr. 44, 81739 München
+49 (089) 4161 7698-1
daniel.kastl@georepublic.de

Links:

[1] Projektseite: http://pgrouting.org

[2] Source Code: https://github.com/pgrouting

# Erfassung von Landnutzungsveränderungen mit FOSS Image Processing Tools – Change Detection mit QGIS

Jakob Tworek

Im Rahmen einer Masterarbeit am Geographischen Instituts der Universität Bonn wurden zwei Change Detection Verfahren mit Free & Open Source Image Processing Tools angewandt und evaluiert. Auf Grundlage von zwei multispektralen Satellitenbilder (Landsat-5) wurde für den Zeitraum von 1989 bis 2010 einerseits ein Post-Classification Comparison und andererseits ein NDVI Image Differencing mit QGIS für die Region Köln/Bonn durchgeführt. Veränderungen der Landnutzung- und -bedeckung konnten mit diesen Verfahren quantitativ erfassen werden. Das Ziel war es zum einen die

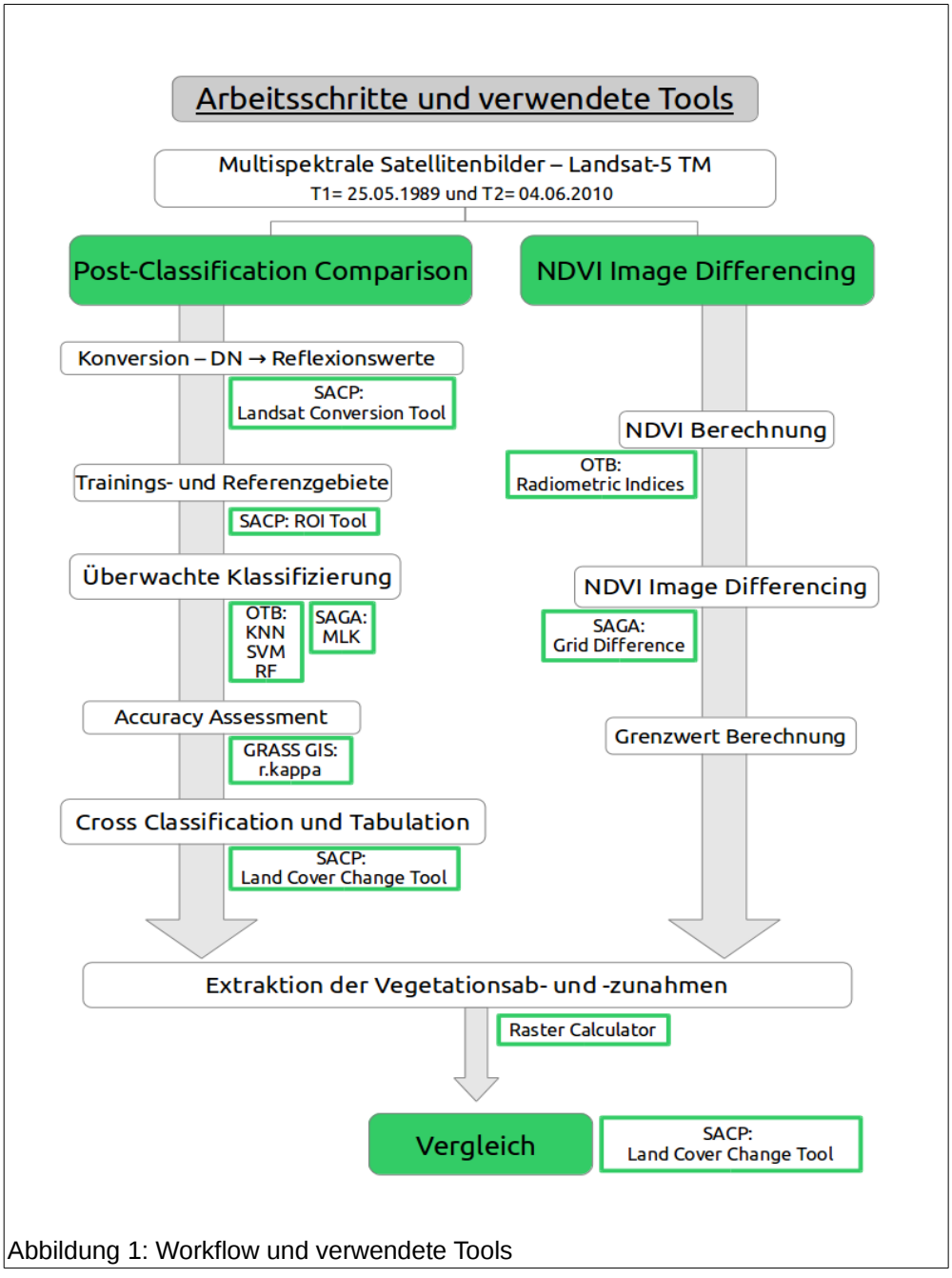



Abbildung 1: Workflow und verwendete Tools

**Erfassung von Landnutzungsveränderungen mit FOSS Image Processing Tools – Change Detection mit QGIS**

Möglichkeiten der Free & Open Source Image Processing Tools für Fernerkundungsanalysen im Kontext von Landnutzungs- und -bedeckungsveränderungen aufzuzeigen und andererseits zwei gängige Change Detection Verfahren hinsichtlich der quantitativ Erfassung von Veränderungen der Vegetation im Untersuchungsgebiet (Köln/Bonn) zu evaluieren.

Für das Post-Classification Comparison wurde das QGIS Plugin Semi-Automatic Classification Plugin (SACP) [1] sowie Algorithmen der Orfeo Toolbox (OTB) [2] und GRASS genutzt und für das NDVI Image Differencing wurden Algorithmen der Orfeo Toolbox und SAGA verwendet. Im Untersuchungsgebiet wurden drei Subsets ausgewählt um mehrere Räume mit unterschiedlichen Anteilen und Verteilungen der Landnutzungs- und -bedeckungsklassen zu haben. Für die Klassifizierung der Gebiete wurden vier unterschiedliche Algorithmen (Maximum Likelihood, K-Nearest Neighbors, Support Vector Machines, Random Forest) verwendet und die Ergebnisse anschließend mittels Accuracy Assessment verglichen. Durch eine Gegenüberstellung von Klassifikationen eines Gebiets zu zwei unterschiedlichen Zeitpunkten konnten die Konversionen bzw. Modifikationen von einer Klasse zu einer anderen Klasse pixelweise quantifiziert werden. Somit konnten Veränderungen der Vegetation detektiert und mit den Ergebnissen der NDVI Image Differencing Analyse verglichen werden. Der Vergleich der erfassten Vegetationszu- und -abnahmen zeigte, dass beide Change Detection Verfahren für jedes Subset unterschiedliche Ergebnisse liefern.

Kontakt zum Autor:

jakob.tworek@googlemail.com

## Weblinks

[1] http://fromgistors.blogspot.com/p/semi-automatic-classification-plugin.html

[2] http://www.orfeo-toolbox.org/packages/OTBSoftwareGuide.pdf

# MTSatellite: Echtzeit-Webmapping für Minetest-Welten

Sascha L. Teichmann

**MTSatellite** [1] ist ein Webmapping System für das Spiel **Minetest** [2], das es erlaubt zeitnah zum Spielgeschehen bereits eine Kartendarstellung online zu haben. MTSatellite ist ein Freizeit-Projekt, entstanden aus Spaß und Freude an Freier Software.

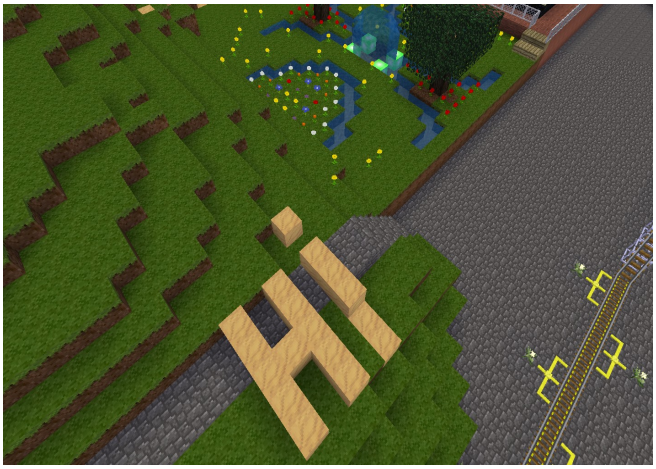


Abbildung 1: Im Spiel

Minetest ist ein 3D- Openworld/Sandbox Spiel, das stark vom bekannten Minecraft [3] oder dem älteren InfiniMiner inspiriert wurde. In einer großen, begehbaren, prozedural generierten 3D-Welt bestehend aus Würfeln (oft etwas ungenau als Voxel bezeichnet) der gedachten Kantenlänge von einem Meter, können die Spieler deren Anordnung und Zusammenstellung nahezu beliebig verändern.

So ist es möglich, Gebäude, Straßen oder mit entsprechenden Erweiterungen (Modifications, Mods genannt) sogar Maschinen zu erstellen, eine einfache Form von Landwirtschaft zu betreiben oder - wenn man dies denn will – gegen künstliche Monster oder menschliche Gegner anzutreten. Der konstruktive Gedanke steht aber klar im Vordergrund und gerade in der Zusammenarbeit mit anderen Mitspielern entstehen teils beeindruckende Monumente.

Hierzu werden Ressourcen benötigt, die von den Spielern teils ober- teils untertage (daher der Namenspräfix "Mine") abgebaut werden können. Des weiteren können diese dann veredelt und zu neuen Werkstoffen kombiniert werden: Mittels geschürfter Kohle und Eisenerz wird durch Verschmelzung Eisen, das zusammen mit Holzstäben, die ihrerseits aus Brettern, welche wiederum aus Baumstämmen entstehen, zu Spaten geformt (crafting) werden kann.

Minetest ist im Gegensatz zu Minecraft Freie Software bzw. Open Source. Es versteht sich des weiteren eher als Game-Engine, die mithilfe von Mods zu vollwertigen Spielen ausgebaut werden kann. Dies führt gerade bei Umsteigern von Minecraft immer wieder zu Irritationen und zu enttäuschten Erwartungen.

Der Kern von Minetest ist in C++ geschrieben, die Erweiterungen werden in der Skriptsprache Lua verfasst. Dies soll im Gegensatz zu im Java geschriebenen Minecraft auch die Ausführung auf leistungsschwächeren Maschinen erlauben. Als 3D-Framework wird in Minetest die Freie 3D-Engine Irrlicht [4] verwendet, die auch in bekannten Spielen wie SuperTuxKart Einsatz findet.

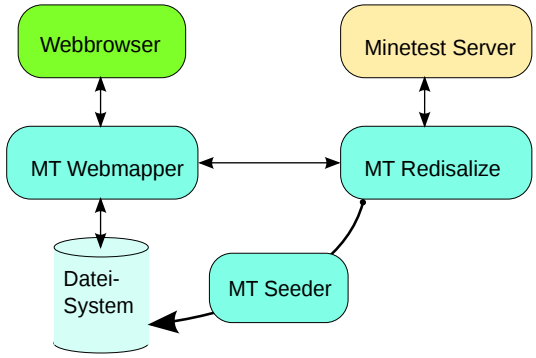
Minetest erfreut sich im Kollegenkreis einer gewissen Beliebtheit. Mit einer Gruppe von ungefähr 5-10 Leuten betreiben wir einen eigenen Minetest-Server. Im Laufe der Zeit kam zusehends der Wunsch auf, die Dinge die erschaffen wurden, auf einer Karte darzustellen. Einer der Kollegen kartierte die Änderungen und stellte sie dann in unregelmäßigen Abständen die resultierenden Karten in Form von PNGs den anderen Spielern zur Verfügung. Dem explorativen Forschergeist der Mitspieler war es aber dann zu verdanken, dass diese Bilddateien alsbald Größen annahmen, mit denen Standardbildbetrachter ihre Probleme bekamen. Der erste Einsatz von GIS-Tools wurde notwendig. Zuerst in Desktop-Systemen wie QGIS dann in ersten Varianten web-basiert mit Leaflet, wobei die große Karte

mittels GDAL in Kacheln zerlegt und mit einer zusätzlichen Tile-Pyramide versehe wurde, um eine elegantere Navigation auf der Karte zu bekommen.

Dies geschah in einem Offline-Prozess. Der Spiel-Server musste angehalten und die Karten generiert werden. Die Frage kam auf, ob man sich nicht in den laufenden Spielbetrieb einklinken und quasi direkt ohne Verzögerung die Änderungen in eine Online-Kartenform bringen könnte. Der Spielbetrieb sollte dadurch nicht beeinflusst werden. Die existenten Kartengeneratoren (genannt Mapper) boten dieses Feature nicht. Dies war die Geburtsstunde von MTSatellite.

Minetest verwaltet seine Raumdaten in einem Key/Value-Store. Die Schlüssel sind zusammengesetzte 3D-Koordinaten von sogenannten Blöcken, wobei die Blöcke die Werte darstellen. Blöcke sind jeweils Raumabschnitte von 16x16x16 1m-Würfeln (genannt Nodes). Sie enthalten neben der geometrischen Form auch die Sachdaten wie etwa das Material (z.B. Sand), aus dem sie bestehen. Die komplette Welt hat in jede Raumrichtung eine begehbare Ausdehnung von ungefähr 63 Kilometern.

Standardmäßig verwendet Minetest eine SQLite-Datenbank für diesen Key/Value-Store. Es ist allerdings auch möglich, hier eine LevelDB [5] oder einen Redis-Server [6] zu benutzen. An dieser Stelle greift MTSatellite transparent in den Prozess ein. MTSatellite implementiert einen Redis-Server, der für den Minetest-Server als Backend konfiguriert wird. Genauer gesagt implementiert die Komponente MT Redisalize den für den Betrieb von Minetest nötigen Subset von Redis-Kommandos. Nebenher werden Modifikationen der Daten mitprotokolliert, so dass man auf dieser Basis effizient existente Karten aktualisieren kann. Als echtes Backend wird eine LevelDB benutzt, die gegenüber dem Original ein Index-Schema in Form von einer 3D-Z-Kurve [7] besitzt. Das Redis-Protokoll wird so erweitert, dass von außen räumliche Anfragen an den MT-Redisalize-Server gestellt werden können. Die Konversion einer existenten Datenbank in diese Form kann mit dem Tool MT- DB-Converter durchgeführt werden.




Architektur von MTSatellite 9.2.2015

Selbst wenn auf dem Spiel-Server viele Spieler gleichzeitig aktiv sind, ist in der Summe betrachtet die zu erstellenden Karte relativ statisch. Aus diesem Grund wird die Karte initial einmal mit der Offline-Komponente MT Seeder vorberechnet und dann on-the-fly von der Komponente MT Webmapper aktualisiert. Diese bekommt in konfigurierbaren Zeitintervallen die aggregierten Koordinaten-Änderungen vom MT Redisalize-Server übermittelt. Unter Einsatz einiger Algorithmen und Heuristiken (im Kern intelligentes Z-Puffern) generiert dieser dann kolorierte Relief-Kacheln der Größe 256x256. Jeder Pixel repräsentiert hierbei einen Würfel in der 3D-Welt. Diese Kacheln werden zudem mit einer Zoom-Pyramide versehen. Die höheren Zoom-Stufen werden on-demand vom MT Webmapper berechnet, da sich hier eine Vorgenerierung und der damit verbundene Platzbedarf im Vergleich mit den Kosten für die Neugenerierung nicht rentieren.

Als Client benutzt MTSatellite Leaflet [8]. Diese Javascript-Bibliothek konnte nahezu ohne Änderung sofort verwendet werden. Nur eine neue Projektion musste ergänzt werden, da das Minetest-Weltkoordinaten-System nur sehr eingeschränkt Ähnlichkeit mit einem Echtwelt-System hat.

Das MTSatellite-System ist in Go [8] implementiert. Go ist eine 2009 veröffentlichte Programmiersprache aus der Feder von Rob Pike, Ken Thompson und Robert Griesemer, die vor allem durch ihre sprachliche Einfachheit im Hinblick auf systemnahe, verteilt skalierende und nebenläufige Netzwerk-Dienste entworfen wurde. MTSatellite stellt ein Experiment dar, in wie weit sich diese Sprache für

den Einsatz im Bereich der raumbezogenen Daten eignet. Die Erfahrungen sind diesbezüglich durchweg positiv. Gerade die Schlichtheit gepaart mit der praktischen Eleganz bzgl. der Nebenläufigkeit waren bei der Entwicklung von MTSatellite von großem Nutzen. Durch die räumliche Unabhängigkeit der Kacheln einerseits und der Trennung der Dienste andererseits lies sich der Gesamtprozess sowohl horizontal als auch vertikal gut in konkurrierend lauffähige Einheiten zerlegen. So skaliert MTSatellite nahezu linear mit der Anzahl der ihm zugeordneten CPU-Kernen. Eine Aufteilung auf verschiedene physikalische Server zur Lastverteilung ist ebenfalls möglich. Diese Erkenntnis wird sich mit großer Wahrscheinlichkeit auch auf Echtwelt-Anwendungen mit Raumbezug übertragen lassen.

Ausblick: Die Server-seitige Infrastruktur insbesondere der MT-Redisalize-Server eröffnen eine Reihe an Möglichkeiten, die über das Rendern von 2D-Kacheln hinausgehen. So wäre es etwa denkbar, einen 3D-Viewer auf Basis von WebGL zu bauen oder Tools, die Analysen sowohl auf den räumlichen als auch auf den Sachdaten durchführen. Aber auch ohne diese Erweiterungen erfreut sich MTSatellite einer kleinen Schar von Nutzern auf GNU/Linux-Systemen, bei denen ich mich an dieser Stelle ausdrücklich bedanken möchte.

Ein besonderer Dank geht an meine Kollegen von der Intevation GmbH [9], die mich bei der Erstellung und beim Betrieb von MTSatellite tatkräftig unterstützt haben.

Kontakt zum Autor:

Sascha L. Teichmann
Intevation GmbH
Neuer Graben 17
49074 Osnabrück
sascha.teichmann@intevation.de

Literatur

[1] MTSatellite: https://bitbucket.org/s_l_teichmann/mtsatellite

[2] Minetest: http://minetest.net/

[3] Minecraft: https://minecraft.net/

[4] Irrlicht: http://irrlicht.sourceforge.net/

[5] LevelDB: https://github.com/google/leveldb

[6] Redis: http://redis.io/

[7] Z-Kurve: http://de.wikipedia.org/wiki/Z-Kurve

[8] Go: http://golang.org/

[9] Intevation GmbH: http://intevation.de/

# PostGIS Memento

## Versionierung von PostGIS-Datenbanken

Felix Kunde

Memento. Gibt es nicht einen Film, der so heißt? Jemand, der kein Gedächtnis hat und sich alle Ereignisse aufschreibt? Zumindest geht es bei pgMemento darum. pgMemento zeichnet alle Veränderungen in einer PostgreSQL Datenbank auf und erlaubt die Wiederherstellung beliebiger frühere Zeitstände.

pgMemento ist ein neues Projekt, dass einen Ansatz Datenbanken zu versionieren vorschlägt. Alle Veränderungen werden mittels Triggern in einer Log-Tabelle als JSON-Fragmente aufgezeichnet. Durch den Einsatz von JSON ist es egal, wie die Tabelle beschaffen ist und ob sie sich über die Zeit strukturell verändert hat. Mittels einer zusätzlichen ID-Spalte wird der Bezug zu allen Versionen einer Zeile hergestellt. So können beliebige frühere Zeitstände eines Tabellentupels ganz einfach wieder hergestellt werden. Mit den JSON-Funktionen von PostgreSQL können auch komplexe Datentypen wie PostGIS-Geometrien verarbeitet werden. Das Erstellen von früheren Versionen der Datenbank erfolgt in separaten Schemata und nicht in der Produktionsdatenbank selbst.

Wen ein Blick in die Vergangenheit der Datenbank interessiert, der stößt vielleicht auch auf Informationen, die im Zuge eines Anwenderfehlers irgendwann einmal verloren gegangen aber eigentlich noch gültig sind. Da pgMemento zu jedem Log-Eintrag auch stets die dazugehörige Transaktion protokolliert, könnten alle Veränderungen einer Transaktion wieder rückgängig gemacht werden. Seien es einfache Update-Befehle oder Löschvorgänge. Das komplette Zurücksetzen der Datenbank zu einem bestimmten Zeitstempel wäre also nicht notwendig. Dieses Feature muss jedoch erst noch entwickelt werden.

pgMemento ist komplett in PL/pgSQL programmiert, d.h. die Versionierungslogik ist für jeden Nutzer transparent und es kann sehr einfach in bestehende Datenbanken eingebunden werden.

# Automatisiertes Geodatenmanagement mit GeoKettle

Jens Schaefermeier

Dieser Vortrag stellt verschiedene Einsatzmöglichkeiten der freien ETL-Software GeoKettle vor. GeoKettle ist die Open Source-Alternative zur verbreiteten Software "FME" und kann nicht nur Geodatenformate konvertieren, sondern beispielsweise auch Objekte verteilen und zusammenfassen, redundante Daten finden oder Prozesse in einer grafischen Oberfläche modellieren.

GeoKettle GeoKettle ist ein ETL-Programm für räumliche Daten. ETL steht für Extract, Load und Transform. GeoKettle basiert auf der OpenSource Software Pentaho Data Integration (Kettle) und ist mit der LPGL lizensiert. GeoKettle unterstützt dabei u.a. die OpenSource Bibliotheken GeoTools, Degree und gdal/ogr und sextante. Es kann als OpenSource-Alternative für die FME eingesetzt werden und bietet vielfältige Einsatzmöglichkeiten. Während des Vortrags werden einige Einsatzmöglichkeiten und Funktionen von GeoKettle vorgestellt, um dem Auditorium einen Einblick in die Leistungsfähigkeit von Geokettle zu geben. Zu den Funktionalitäten gehören: Import verschiedener (Geo-)Datenformate Verteilen von Objekten in einem Shapefile auf mehrere Tabellen Zusammenfassen von Objekte aus mehreren Shapefiles mit unterschiedlichen Attributfeldern in einer Tabelle Veränderung von Attributen Entfernen von redundanten Daten Benutzten des graphischen Benutzeroberfläche zur Modellierung von Prozessen. Benutzen der Shellskripte zur automatisierten Verwendung von GeoKettle über cronjobs ...

# Straßenrennen in der Innenstadt

## SuperTuxKart-Strecken bauen mit OSM2World

Tobias Knerr

Der Gründer des Open-Source-Projekts OSM2World erzählt von einer weiteren spannenden Anwendungsmöglichkeit des 3D-Renderers: Die Erstellung von Rennstrecken für das Computerspiel SuperTuxKart.

Seit es die Möglichkeit gibt, 3D-Daten in OpenStreetMap zu erfassen, wird gerne auch die Verwendung in Computerspielen als Anwendungsfall genannt. Einige kommerzielle Beispiele dazu gibt es schon länger; dass dies aber auch mit Open-Source-Tools möglich ist, wird hier mit der Erstellung von Rennstrecken für das freie Spiel SuperTuxKart unter Beweis gestellt.

Der Vortrag beschreibt die Vorgehensweise, um die OSM-Daten einer Stadt in eine Rennstrecke zu verwandeln. Voraussetzung ist eine ausreichend gute Abdeckung der jeweiligen Region mit 3D-Attributen in OpenStreetMap. Solche Daten, insbesondere zu Gebäuden und Straßen, werden von der Community zunehmend erfasst. Aus diesen Daten wird mit dem 3D-Renderer OSM2World ein realitätsnahes 3D-Modell erstellt, das optional auch mit einem Geländemodell ergänzt werden kann. In diesem 3D-Modell werden mit dem 3D-Modelling-Tool Blender noch einige Verbesserungen vorgenommen und der gewünschte Streckenverlauf eingezeichnet. Ein Blender-Plugin erlaubt schließlich den Export einer fertigen Strecke für SuperTuxKart.

Stolperfallen sind dabei unter anderem subtile Unterschiede bei der Interpretation der Daten zwischen den beteiligten Anwendungen. Auf solche technischen Hürden wird dabei ebenso eingegangen wie auf neue Features in OSM2World, die diesen Anwendungsfall möglich gemacht haben.

### Links

- http://osm2world.org/
- http://supertuxkart.sourceforge.net/

# Docker für den GIS-Einsatz

Björn Schilberg

Mit der leichtgewichtige Virtualisierungs-Software Docker schnell und einfach Anwendungs-Container für Ihre GIS-Fachanwendung erstellen. Haben die GIS-Admins demnächst wieder mehr Zeit für ungleich spannendere Aufgaben als die Installation von Fachanwendungen?

Die leichtgewichtige Virtualisierungs-Software Docker [1] ist zur Zeit in aller Munde. Mit Docker lassen sich Anwendungen samt ihrer Abhängigkeiten in Container verpacken, in denen sie sich später leicht weitergeben und ausführen lassen. ImVergleich zu virtuellen Maschinen sollen Docker-Container sparsamer im Umgang mit Ressourcen sein und schneller starten können. Doch was bedeutet Docker für den GIS-Einsatz oder die GIS-Entwicklung? Fallen bald die 100seitigen Installationshandbücher für die Installation von Fachanwendungen weg? Haben die GIS-Admins demnächst wieder mehr Zeit für ungleich spannendere Aufgaben als die Installation von Fachanwendungen? Welche Vorteile ergeben sich durch die Anwendungs-Container? Wann lohnt sich der Einsatz von Anwendungs-Container und wann sollte man lieber zur Virtualisierungs-Software wie VirtualBox und Co. greifen?

All diese Fragen werden im Vortrag auf der FOSSGIS 2015 in Münster beleuchtet und mit der Erstellung eines Anwendungs-Container für die Breitstellung eines WMS-Dienstes abgerundet.

### Links

- [1] Docker - Build, Ship and Run Any App, Anywhere: https://www.docker.com/

# WPS, GeoServer und SHOGun

Till Adams

Der Vortrag stellt die Erweiterung des SHOgun Frameworks als WPS-CLient dar. Als WPS Server kommt der GeoServer-WPS zum Einsatz. Der Vortrag wird zum einen kurz die Mechanismen des WPS erläutern, die Einbindung in das SHOGun Framework an praktischen Beispielen zeigen und am Ende die Möglichkeiten, die sich daraus für ein WebGIS ergeben vorstellen.

SHOGun ist ein OpenSource WebGIS Framework, das bereits auf den letzten FOSSGIS Konferenzen vorgestellt wurde.

In der derzeitigen Version bietet SHOGun die Möglichkeit aus einer Installation Layer aus verschiedenen Kartendiensten, WebGIS-Oberflächen und Benutzer zu verwalten, WMS und WFS-Dienste abzusichern sowie darüber hinaus viele Webschnittstellen über Mittel des Frameworks Java Spring zur Verfügung zu stellen. SHOGun ist somit eine mächtige, datenbankunabhängige Middleware, die in großen Verwaltungen ein komplettes GIS ersetzt.

Der WebGIS-Client zeichnet sich durch eine Fülle an Funktionen aus, die weit über den normalen Funktionsumfang eines WebGIS hinaus gehen. Dennoch erfordert jede funktionale Erweiterung bisher eine entsprechende Programmierung in SHOGun. Aus diesem Grunde wurde für die Wasserwirtschafts-Verwaltung Rheinland-Pfalz SHOGun um eine Web Processing Service (WPS) Client-Schnittstelle erweitert. Da SHOGun ohnehin in der Lage ist, GeoServer über seine REST-API anzusprechen, wurde auch der GeoServer WPS verwendet. Über die Verwaltungsoberfläche von SHOGun lassen sich einzelne oder verkettete WPS-Prozesse in einen WebGIS-Clienten einbinden. Damit ist eine funktionale Erweiterung mittels Konfiguration über die Oberfläche möglich. Zur Ergebnisverarbeitung stellt SHOGun wiederum verschiedene Methoden und Funktionen bereit. Selbstverständlich hat die Implementierung Einschränkungen, dies alleine aufgrund der de facto unbegrenzten Möglichkeiten, die die WPS-Spezifikaiton bietet.

Der Vortrag stellt das Prinzip eines WPS vor und die Umsetzung innerhalb von SHOGun, zeigt Stärken, Potentiale, aber auch Schwächen oder mögliche Begrenzungen auf.

# Prokitektura: prozedurale realistische 3D Gebäude und Städte

Vladimir Elistratov

Prokitektura is ein prozedurales und iteratives Verfahren für Schaffung architektonischer 3D Modelle. Ein Satz kleiner Funktionen auf Python wird verwendet um 3D Gebäude zu generieren.

Man nutzt 3D Elemente oft in heutigen GIS-Anwendugen. Aber manuelle Schaffung der 3D Gebäude ist anstrengend. Prokitektura bietet eine Alternative dafür. Prokitektura is ein prozedurales und iteratives Verfahren für Schaffung architektonischer 3D Modelle.

Ein Satz kleiner Funktionen auf Python wird verwendet um 3D Gebäude zu generieren. Eine solche Funktion nennt man Regel. Jede nachfolgende Regel verfeinert das Modell und ergänzt es mit zusätzlichen Details. Zur Zeit ist Prokitektura als ein Add-on für Open Source 3D Plattform Blender realisiert.

Hier ist eine kurze Beschreibung wie man ein einfaches 3D Gebäude mit Prokitektura schafft. Man startet mit einem Gebäudeumriss den man aus einem üblichen GIS-Format wie OSM, GeoJSON importieren kann. Eine Extrusion mit passender Höhe wird erzeugt. Das extrudierte 3D Objekt wird in eine Anzahl der senkrechten Rechtecke und das obere Vieleck dekomposiert. Jedes senkrechte Rechteck entspricht einer Gebäudefassade die in Stockwerke aufgeteilt wird. Aus dem oberen Vieleck wird ein Dach erstellt. Jede Etage wird in Sektionen mit Fenstern aufgeteilt. Jede Sektion kann weiter verfeinert werden. Am Ende wird das 3D Modell ins gewünschte Format exportiert.

Um einen bestimmten Bebauungstyp in einer Stadt zu simulieren, ändert man gewisse Größen in jedem Gebäude zufällig und verwendet stochastische Verfahren um eine Variante aus mehreren Gebäudeteilen auszuwählen.

**Links**

- Source Code und Tutorial für Prokitektura: https://github.com/vvoovv/prokitektura-blender
- Webseite des Projekts: https://github.com/vvoovv/prokitektura-blender

# GeoCouch

## Ein multidimensionaler Index für Couchbase

Volker Mische

GeoCouch ist ein multidimensionaler Index für Couchbase, einer verteilten dokumentenorientierten Datenbank. Dieser Vortrag wird eine kurze Einführung in Couchbase, die räumliche Abfragen ermöglicht, und die Datenmodellierung geben und dann in eine Live-Demonstration übergehen.

Couchbase ist eine verteilte dokumentenorientierten Datenbank und gehört damit zur Kategorie der NoSQL Datenbanken. Das bedeutet, dass das Datenmodell nicht relational ist, also ein gewisses Umdenken bei der Strukturierung der Daten nötig ist. GeoCouch ist in Couchbase integriert und ermöglicht räumliche Abfragen. Diese sind nicht auf zwei Dimensionen begrenzt, sondern können durch weitere Attribute erweitert werden. So ist es möglich Faktoren, wie Zeit, Ausmaß oder Kategorien einzubeziehen. Eine beispielhafte multidimensionale Datenbankabfrage wären alle Einwohner eines bestimmten Gebiets, die unter 30 Jahre alt sind und in einer Mietwohnung mit weniger als 50m² wohnen. Dieser Vortrag wird eine kurze Einführung in Couchbase und die Datenmodellerung geben und dann in eine Live-Demonstration übergehen. Couchbase ist Open-Source und unter der Apache Lizenz 2.0 lizenziert.

# Mapbender3 für den einfachen Aufbau von WebGIS Anwendungen

So einfach ist der Aufbau von WebGIS Anwendungen mit Mapbender3

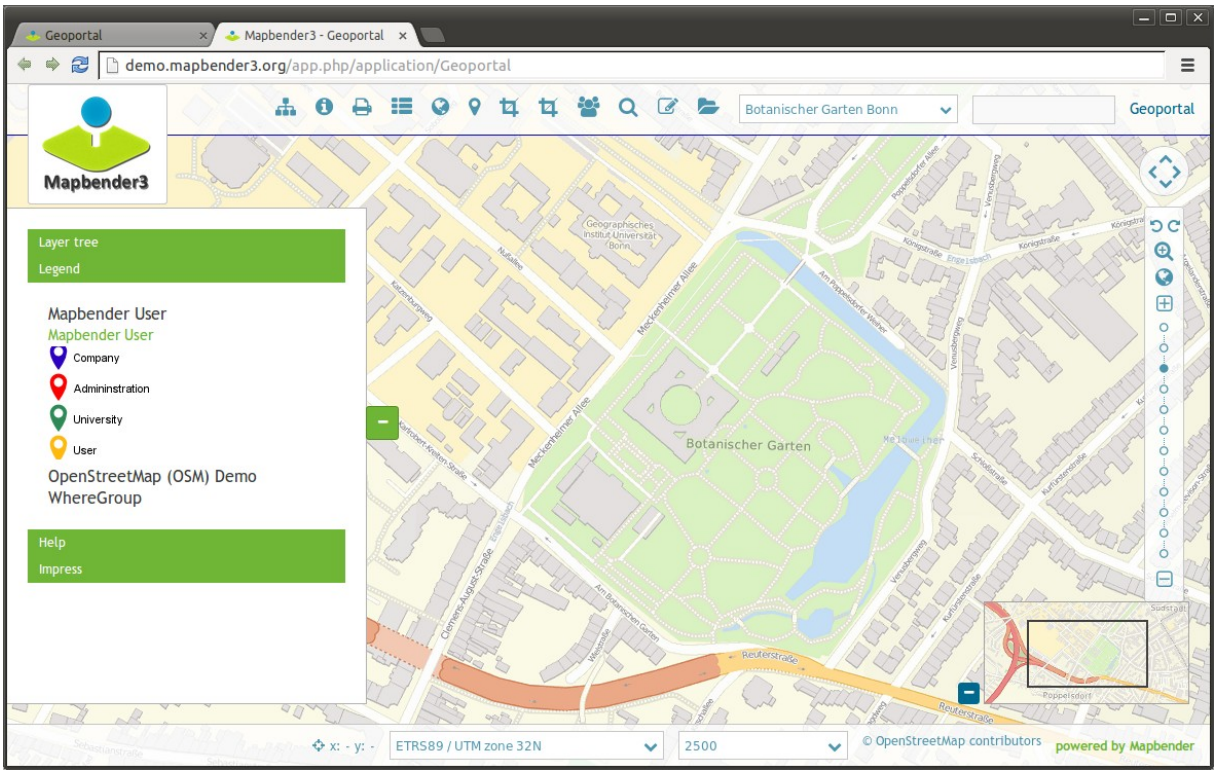
Astrid Emde

Mapbender3 [1] ist eine Software zur einfachen Erstellung von WebGIS Anwendungen. Über ein paar Klicks können Anwendungen erstellt und mit Kartendiensten bestückt werden. Die Anwendungen können öffentlich bereitgestellt oder nur bestimmten Benutzern oder Gruppen zur Verfügung gestellt werden.

Basierend auf dem Symfony Framework wurde eine moderne Webanwendung geschaffen, die das Baukastensystem der Bundles nutzt. Derzeit kommen die Symfony Version 2.3 und als Kartenkomponente OpenLayers in der Version 2.13 zum Einsatz. Mapbender3 nutzt außerdem MapQuery als Brücke zwischen OpenLayers und der jQuery Welt. Mapbender3 speichert die Daten der Konfiguration in einer Datenbank. Hier können eine ganze Reihe von Datenbanksystemen verwendet werden, z.B. PostgreSQL, MySQL, Oracle oder SQLite.

Zusammen mit einem modernen Verwaltungsbackend für die Kartenanwendungen ist Mapbender3 ein komfortables Werkzeug für die Erstellung und Pflege von Kartenanwendungen.

Auf der FOSSGIS 2015 soll die Version 3.0.4 vorgestellt werden, die im Herst 2014 veröffentlicht wurde (siehe Roadmap [2]). Mit der Version kam es zu einem Lizenzwechsel zur MIT-Lizenz, die als eine der nutzerfreundlichsten OpenSource-Lizenzen angesehen wird. Der Schwerpunkt der Version lag auf der Verbesserung der Bedienung. Es wurde ein CSS-Editor integriert, über den leicht Anpassungen der Anwendungen erfolgen können. Über das neue HTML-Element können Texte, Bilder und Links integriert werden.



### Mapbender3 - das Frontend

Der Vortrag stellt zuerst das Frontend mit seinen Komponenten vor und geht dabei auf die neuen Funktionen ein.

- Druck mit Übersichtskarte, Maßstabsleiste, Übernahme der Transparenz sowie von Messungen und Treffern
- Erweitertes Kontextmenü mit Transparenzregler, Metadaten, Zoom aus Ebene
- HTML-Element zur Integration von Texten, Bildern und Links
- CSS-Editor zum Anpassen der Oberfläche
- Buttons zum Wechseln des Hintergrunds
- Bereitstellung und Auswahl unterschiedlicher Templates
- Aufbau einer Reiterstruktur
- WMC Editor, WMC Loader, WMC Auswahl, Karte weiterempfehlen
- Internationalisierung (englisch, deutsch, italienisch, spanisch)
- Mapbender3 mobil

Eine Übersicht der Elemente findet sich in der Mapbender3-Dokumentation [3].

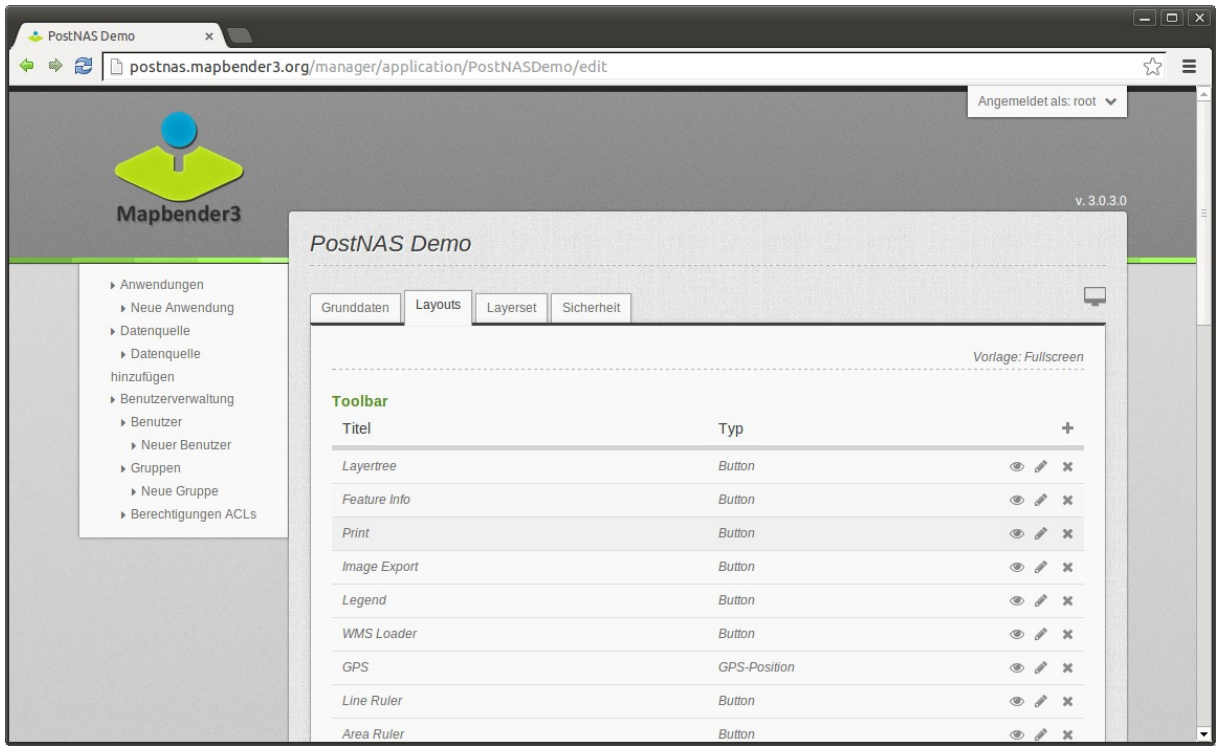
### Mapbender3 – das Administrations-Backend

Mapbender3 verfügt über ein Administrations-Backend, das nur berechtigten Benutzern zur Verfügung steht. Über das webbasierte Backend kann einfach über ein paar Klicks eine neue Anwendungen erstellt werden. Eigene Anwendungen können mit Diensten und den gewünschten Elementen bestückt werden. Die einzelnen Elemente bieten elementspezifische Konfigurationsmöglichkeiten. Darüber hinaus können Funktionalitäten nur für bestimmte Benutzer freigegeben werden. Zu jeder Anwendung kann ein Vorschaubild hochgeladen werden.

Das Administrations-Backend beinhaltet die Möglichkeit, ein Dienste-Repository aufzubauen. Derzeit werden WMS 1.1.1 und WMS 1.3.0 als Datenquelle im Repository unterstützt. Weitere von OpenLayers unterstützte Datenquellen wie WMTS und WFS sollen hinzukommen.

Mapbender3 bietet eine Benutzer- und Gruppenverwaltung mit der Möglichkeit Rechte zuzuweisen. Hier kommen die Access Control Lists (ACLs) von Symfony2 zum Einsatz.

Anwendung können seit der Version 3.0.4.0 als JSON oder YAML exportiert und in andere Installationen importiert werden, was den Austausch von Anwendungen erleichtert.




Testen Sie Mapbender3 selbst! Die Installation der Software ist sehr einfach [4]. Sie können Mapbender3 auch online testen. Hierzu stehen Demos [5] zur Verfügung. Ein Quickstart-Dokument ermöglicht es Ihnen in einfachen Übungen Mapbender3 kennen zu lernen [6]. In der Mapbender3 Galerie können Sie Mapbender3 Anwendungen im Einsatz sehen [7].

Kontakt zur Autorin:

Astrid Emde
WhereGroup GmbH & Co. KG
Eifelstraße 7
53119 Bonn
+49 (0)228 909038-0
astrid.emdewheregroup.com

Literatur

[1] Mapbender3 Webseite http://mapbender3.org

[2] Roadmap http://mapbender3.org/roadmap

[3] Dokumentation (englisch / deutsch) http://doc.mapbender3.org

[4] Download http://mapbender3.org/download

[5] Demos http://demo.mapbender3.org

[6] Quickstart Tutorial http://doc.mapbender3.org/de/book/quickstart.html

[7] Mapbender3 Galerie http://mapbender3.org/de/galerie

# Die neue WebGL-basierte Plattform für OSM-3D

HONGCHAO

In diesem Vortrag wird die technische Archtektur, das Funktiondesign und die Implementierung der neue WebGL-basierte Plattform von OSM-3D präsentiert. Es werden interaktive Visualisierungen der 3D Stadtmodelle und 3D Geländemodelle demonstriert. Das neue Interface für interaktive Erfassungen von Dach- und Fassadenstrukturen wird vorgestellt und mit Beispielen demonstriert. Diskussion von Vorschlägen zur Verbesserungen der neuen Plattform.

In den letzten Jahren hat sich „Volunteered Geographic Information“ (VGI) rasch entwickelt. Der Detailreichtum im OpenStreetMap (OSM) Projekt wächst stetig. Besonders in urbanen Räumen werden zunehmend Gebäude und weitere Objekte wie Straßenmöbel aufgenommen. Die neueste Statistik zeigt, dass es bis zum 25.11.2014 über 200 Millionen Gebäudegrundrisse in OSM gab. Davon ist Deutschland mit 18107565 Gebäudegrundrissen weltweit das meisten gezeichnet Land hinsichtlich des Gebäudes. Seit einigen Jahren, entstehen einige Projekte wie z.B. OSM2World, Kendzi3D, F4map, OSM-3D, die 3D-Stadt- und Landschaftsmodelle aus OSM-Daten erstellen. Dafür werden hauptsächlich die Gebäudegrundrisse extrudiert. Des Weiteren werden vereinzelte 3D-bezogene Informationen genutzt, die in OSM bereits vorliegen, z. B. Gebäudehöhen, Dachtypen, usw. OSM-3D stellt die kreierten Modelle standardisiert als Web 3D Service zur Verfügung. Eine Client-Software wie z. B. der XNavigator setzt die einzelnen W3DS-Szenen zu einem vollständigen 3D-Globus zusammen. Die Visualisierung und Interoperabilität mit 3D Stadtmodellen in Web Browser ist leider nur mit Hilfe ein Java PlugIn erst realisierbar. Dies muss ständig Fortführungen wegen Update von Java gefordert werden. Darüber hinaus ist PlugIn Technology aus vielen Gründen nicht mehr die Tendenz der Zukunft. Aus dem oben genannten Grund, wird eine neue WebGL-basierte Plattform für OSM-3D an der Universität Heidelberg entwickelt. Die neue Plattform verwendet JavaScript-Bibliothek „Cesium“, die WebGL nutzt und so plattformunabhängige Visualisierungen mit Hardwarebeschleunigung erlaubt. Die sämtliche Funktionen von Xnavigator werden nach der neuen Plattform „übertragen und umgesetzt“. D.h. für die neue WebGL-basierte Plattform werden gleiche Funktionen wie beim Xnavigator entwickelt. Diese passieren an die Client-seite. An die Server-Seite werden die vorhandenen Ansätze zur Gebäudengenerierung, Datenbankstrukturen, sowie W3DS weiter verwendet. Allerdings werden diese Ansätze nach Anförderungen der neuen Plattform zur effiziente interaktive Visualisierung und Editierung verändert. Außerdem werden auch neue Ansätze entwickelt. In dieser Arbeit wird die technische Archtekturen, Funktiondesign, und Implementierung der neue WebGL-basierte Plattform von OSM-3D vorgestellt. Zweitens werden die interaktive Visualisierungen der 3D Stadtmodelle und 3D Geländemodelle demonstriert. Neue Interface für interaktive Erfassungen von Dach- und Fassadenstrukturen wird vorgestellt und mit Beispielen demonstriert. Schließlich werden einige Vorschläge zur Verbesserungen der neue Plattform disskutiert.

# Vector Tiles

## Performante Übertragung von umfangreichen Vektordaten

Johannes Weskamm

Der Vortrag beleuchtet die Vor- und Nachteile von Vectortiling und der Vektordaten-Prozessierung im Client allgemein. Als praktischer Anwendungsfall wird das Prinzip der Vectortiles anhand des Software-Stacks TileStache, OpenLayers3 und PostGIS auf Basis von OSM-Daten vorgestellt.

Die Nutzung von Kacheln bzw. „Tiles“ für Rasterdaten in GIS-Anwendungen ist ein alter Hut. Mit der allgemein steigenden Bandbreite von Internetanbindungen, verbesserten Performance der Clients und auch moderneren Webbrowsern rückt jedoch wieder eine Idee in den Fokus der WebGIS-Welt, die in der Vergangenheit zwar bereits diskutiert wurde, jedoch mangels technischer Voraussetzungen zum Scheitern verurteilt war: Das Laden und Rendern von umfangreichen Vektordaten im Client. Um eine möglichst performante Übertragung an den Client zu erreichen, bietet sich die Nutzung von so genannten Vectortiles an.

Die Vorzüge von Vektoren im Client liegen auf der Hand: Liegen sowohl Geometrien als auch Attribute im Browser vor, so erlaubt dies ein dynamisches Styling im Client, während der Nutzer bei Rasterdaten keinen Einfluss auf die serverseitig gerenderten Kartenansichten hat. Durch den direkten Zugriff auf Geometrien und Attribute ergeben sich ausserdem Funktionen wie Highlighting, Selektion, Attributabfrage oder Filterung, ohne das jedesmal eine Client-Server-Kommunikation vonnöten wäre. Ein Großteil der Prozessierung wird somit vom Server auf den Client ausgelagert.

Der Vortrag beleuchtet die Vor- und Nachteile von Vectortiling und der Vektordaten-Prozessierung im Client allgemein. Als praktischer Anwendungsfall wird das Prinzip der Vectortiles anhand des Software-Stacks TileStache, OpenLayers3 und PostGIS auf Basis von OSM-Daten vorgestellt.

# Erfahrungen mit Sensor Web-Anwendungen für mobile Geräte und Desktop-Browser

Dr. Simon Jirka, Jan Schulte, Dr. Christoph Stasch

Sensor Web-Dienste, insbesondere der OGC Sensor Observation Service (SOS) erlauben die Bereitstellung von Messdaten, sowohl Nah-Echtzeit-Messungen als auch Archivdaten, in Geodateninfrastrukturen [1]. Ebenso besteht die Möglichkeit über SOS-Server, Messdaten entsprechend der INSPIRE-Richtlinie zum Download anzubieten [2]. Typische Anwendungsgebiete sind beispielsweise die Hydrologie, Ozeanographie, Meteorologie, und die Umweltbeobachtung (z.B. Messung der Luftqualität).

Die meisten zurzeit verfügbaren Sensor Web-Anwendungen sind auf die Analyse und Visualisierung von Messdaten in Desktop-Anwendungen (Web Browser oder GIS) ausgerichtet. Allerdings steht im Zuge der immer weiter fortschreitenden Technik ein zusätzliches Spektrum an Endgeräten, von Smart Phones bis hin zu Tablet-Computern, zur Verfügung. Insbesondere mobile Anwendungen, sowohl innerhalb von Organisationen zur Unterstützung der Mitarbeiter als auch extern zur Kommunikation mit der Öffentlichkeit, profitieren von der Verfügbarkeit von Messdaten. Allerdings ist die Pflege von unterschiedlichen Anwendungen für verschiedene Plattformen aufwendig und unwirtschaftlich.

Daher wurde in einem Kooperationsprojekt des Wupperverbands, der belgischen interregionalen Umweltagentur (IRCEL-CELINE) und der 52°North GmbH ein neuer Sensor Web-Client entwickelt, welcher auf JavaScript und verschiedenen JavaScript-Frameworks bzw. -Bibliotheken beruht. Ziel dieser Open Source-Entwicklung ist es, eine Anwendung bereitzustellen, welche einerseits flexibel an die unterschiedlichen Bedürfnisse verschiedener Nutzer angepasst werden kann und gleichzeitig auf der gesamten Bandbreite von Mobiltelefonen bis hin zu Desktop-Browsern nutzbar ist (Responsive Design).

Bei der Entwicklung eines solchen Sensor Web-Clients sind einerseits die unterschiedlichen Bildschirmgrößen und Nutzerinteraktionsmechanismen (z.B. Maus und Touch-Displays) zu berücksichtigen, aber auch die ggf. eingeschränkte Rechenleistung, Speicherkapazität und Internetanbindung mobiler Geräte.

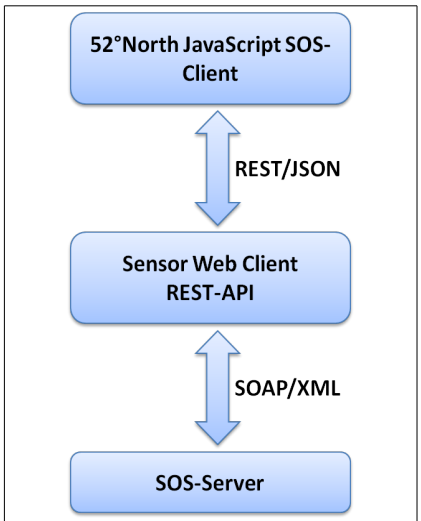


Abb. 1: Schematische Darstellung der System-Architektur

Um mit unterschiedlichen Geräteklassen umgehen zu können wurde HTML/JavaScript als technologische Grundlage ausgewählt. Hierdurch können sowohl Browser auf Desktop-Systemen als auch in mobilen Geräten zur Anzeige genutzt werden. Durch die Verwendung von Bibliotheken und Frameworks wie JQuery, Bootstrap, Leaflet und Flot wurde gleichzeitig der Entwicklungsaufwand minimiert und die Unterstützung einer möglichst großen Bandbreite von Plattformen vereinfacht. Abbildung 2 und 3 zeigen wie der JavaScript SOS-Client auf verschiedenen Typen von Geräten dargestellt wird.

Um das Problem einer eingeschränkten Rechenleistung, Speicherkapazität und Internetanbindung mobiler Geräte zu adressieren, wurde die Geschäftslogik des JavaScript SOS-Clients auf eine serverseitige Komponente ausgelagert: die 52°North Sensor Web Client REST-API [3]. Diese API ist in der Lage Aufgaben wie die Abfrage und Verarbeitung von (großen) XML-Dokumenten des SOS sowie das Caching von Metadaten zu übernehmen und gegenüber dem JavaScript SOS-Client eine leichtgewichtige REST/JSON-Schnittstelle anzubieten. Dadurch kann der JavaScript SOS-Client mit minimalem Datenvolumen und minimaler Anzahl an Abfragen die notwendigen Informationen anfordern, ohne dass umfangreiche XML-Dokumente übertra-

gen und verarbeitet werden müssen. Das zugrundeliegende Architekturkonzept ist in Abbildung 1 schematisch dargestellt.

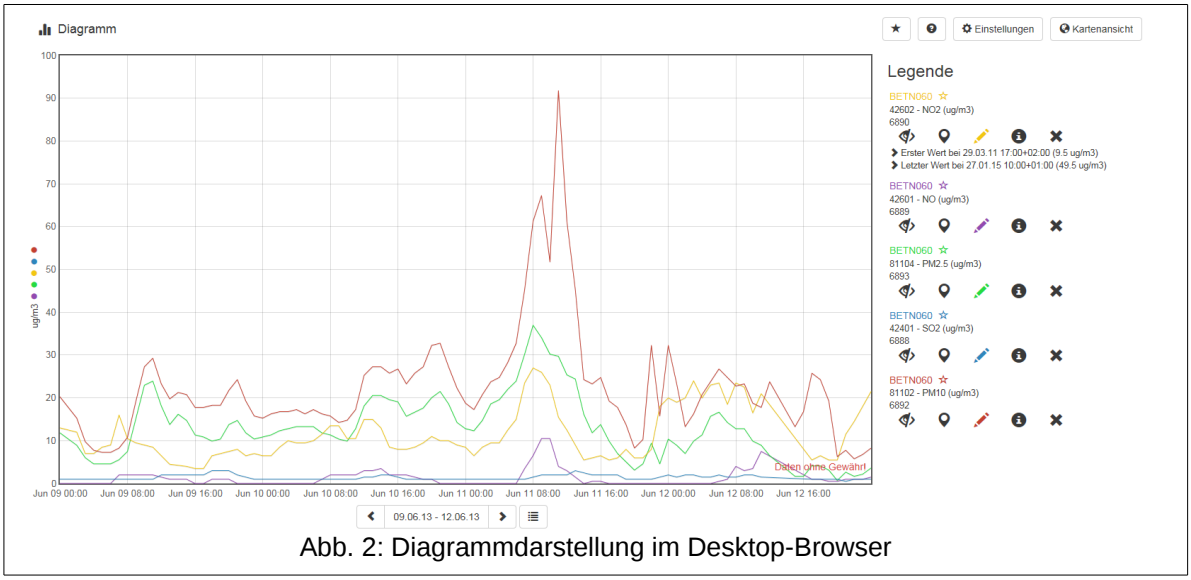



Abb. 2: Diagrammdarstellung im Desktop-Browser

Als Ausblick auf mögliche zukünftige Entwicklungen ist insbesondere die Erweiterung um einen sogenannten Publish/Subscribe-Mechanismus zu nennen. Hierbei erhält der Nutzer eine Oberfläche zur Definition von Regeln, die festlegen, wann eine Benachrichtigung über das Vorliegen bestimmter Messwerte erfolgen soll. Hierüber lassen sich beispielsweise Benachrichtigungen abonnieren, mit denen der Nutzer über kritische Messwerte wie Wasserstand, verschiedene Luftqualitätsparameter oder Windgeschwindigkeit informiert werden kann. Eine vergleichbare Lösung existiert bereits im 52°North Sensor Web-Client auf Google Web Toolkit-Basis. Diese ist jedoch auf Desktop-Browser ausgelegt, so dass eine Integration in den vorgestellten JavaScript SOS-Client eine deutlich größere Nutzergruppe erreichen würde.

Der vorgestellte 52°North JavaScript SOS-Client ist unter einer Open Source-Lizenz (Apache License Version 2.0) verfügbar.

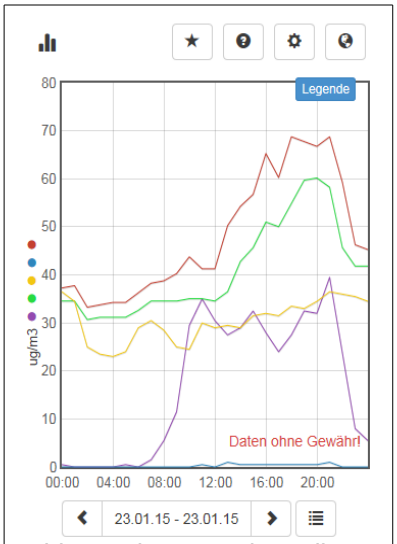



Abb. 3: Diagrammdarstellung auf einem kleinen Display (z.B. Mobiltelefon)

Kontakt zum Autor:

Dr. Simon Jirka
Martin-Luther-King-Weg 24, 48155 Münster
0251-396371-31
jirka@52north.org

Literatur

[1] Arne Bröring, Johannes Echterhoff, Simon Jirka, Ingo Simonis, Thomas Everding, Christoph Stasch, Steve Liang und Rob Lemmens (2011). "New Generation Sensor Web Enablement." MDPI Sensors 11(3): 2652-2699.

[2] Simon Jirka, Alexander Kotsev, Michael Lutz, Matthes Rieke, Robin Smith und Paul Smits (2014). Using the OGC SOS as INSPIRE Download Service for Observation Data. INSPIRE Conference 2014, Aalborg, Dänemark.

[3] Jirka, Simon, Henning Bredel und Jan Schulte (2013). An API For Visualizing Data From The Sensor Web. FOSS4G 2013. Nottingham, United Kingdom.

# Öffentliche Projekte und Open Source

## Open Source Software in der öffentlichen Verwaltung am Beispiel der GDI-DE Testsuite

Sven Böhme

Anhand der GDI-DE Testsuite wird dargestellt, wie die GDI-DE die Strukturen von Open Source Projekten, wie sie u.a. durch die OSGeo beschrieben werden, in Ihren Komponenten verwendet. Betriebsstelle GDI-DE im Bundesamt für Kartographie und Geodäsie (BKG) wurde als Schnittstelle zwischen der fachlichen und der technischen Umsetzung geschaffen. Der Vortrag bietet einen Einblick in die tägliche Arbeit der Betriebsstelle. Die Ziele sowie die Hürden im täglichen Betrieb werden vorgestellt.

Im Rahmen der europäischen Richtlinie INSPIRE betreibt die Geodateninfrastruktur Deutschland (GDI-DE) vier zentrale Komponenten, die verschiedene Servicefunktionen, wie die Geodatenrecherche, die Überprüfung der Konformität von Geodaten und Geodatendiensten anbieten. Bei der Entwicklung dieser Komponenten wurden explizit auf den Einsatz von Open Source Software sowie die Lizensierung der entstehenden Produkte als Open Source Software gesetzt. Doch was bedeutet das genau? Welche Ziele werden damit verfolgt? Sollen nur Kosten eingespart werden oder steckt noch mehr dahinter? Und welche Auswirkungen hat diese Entscheidung auf den täglichen Betrieb? Anhand der Komponente GDI-DE Testsuite soll verdeutlicht werden, wie die GDI-DE die Strukturen von Open Source Projekten, wie sie z.B. durch die Open Source Geospatial Foundation (OSGeo) beschrieben werden, nutzt, um die Öffentlichkeit über den aktuellen Stand der Entwicklungen zu informieren, dem Fachpublikum einen Plattform zum Wissensaustausch zu bieten und gleichzeitig international Maßstäbe zu setzen. Mit der Einrichtung der Betriebsstelle GDI-DE im Bundesamt für Kartographie und Geodäsie (BKG) wurde eine Schnittstelle zwischen der fachlichen und der technischen Umsetzung geschaffen, die neben dem technischen Betrieb der Komponenten auch die Einrichtung und Wartung zusätzlicher Produkte zur Kommunikation, wie Mailing-Listen, Bugtracker etc. gewährleistet. Der Vortrag bietet einen Einblick in die tägliche Arbeit der Betriebsstelle.

# Projekt Umweltzone

## zwischen Open Data und OpenStreetMap

Tobias Preuss

In diesem Vortrag wird das Projekt "Umweltzone" vorgestellt. Dabei handelt es sich um eine kostenlose Open-Source-Anwendung für Android-Geräte, mit der sich Benutzer über die Lage der Umweltzonen verschiedener deutscher Städte informieren können. Die Herausforderung bestand in der Recherche, Beschaffung, Digitalisierung bzw. Umwandlung und Integration der Geoinformation zu den Umweltzonen.

Das Projekt "Umweltzone" startete im Oktober 2013 als direkte Reaktion auf das Fehlen eines zeitgemäßen Informationsangebots für mobile Geräte. Zu diesem Zeitpunkt boten deutsche Städte Informationen zu ihren Umweltzonen lediglich in Form schlecht auflösender Bilder, als PDFs mit eingescannten Stadtplänen oder in Geoportalen an. Eine Anwendung für Android-Geräte, die den mobilen Nutzer ansprach, war nicht vorhanden. Diese Lücke sollte die "Umweltzone"-Anwendung schliessen. Der Vortrag dokumentiert, dass schnell klar wurde, dass die eigentliche Programmierung der Anwendung nicht die größte Hürde darstellte. Die Herausforderung bestand in der Recherche, Beschaffung, Digitalisierung bzw. Umwandlung und Integration der Geoinformation zu den Umweltzonen. Die ersten Tage des Projekts waren von Euphorie geprägt: einige Umweltzonen wurden manuell abgezeichnet, um die Geokoordinaten von einem Bild in digitale Verlaufspunkte zu übertragen. Bald darauf versuchte das Projekt mit Open-Data-Anfragen, Daten in den Behörden deutscher Städte zu befreien. Über die Resultate dieser Bemühungen wird in diesem Vortrag berichtet. Nach der Veröffentlichung der ersten Version der Anwendung kam eine neue Datenquelle hinzu: OpenStreetMap. Das Verzeichnis freier Geodaten offenbarte sich als vielversprechende Alternative zu den offiziellen Publikationen deutscher Städte. Der Vortrag berichtet über die Herausforderungen bei der Datenextraktion, Umwandlung und Prüfung. Zudem soll der Zwiespalt zwischen per Crowdsourcing gesammelten Daten und Veröffentlichungen von Behörden diskutiert werden. Weiterhin berichtet der Vortrag über den Verlauf und Erfolg der "Umweltzonen"-Wochenaufgabe im August 2014 in der deutschen OpenStreetMap-Nutzergemeinde. Es werden die Ergebnisse des Hack Weekends zur Umweltzone vorgestellt. Nicht zuletzt sollen die zukünftigen Schritte des Projekts dargestellt werden.

### Links

- Umweltzone Quellcode: https://bitbucket.org/tbsprs/umweltzone
- Umweltzone im Google PlayStore: https://play.google.com/store/apps/details?id=de.avpptr.umweltzone
- Umweltzonen-Wochenaufgabe 08/2014: http://blog.openstreetmap.de/blog/2014/08/wochenaufgabe-umweltzonen-kw-3536-25-08-7-09-2014/
- Hack Weekend Berlin 10/2014: http://wiki.openstreetmap.org/wiki/Berlin_Hack_Weekend_October_2014

# Indoor Routing in Gebäuden des öffentlichen Verkehrs auf Basis von Openstreetmap Daten

Thomas Jakubicka

**Abstract**

In den letzten Jahren sind immer mehr Bestrebungen zu sehen, das Innenleben von öffentlichen Gebäuden kartografisch darzustellen. Gleichzeitig wird es natürlich auch immer interessanter diese zusätzlichen Informationen weiter zu verarbeiten. Ein für die Anwendbarkeit dieser Daten prädestiniertes Feld ist das Routing und die Navigation innerhalb von Gebäuden des öffentlichen Verkehrs. Der Bedarf an genauer, schneller und vor allem individueller Orientierungshilfe ist in solchen Gebäuden besonders hoch.

Wir, die Mentz Datenverarbeitung GmbH (mdv), sind ein Dienstleister für Informationstechnologie im Bereich öffentlicher Verkehr. Die Verwendung von OSM Daten erlaubt es uns, auch bei den räumlichen Daten in den kompletten Prozess von der Datenerfassung über die Datenverarbeitung bis zur Ausgabe involviert zu sein. Unser Ziel ist hier, die bestmögliche Integration der räumlichen Daten und der Datenmodelle des öffentlichen Verkehrs zu erreichen, auf deren Basis das Routing und die Navigation erfolgen soll.

Das Routing und die Navigation innerhalb von komplexen Umsteigebauwerken hat großes Potential, die Nutzung von öffentlichen Verkehrsmitteln, gerade auch für Personen mit speziellen Mobilitätsanforderungen zu vereinfachen. Bisher sind viele der relevanten Gebäude in OSM nur teilweise als Indoor Bauwerk erfasst.

**Einleitung**

In unserem Vortrag möchten wir unser Konzept der Indoor Navigation vorstellen und Erfahrungen und „Best Practices“ präsentieren, die wir bei der Indoor Erfassung von Gebäuden des öffentlichen Verkehrs gemacht haben.

Seit einigen Jahren ist das Navigieren mithilfe von Mobilen Geräten, wie zum Beispiel Mobiltelefonen selbstverständlich geworden. Das Anwendungsfeld für Navigationsanwendungen hat sich somit von der klassischen Auto Navigation hin zur Fußgänger Navigation erweitert. Dieses neue Anwendungsfeld hat eine Anpassung der Navigationsanwendungen auf die speziellen Bedürfnisse von Fußgängern notwendig gemacht. Zusätzlich zu den vielfältigeren Bewegungsmöglichkeiten von Fußgängern (Treppen steigen, alle Straßen verwenden), die berücksichtigt werden müssen, können sich Fußgänger auch innerhalb von Gebäuden aufhalten und bewegen.

Auch das Innenleben (Indoor) von öffentlichen Gebäuden wird seit kurzem in digitalen Karten erfasst und dargestellt. So gibt es in Online Karten wie GoogleMaps oder OpenStreetMap eine wachsende Zahl von öffentlichen Gebäuden, die über Indoor- Daten und -darstellung verfügen.

Für das Routing und die Navigation, die sich auf die elektronische Fahrplanauskunft des öffentlichen Verkehrs bezieht, sind natürlich Umsteigebauwerke des öffentlichen Verkehrs (z.B. Bahnhöfe) von besonderem Interesse. Insbesondere große Gebäude mit mehreren Ebenen, wie zum Beispiel der Münchner Hauptbahnhof, sind für den Benutzer sehr unübersichtlich und es ist daher schwierig ohne Ortskenntnisse die richtige und schnellste Route innerhalb des Gebäudes zu finden. Das Routing und die Navigation innerhalb von solchen Gebäuden steht hier im Vordergrund.

**Zielsetzung**

Das Navigieren für Benutzer des öffentlichen Nahverkehrs innerhalb von komplexen Umsteigebauwerken soll mithilfe einer mobilen Indoor Navigationsanwendung wesentlich vereinfacht werden. Mithilfe der Anwendung soll es nicht nur möglich sein, die optimale Route innerhalb eines Umsteigebauwerks voraus zu berechnen, sondern es soll auch laufend die Position innerhalb des Gebäudes ermittelt werden, um das Routing permanent aktualisieren zu können.

Außerdem sollen die besonderen Anforderungen der Benutzer an die Route, wie zum Beispiel „Rollstuhlgerecht“, in der Anwendung berücksichtigt werden.

Für die Umsetzung der beschriebenen Funktionalität ist das Vorhandensein einer GIS Datengrundlage mit Indoor Daten notwendig. Wir konzentrieren uns für die Umsetzung der Anwendung auf OpenStreetmap (OSM) Daten als Datengrundlage. Die Erfassung von Indoor Daten in OSM ist noch nicht sehr weit verbreitet. Damit die Datenqualität der OSM Indoor Daten für die Verwendung in Navigationsanwendungen gewährleistet ist, müssen die Erfassungsstandards gut definiert werden und die Gebäude möglichst genau und vollständig erfasst werden.

**Indoor Erfassung in OSM**

Bevor man mit der Indoor Erfassung eines Gebäudes beginnen kann, ist es besonders bei großen Gebäuden hilfreich, sich vorab ein Konzept für die Erfassung zu überlegen. Zunächst sollten alle vorhandenen Quellen gesichtet werden, zum Beispiel Bahnhofspläne. Oft ist auch eine Ortsbegehung sinnvoll, um sich ein Bild von der Lage der Gebäudeelemente machen zu können.

Anschließend lädt man sich den in OSM vorhandenen Datenstand herunter und analysiert die Daten in Bezug auf Geometrie und Attribute sowie fehlende Elemente. Es ist wichtig, sich einen Überblick zu verschaffen, wie viele Ebenen vorhanden sind, ob etwas Neues hinzugefügt werden muss oder Änderungen gemacht werden müssen. Man sollte, bevor man mit dem Erfassen beginnt, eine gedankliche Verteilung von Level und Layer vornehmen.

Mögliche Reihenfolge bei der Bahnhofsmodellierung:

- Schienen
- Haltestellenpunkte: Stop-Points
- Plattform (inkl. Überdachung)
- Unter- und Überführungen mit Fußwegen
- Zwischengeschosse
- Zugänge zur Plattform (Aufzug, Rolltreppe, Treppe…)
- Zugang zum Bahnhof (Aufzug, Rolltreppe, Treppe, Eingang…)
- Bahnhofsausstattung (Fahrkartenautomat, Warteraum, Fahrradstellplatz…)
- Bahnhofsumfeld

Für die genaue Modellierung der einzelnen Objekte sollte man sich unbedingt an die gängigen Modellierungsvorschläge aus dem OSM-Wiki halten. Eine Zusammenfassung der Modellierungsvorschläge zu Objekten des ÖPNV und insbesondere zu Indoor Objekten des ÖPNV kann man auf der OSM-Wiki Seite der Firma Mentz Datenverarbeitung GmbH finden (http://wiki.openstreetmap.org/wiki/%C3%96V_Firma_Mentz_Datenverarbeitung_GmbH/Modellierungsvorschl%C3%A4ge)

Das OSM Datenmodell ist ursprünglich nicht dafür ausgelegt gewesen, Indoor Routing zu ermöglichen zum Beispiel wahren mehrere Ebenen übereinander in einem Gebäude so nicht vorgesehen. Daher entstand die Notwendigkeit, einige Modellierungsschemata neu zu entwickeln. Aufgrund der Aktualität dieses Themas ist die OSM Community demgegenüber aber sehr aufgeschlossen.

Die genaue und vollständige Erfassung der Gebäude ist für den Erfolg von Routing Anwendungen in Gebäuden unumgänglich. Nur mit qualitativ hochwertigen Daten kann ein Anwender zuverlässig durch ein Gebäude navigiert werden.

**Zusammenfassung und Ausblick**

Durch das steigende Angebot an Indoor Karten auf der einen Seite und das daraus bedingte wachsende Interesse der ÖPNV Betreiber auf der anderen Seite lässt sich eine stetige Entwicklung in diesem Bereich beobachten. Mittlerweile gibt es einige ÖPNV Betreiber, die als GIS Datenbasis für ihre Auskunftssysteme OSM einsetzen. Durch dieses Interesse an OSM ist eine Verbesserung der Datenqualität und Dichte von ÖPNV Daten in OSM insbesondere auch bei den Gebäuden des ÖPNV zu erwarten.

So ist es heute schon möglich, eine Route innerhalb eines Gebäudes vorauszuberechnen und dabei auf verschiedenste Parameter, wie Rollstuhlgerecht, kürzester Weg, etc. Rücksicht zu nehmen.

**Indoor Routing in Gebäuden des öffentlichen Verkehrs auf Basis von Openstreetmap Daten**

Die zukünftige Herausforderung in Bezug auf Indoor Routing und Navigation besteht in der Ortung des Benutzers einer mobilen Anwendung innerhalb eines Gebäudes und eines echten live Routings mit Fahrplanauskunft. Solche Anwendungen funktionieren unter freiem Himmel schon sehr gut. Bis jetzt hat sich noch keine zuverlässige Technologie zur genauen Ortung von mobilen Geräten in Gebäuden durchgesetzt. Die Forschung an diesem Thema ist zur Zeit hochaktuell und es ist zu erwarten, dass es hier bald einen Durchbruch geben wird, mit dem dann auch live Indoor Routing und Navigation möglich sein wird.

Kontakt zum Autor:

Thomas Jakubicka
Mentz Datenverarbeitung GmbH
Grillparzerstr. 18, 81675 München
+49 (0)89 418 68-0
jakubicka@mentzdv.de

# Das audiovisuelle Erbe der OSGeo-Projekte: Referenzfall GRASS GIS - und Star Trek

Peter Löwe, Margret Plank, Frauke Ziedorn

Seit ihrer Gründung im Jahr 2006 hat sich Open Source Geospatial Foundation (OSGeo) als Referenzierungsstelle für die Qualität von Free and Open Source Software (FOSS) im Geoinformatikbereich etabliert. Eine ständig wachsende Anzahl von Softwareprojekten ist nach erfolgreichem Abschluss des OSGeo-Inkubationsprozeßes als zertifizierte Open Source Projekte akkreditiert. Ihre Web-Portale sind Teile der übergreifenden OSGeo-Webpräsenz. Jedes dieser OSGeo-Softwareprojekte wird von einer Gemeinschaft von Freiwilligen weiterentwickelt. Dafür werden gleichzeitig eine Reihe verschiedener Tätigkeiten geleistet. Hierzu zählt sowohl das Entwickeln und Testen der Software, wie auch das Erstellen von Dokumentationen und Anleitungsmaterialien. Für letzteres werden immer stärker audiovisuelle Videofilmsequenzen eingesetzt, wie etwa kommentierte Screencasts oder Animationsfilme welche die Arbeiten an einer Codebasis in einem Software-Repository visualisieren. Dieses Informationsmaterial wird aktuell zum größten Teil auf Web 2.0 Portalen wie Youtube und Vimeo abgelegt. Damit liegen diese Inhalte außerhalb des Zuständigkeitsbereichs der OSGeo-Gemeinschaft. Die Menge solcher audiovisuellen Inhalte hat in den letzten Jahren stark zugenommen und wächst weiterhin stark.

Die Verbreitung dieser Fachinformationen über Web2.0 Portale ist sowohl aus bibliothekarischer wie Nutzer-Perspektive suboptimal. Es bestehen deutliche Defizite bei der langfristigen Verfügbarkeit der Informationen, ihrer Auffindbarkeit und Zitierfähigkeit. Aktuell ist die Suche über OSGeo-relevante Inhalte nur über vom Verfasser angegebene Metainformationen wie Titel und textuelle Schlagworte möglich. Die Vorhaltung der Inhalte wird durch die Lebensdauer des genutzten Web-Portals bzw. der Registrierung des Verfassers begrenzt. Eine Langzeitarchivierung ist nicht gesichert. Weiterhin kann ein Verweis auf derartige Inhalte lediglich über kurzlebige URL-Referenzen erfolgen.

Zusätzlich zu diesen alle OSGeo-Projekte betreffenden Problemfeldern besteht für das GRASS GIS Projekt zusätzlich die Notwendigkeit der Archivierung der audiovisuellen Projektmaterialien aus der Zeit vor Gründung von OSGeo. GRASS GIS ist das älteste aktive FOSS GIS Projekt, dessen Codebasis bereits 1982 entstand und kontinuierlich weiterentwickelt wird. Historische Materialien zu GRASS GIS aus den 1980er und 1990er Jahren sind bisher nicht vollständig erschloßen. Die fortschreitenden Alterungsprozeße der Speichermedien gefährden ihre weitergehende Verfügbarkeit und erfordern zeitnahe Konservierungsmaßnahmen. Somit bestehen aus der Perspektive der OSGeo-Community Bedarfe für die Erschließung aktueller Inhalte, der Etablierung von Best Practices für weiter zunehmende Informationsmengen, sowie den Erhalt historischer technischer Informationen. Eine zukunftssichere Alternative bietet die Nutzung von innovativen Bibliotheksdiensten, die aktuell von Forschungsbibliotheken weltweit entwickelt werden. Die Technische Informationsbibliothek (TIB) ist die Deutsche Zentrale Fachbibliothek für Technik sowie Architektur, Chemie, Informatik, Mathematik und Physik in Hannover, sowie die weltweit größte Spezialbibliothek für Technik und Naturwissenschaften. Die TIB ist Teil der nationalen Forschungsinfrastruktur und zugleich die größte Fachbibliothek in ihren Bereichen. Ihr Auftrag ist die Versorgung der nationalen und internationalen Forschung und Industrie mit Literatur und wissenschaftlich-technischer Information. Der Bereich Geoinformatik, und damit das Themenfeld FOSS GIS wird dabei als Teilmenge der Angewandten Informatik betrachtet.

Das seit 2014 verfügbare AV|Portal (http://av.getinfo.de) der TIB bietet langfristige Vorhaltung der audiovisuellen Inhalte, deren Zitierbarkeit mittels persistenter Identifikatoren und umfangreiche Suchfunktionen: Hierfür werden die durchsuchbaren Metadaten der Videoinhalte durch Analysen von gesprochener Sprache, Texteinblendungen und Bildinformationen angereichert. Dies führt zu einer erheblich verbesserten Suche nach und in audiovisuellen Ressourcen. Durch die Verbindung eines Digital Object Identifiers (DOI) mit einem Media Fragment Identifier wird die sekundengenaue Zitierfähigkeit der Materialien gewährleistet.

**Das audiovisuelle Erbe der OSGeo-Projekte: Referenzfall GRASS GIS - und Star Trek**

Dieser Beitrag stellt die Erschließung audiovisueller Inhalte aus dem OSGeo-Umfeld durch im AV|Portal exemplarisch anhand von Fallbeispielen vor: Es wird die erfolgreiche Erschließung des GRASS GIS Videos des U.S. Army Corps of Engineers Research Laboratory (CERL) aus dem Jahr 1987 vorgestellt. Der Inhalt dieses historischen Videos bietet einen Einblick in die Frühphase der GIS Entwicklung. Als Beispiele für das Nutzungspotential aktueller Inhalte werden die im AV|Portal verfügbaren OSGeo-Konferenzbeiträge des letzten Jahres präsentiert.
Kontakt zum Autor:

Dr. Peter Löwe
TIB Hannover
Welfengarten 1B
0511-762-3428
Peter.loewe@tib.uni-hannover.de

# Verkehrsmittelbezogene Erreichbarkeitsvisualisierung

Marc Beiling

Initiiert durch eine Idee innerhalb der Mobilitäts- und Verkehrsstrategie MOVE 2013 an der Ruhr-Universität Bochum wurde im Rahmen einer Abschlussarbeit eine Webmapping-Anwendung entwickelt, die Informationen rund um die Mobilität für (insbesondere neue) Studierende darstellt, die bisher noch nicht im Web der Universität zu finden sind. Der Fokus lag auf einer vom Verkehrsmittel (Auto, ÖPNV, Rad, Zu Fuß) und der Zeit abhängigen flächenhaften Darstellung der Erreichbarkeit rund um einen möglichen Wohnort der Studierenden. Zusätzlich integriert wurden POI, die entsprechend der Daseinsgrundfunktionen im Kontext von Alltagsmobilität eine wichtige Rolle spielen. Neben ihrer Einbindung als Layer wird im Anschluß an die Flächenberechnung die Anzahl der POI im Erreichbarkeitspolygon in einem Diagramm wiedergegeben. Der Studierende soll so nach individuellen Prioritäten bezüglich des zeitlichen Aufwands seiner Mobilität und der „POI-Infrastruktur“ einen für ihn geeigneten Wohnort suchen können.

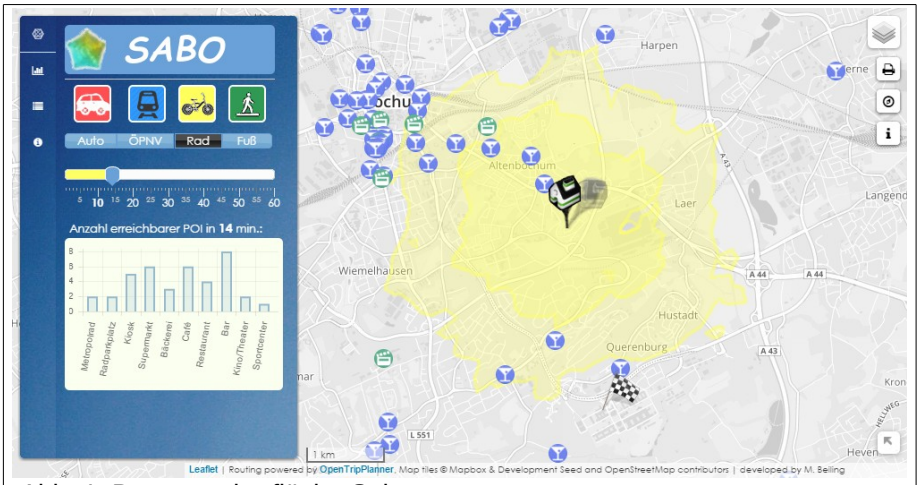


Abb. 1: Benutzeroberfläche Sabox

Verwendet wurden clientseitig die Javascript-Bibliotheken jQuery, Leaflet.js, Chart.js. Für die serverseitige Berechnung der Erreichbarkeit und Weitergabe des Ergebnisses im GeoJSON-Format an den Client wurden zwei unterschiedliche Varianten getestet. Erstens die Umsetzung mit PHP und Postgresql/PostGIS und pgrouting. Zweitens der Betrieb des OpenTripPlanner (OTP) als Standalone-Server und die Abfrage über seine API.

Nicht nur, aber gerade wegen des Routings im öffentlichen Nahverkehr und der Einbindung dafür notwendiger Daten im vielversprechenden GTFS-Format, fiel letztlich die Wahl auf die OpenTripPlanner-Variante.

Die Anwendung ist als beispielhafte Umsetzung zu verstehen, mit der Herausforderungen beim Routing mit OSM-Daten aufgezeigt werden können und der Neulingen im OSM/FOSS-Bereich einen Einblick in netzwerkbasierte Berechnungen/Visualiserungen und dafür zu verwendende Komponenten ermöglichen soll.

Kontakt zum Autor:

Marc Beiling

marcbeiling@gmx.de

# Neues zu BRouter

Dr. Arndt Brenschede

Das Open-Source Projekt BRouter [1] hat fast 3 Jahre Entwicklung hinter sich und sich im Bereich der Outdoor-Navigation auf Android Smartphones einen festen Platz erobert. Aber die Entwicklung geht weiter, und so will dieser Vortrag denn auch nur eine ganz kFleurze Einführung in das Projekt geben um dann einige der Neuerungen des letzten Jahres vorzustellen, die selbst die regelmässigen Nutzer noch kaum kennen. Beispiele sind steigungsabhängige Weg-Präferenzen, schnellere Neuberechnungen und die Integration der 30m-SRTM Höhendaten. Die wichtigste Neuerung aber ist ein neues, flexibleres internes Datenmodell, mit dem ein viel grösserer Teil der Primär-Information aus der OSM Datenbank in den vorverarbeiteten Datenfiles dargestellt werden kann. Dadurch werden sowohl Spezialanwendungen im Wegenetz möglich, etwa im Hinblick auf Barrierefreiheit, Einsatzfahrzeuge oder Schwerlastverkehr, aber auch Anwendungen in ganz anderen Netzen wie dem Schienen, Fluss- oder Stromnetz.

Kontakt zum Autor:

Dr. Arndt Brenschede
Arndt.Brenschede@web.de

Literatur

[1] *www.brouter.de/brouter*

# Der schwere Werdegang zu einem FOSSGIS-Open Source Projekt

Till Adams, Doninik Helle

OpenSource machen ist einfach. Ein bisschen Code geschrieben, einen schicken Lizenz-Header oben drüber gepastet und ab damit auf Git oder eine andere hippe Plattform. Aber damit ist es dann meistens doch nicht getan. Der Vortrag beschreibt warum.

OpenSource-Projekte, natürlich nicht nur im FOSSGIS Umfeld, leben von Ihrer aktiven Community, sie leben davon, dass sie möglichst wenig „Stallgeruch“ einer Firma haben, dass sie an vielen Stellen zum Einsatz kommen und in aller Munde sind. Dies alles sind Schritte, die ein Projekt durchlaufen muss. Dazu kommt die Co-Existenz und eine Art von positivem Wettstreit mit anderen Projekten, die vielleicht sehr ähnliches tun. Weiterhin zu nennen wären auch unbezahlte Aufwände für Homepage, Dokumentation, Übersetzungen, PSC und und und…

Der Vortrag widmet sich dem Prozess hin zu einem „echten“ OpenSource-Projekt. Aufgezeigt wird dies an verschiedenen Beispielen. Oft fangen Open Source Projekte auf Basis von mehreren Entwicklungen an und werden langsam zu einem OpenSource-Projekt entwickelt. Warum das schwierig ist, wird im Vortrag erläutert. Dazu wird das Dilemma dargestellt, in das eine Firma zwangsläufig hineinläuft, nämlich die Balance zwischen Projektarbeit und OpenSource-Projekt zu finden. In diesem Zusammenhang werden auch strategische Entscheidungen, die in den vorgestellten Beispiel-Projekten gelaufen sind, vorgestellt und Ihre Wirkung auf das OpenSource-Projekt projiziert.

# Linked Data basierter Explorer für krebsrelevante Ursache-Wirkungs-Beziehungen im raum-zeitlichen Kontext

Friedrich Müller (MSc Student, Institut für Geoinformatik, Münster), Dorothea Lemke (Institut für Epidemiologie und Sozialmedizin, Münster)

## 1. Einführung

Krebs-Cluster sind ein wichtiges und sehr kontrovers diskutiertes Thema, nicht nur in Deutschland [3]. Ein Cluster wird üblicherweise als räumlich begrenztes, überzufälliges Auftreten von Krebsfällen über den Erwartungswert hinaus definiert [5]. Dabei werden häufig Anfragen aus der Bevölkerung oder von Gesundheitseinrichtungen an die epidemiologischen Krebsregister [2] gerichtet, ob es sich bei dem vermuteten Cluster um eine wirkliche räumliche und zeitliche Erhöhung der Krebsfälle handelt. Das Vorgehen sieht derzeitig so aus, das mit Hilfe von demographisch-epidemiologischen Kennzahlen ermittelt wird, ob das Risiko für diese Krebserkrankung in der Region tatsächlich statistisch-signifikant, d.h. nicht zufällig, erhöht ist. Wenn dieser Zusammenhang positiv ist, wird untersucht, ob dieses tatsächliche Cluster räumlich und zeitlich mit externen Karzinogenen (z. B. Kohlenstoffmonoxid) assoziiert werden kann. Der erste Teil der Cluster-Untersuchung wird standardisiert mit Hilfe der Daten der epidemiologischen Krebsregister durchgeführt und beantwortet. Der zweite Teil der Untersuchung, die Suche nach potentiellen Expositionsquellen, wird häufig unstrukturiert durch Zusammentragen von Informationen aus dem Internet gestaltet, was zusätzlich sehr kosten- und zeitintensiv ist. Dabei basiert die Suche nach potentiellen Karzinogenen auf den Arbeiten/Monographien der IARC (*International Agency for Research on Cancer)* [16], die aber keine direkte räumliche Verknüpfung besitzen. Es handelt sich hierbei um eine Liste unterschiedlicher chemischer Stoffe, die Krebs verursachen oder im Verdacht davon stehen. Ziel dieser vorliegenden Anwendung ist dieses Experten-Wissen der IARC mit räumlichen Ausprägungen dieser Karzinogene (z.B. CO und deren räumliche Emittenten) zu verknüpfen um eine strukturierte und fundierte Evaluation dieser Cluster-Anfragen zu ermöglichen.

Aktuell sind offene krebsrelevante, umweltbezogene Daten von verschiedenen Services verfügbar z. B. Emissionsdaten [15]. Neben der genannten Verteilung der Datenquellen sind unterschiedliche Datenformate und die Heterogenität der Daten eine Herausforderung hinsichtlich der Integration der Daten und deren Visualisierung innerhalb der Anwendung. In diesem Zusammenhang können semantische Technologien hilfreich sein um die verteilten Daten auf eine Weise zu verknüpfen, die die einzelne Bedeutung der Ressourcen explizit darstellt und automatische Assoziationen ermöglicht [4]. Des weiteren zeigen vorangegangene Projekte aus dem Gesundheitsinformationsbereich, dass eine verbesserte und flexiblere Darstellung von Gesundheitsdaten, Analyse- und Abfragemöglichkeiten und Visualisierungen benötigt werden [6, 7].

Die hiermit präsentierte Webanwendung soll exemplarisch aufzeigen wie *Linked Data* und weitere semantischen Technologien dazu geeignet sind die Erreichbarkeit krebsrelevanter Informationen zu erhöhen. Neben der Möglichkeit sich über die krebsbezogenen Ursache-Wirkungs-Beziehungen (aus den Monographien der IARC) zu informieren, ist der Hauptnutzen der Applikation die ermöglichte Erforschung von Umweltdaten (z. B. Luftqualität, Altlasten) im Zusammenhang mit epidemiologischen Datensätzen (z. B. statistische Vergleichswerte von Krebsinzidenzen) u. a. per Geovisualisierungen für die Beispielregion Westfalen-Lippe. Der Fokus liegt hierbei auf der Verkettung: Karzinogen (z. B. $CO_2$) - Emissionsprozess

(z. B. Verkehr) - Emissionsquelle (z. B. Auto) - Transportwege (z. B. Luft) - Exponent (z. B. Männlich/Weiblich) - Krebstyp (z. B. Lungenkrebs).

## 2. Realisierung

Der Workflow (vgl. Abb. 1), beginnend mit den Rohdaten über die semantische Modellierung bis hin zur Webanwendung, ist komplett auf *Open Source* Software (z. B. *Protegé* [12]*, Apache Jena* [8]*, OSM* [11] *& Leaflet* [10]) basierend, und ist in folgenden Arbeitsschritte untergliedert:

- **Definition des Anwendungsbereichs**

  Als Ausgangspunkt der Datenmodellierung wurden Elemente und Beziehungen der Ursache-Wirkungs-Kette vom Karzinogen bis hin zur Krebsinzidenz wie auch die zu betrachtenden Krebsarten und die Referenzregion für die Beispielanwendung festgelegt. Die Festlegung einer Referenzregion begrenzt den Datenfokus auf ein zu realisierendes Level. Bezüglich der Krebstypen wurde sich vorwiegend auf Typen beschränkt, die potentiell von äußeren Faktoren abhängig sind wie z. B. Lungenkrebs.

- **Aggregation der Daten**

  Der nächste Schritt ist das Auffinden, Sammeln und Produzieren von Datensätzen (z. B. über Vorkommen von Karzinogenen in Luft oder Boden, Emissionsquellen wie Industrieanlagen), die das Informationsfundament für die Webapplikation bilden und eine Erforschung der krebsrelevanten Daten ermöglichen. Die Rohdaten kommen von unterschiedlichen öffentlichen Services z. B. vom LANUV NRW (*Landesamt für Natur, Umwelt und Verbraucherschutz Nordrhein-Westfalen*) [17]. Als epidemiologisches Risikomaß wird mit Hilfe der Daten aus dem epidemiologischen Krebsregister und mit entsprechenden demographischen Daten das „standardisierte Inzidenzverhältnis“ (SIR) und deren 95% Konfidenzintervalle für jede Gemeinde berechnet, welches das Verhältnis zwischen beobachteten und erwarteten Krebsfällen in der Referenzregion Westfalen-Lippe wiedergibt.

- **Konvertierung der Datensätze ins RDF (*Resource Description Framework*) Format**

  Die bereinigten Rohdatensätze wurden per Skripte z. B. in Verbindung mit der *Jena Library* [8] oder per *Open Refine* mit RDF Extension [14] in RDF umgewandelt. Die RDF Strukturierung wurde anhand geeigneter Vokabulare z. B. mit der *Datacube Vocabulary* [18] erarbeitet. Die *Datacube Vocabulary,* die die Einteilung von multidimensionalen Datensätzen u. a. in Attributen, Dimensionen und Messungen erlaubt, eignet sich somit für Umweltdatensätze.

- **Encodierung der Datensätze als Linked Data**

  Der Linked Data Aspekt, die Informationen aus den erstellten Datensätzen mit anderen Ressourcen von externen RDF Sets oder Endpoints zu verlinken, ist durch die Verwendung von RDF Properties wie *rdfs:seeAlso, owl:equivalentClass* oder *owl:sameAs* bewerkstelligt. Zusätzliche Informationen, die nicht per URL verlinkt werden konnten wurden textlich per *rdfs:comment* hinzugefügt.

- **Modellierung der Domain Ontology**

Parallel zur Erstellung der RDF Datensätze wurde die Domain Ontology, die die krebsrelevanten Ursache-Wirkungs-Beziehungen darstellt, entwickelt. Die Ontology strukturiert weitere Informationen der Ursache-Wirkungs-Kette z. B. Krebstyp, Karzinogene, Emissionsquellen, Emissionsprozesse, Transportwege, exponierte Orte, exponierte Gruppen und Geschlecht. Die Hintergrundinformationen über diese Beziehungen wurden von den Monographien der IARC herausgearbeitet. Die Erstellung der Ontology wurde mit dem Open Source Ontology Editor *Protégé* [12] durchgeführt.

- **Implementierung der Webapplikation**

Nach der Generierung der RDF Datensätze und der Domain Ontology müssen diese zur Abfrage verfügbar gemacht werden. Dies ist zum jetzigen Zeitpunkt des Projekts unter Verwendung des Sesame RDF Frameworks [13] realisiert. Um die Anforderungen der Applikation wie Visualisierungen per Karte, Grafik oder Text webbasiert umzusetzen wurde die Anwendung u. a. per HTML, CSS, Javascript und SPARQL Technologien in Kombination mit diversen Bibliotheken wie z. B. Leaflet [10] oder jQuery [9] implementiert. Derzeit können Informationen direkt über selbstdefinierte SPARQL- Abfragen und indirekt über SPARQL eingebettete Funktionen erhalten werden.

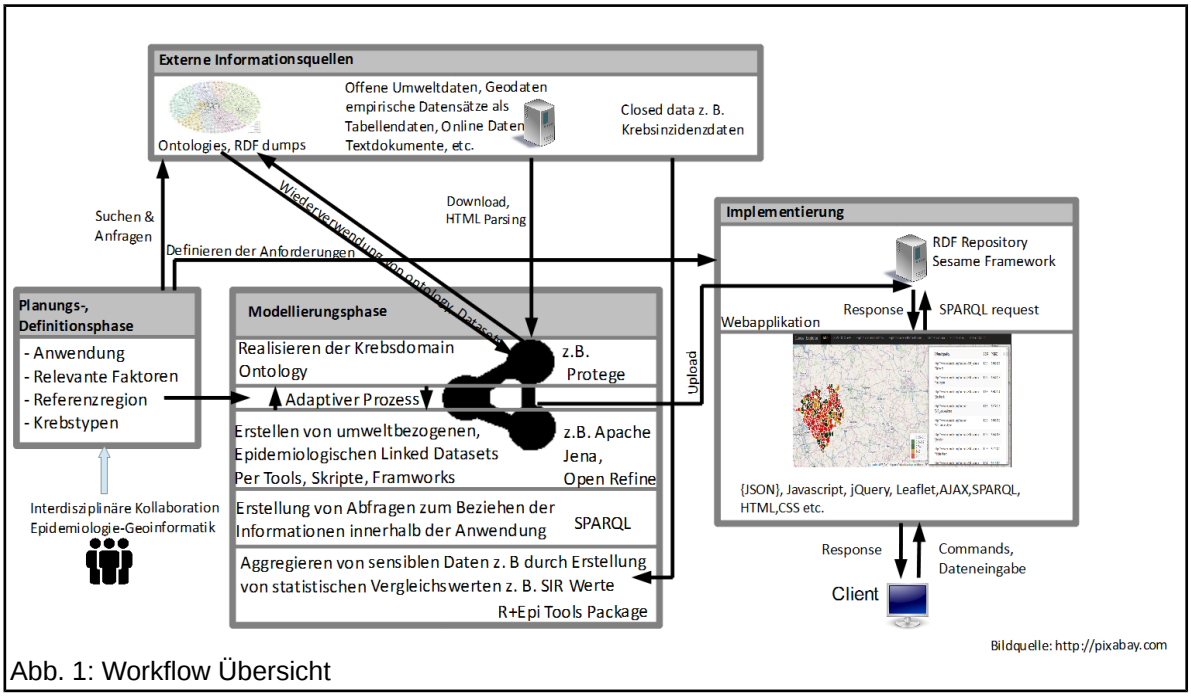



Abb. 1: Workflow Übersicht

## 3. Zusammenfassung

Ein erwartetes Ergebnis ist Linked Open Data, das von ursprünglich verteilten, isolierten umweltbezogenen und epidemiologischen Daten modelliert wurde. Ein weiteres Ergebnis ist die Domain Ontology, die die Ursache-Wirkungs-Kette vom Karzinogen bis hin zur Krebsinzidenz aufzeigt. Insgesamt beschäftigt sich das Projekt mit der Frage wie Linked Data dazu beitragen kann krebsrelevante Informationen, in Zusammenhang mit Krebs-Cluster Anfragen in Raum, Zeit und Semantik in geeigneter Form innerhalb einer epidemiologischen Anwendung, bereitzustellen. Die Anwendung dient als Erforschungs- und Informations-Werkzeug für krebsbezogene Ursache-Wirkungs-Beziehungen. Im größeren Rahmen trägt die Arbeit zu aktuellen Forschungsbereichen wie Geovisualisierungen im Gesundheitsbereich [1] und semantischen Technologien in Informationssystemen bei. Produkte des Projektes (z.B. Domain Ontologie) sind neben dem Code der Applikation zur Wiederverwendung auf *Github* (https://github.com/lodum/CancerExplorer) zugänglich.

Kontakt zu den Autoren:

Friedrich Müller
MSc Student, Institut für Geoinformatik, Münster
f_muel25@uni-muenster.de

Dorothea Lemke
Institut für Epidemiologie und Sozialmedizin, Münster
dorothea.lemke@uni-muenster.de

Literatur

[1] Diez, E., McIntosh B.S. 2009. A review of the factors which influence the use and usefulness of information systems. *Environmental Modelling and Software,* 24.

[2] GEKID: Bevölkerungsbezogene Krebsregister in Deutschland [http://www.gekid.de/registries.html]

[3] Kieschke J. 2010. Auswertung des EKN zur Krebshäufigkeit in der Samtgemeinde Asse, In *Book Auswertung des EKN zur Krebshäufigkeit in der Samtgemeinde Asse.*

[4] Lee T. B., Hendler J., Lassila O. 2001. The semantic web. *Scientific American* 284, 28–37.

[5] Olsen S.F., Martuzzi M., Elliott P. 1996. Cluster analysis and disease mapping - Why, when, and how? A step by step guide. *Brit Med J*, 313:863-866.

[6] Tilahun B., Kauppinen T., Keßler C., Fritz F. 2014. Design and Development of a Linked Open Data-Based Health Information Representation and Visualization System: Potentials and Preliminary Evaluation, *JMIR Med Inform,* 2(2):e31.

[7] Zhou Q., Wang C., Xiong M., Wang H., Yu Y. 2007. SPARK: adapting keyword query to semantic search, in: *ISWC*.

Internet:

[8] https://jena.apache.org/

[9] http://jquery.com/

[10] http://leafletjs.com/

[11] http://www.openstreetmap.de/

[12] http://protege.stanford.edu/

[13] http://rdf4j.org/

[14] http://openrefine.org/ + http://refine.deri.ie/

[15] http://www.gis.nrw.de/ims/ekatsmall2008/small/info.htm

[16] http://www.iarc.fr/

[17] http://www.lanuv.nrw.de/

*[18] http://www.w3.org/TR/vocab-data-cube/*

# 3D GIS Stack aus OpenSource Komponenten

Felix Kunde

In den letzten Jahren hat die dritte Dimension auch Einzug in den gängigen FOSSGIS Lösungen (PostGIS, QGIS, OpenLayers etc.) gehalten, so dass mittlerweile ein kompletter 3D-GIS-Stack aus OpenSource Lösungen realisiert werden kann. Das wichtigste Ziel der hier vorgestellten Projekte ist die Interaktion mit 3D-Webkarten. Der Anwender soll in der Lage sein, mit den 3D-Modellen über das Web arbeiten zu können und sie nicht nur zu betrachten.

Derzeit gibt es einen kleinen Hype um 3D Web-Viewer mit WebGL-Unterstützung. Immer mehr Softwarefimen bieten eigene Lösungen an, die häufig auf offen verfügbaren Engines wie etwa Cesium [1] oder OpenWebGlobe [2] basieren. WebGL-Viewer benötigen keine clientseitigen Plugins und funktionieren auf vielen Endgeräten. Sie laufen auch ohne High-End-Ausstattung sehr schnell und erlauben ein flüssiges interaktives Bewegen durch eine 3D Szene.

Plattformen mit vielen detaillierten volltexturierten Objekten, wie etwa 3D-Stadtmodelle (damit sind keine texturierten Oberflächenmodelle von Städten gemeint), müssen zwar derzeit noch mit stärkeren Performanceeinschränkungen leben, durch die rasant steigende Anzahl an Entwicklern und Anwendern von WebGL sollte dieser Engpass aber bald der Vergangenheit angehören. Aus demselben Grund dürften zunehmend auch freie Viewer in den derzeit von prorietären Angeboten dominierten Markt drängen.

Einen Anfang macht Cuardo [3] – eine OpenSource JavaScript-Bibliothek basierend auf THREE.js und WebGL. Cuardo wird von der französischen Firma Oslandia entwickelt, die sich auch verantwortlich zeigt für den 3D-Support in PostGIS und ein 3D-Plugin für QGIS namens Horao [4]. Oslandia hat das Ziel einen kompletten 3D-GIS-Stack von der Datenbank bis zur Webvisualisierung zu realisieren. Datenbank-seitig wird u.a. die 3D City Database [5] eingesetzt, eine OpenSource-Lösung zum Speichern von CityGML-basierten 3D Stadt- und Landschaftsmodellen in PostGIS.

Seit der neusten Version der 3D City Database steht eine WFS-Schnittstelle (Simple) zur Verfügung, die Viewern wie Cuardo die Möglichkeit bietet, Inhalte der Datenbank abzurufen, ohne das spezifische Abfragen gegen das zugrunde liegende Datenbankschema notwendig sind. Der Client muss nur den WFS 2.0-Standard des OGC unterstützen. Für die Zukunft ist geplant, das neben einer vollen Unterstützung des OGC Filter Encoding auch Transaktionen mit dem WFS möglich sein werden.

Das wichtigste Ziel der hier vorgestellten Projekte ist die Interaktion mit 3D-Webkarten. Der Anwender soll in der Lage sein, mit den 3D-Modellen über das Web arbeiten zu können und sie nicht nur zu betrachten. Der momentane Wirbel um WebGL erhöht nicht nur die Aufmerksamkeit für 3D-Webmapping, er schürt auch Erwartungen an ein 3D-WebGIS, das seinem 2D-Pendant in nichts nach steht.

# Location-based Task Management

Daniel Kastl

FOSS4G Software kann bei vielerlei Aufgaben mit Raumbezug helfen, Arbeitsabläufe zu verbessern und zu optimieren, und die Mitarbeiter bei Ihrer täglichen Arbeit zu entlasten. Dieser Vortrag stellt ein Konzept vor für "Location-based Task Management".

Geschäftsleute müssen ihre Kunden besuchen, ambulantes Pflegepersonal Ihre Patienten. Städtische Mitarbeiter müssen in regelmäßigen Intervallen die öffentlichen Vermögenswerte und Infrastruktur überprüfen, usw.. Es gibt viele Beispiele aus dem Arbeitsalltag, bei denen Aufgaben einen Raumbezug haben.

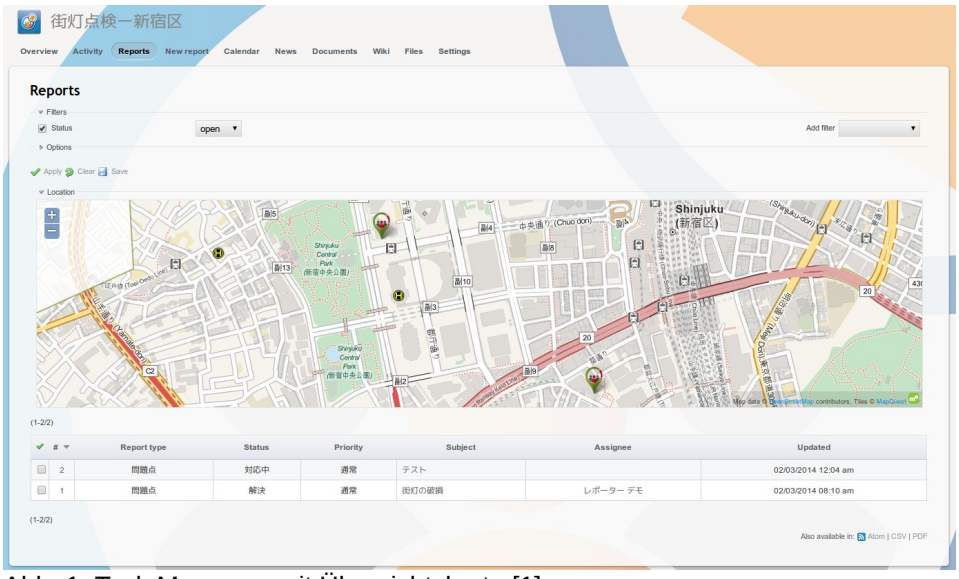


Abb. 1: Task Manager  mit Übersichtskarte [1]

FOSS4G Software kann dabei helfen, diese Aufgaben zu organisieren und zu optimieren. Denn GIS Technologie bedeutet weit mehr, als ein paar farbige Marker auf einer Online-Karte darzustellen. Ortsbasierte Aufgaben durchlaufen in der Regel verschiedene Zustände, haben unterschiedlich hohe Prioritäten, können während der Bearbeitung an unterschiedliche Personen übertragen werden und folgen fast immer einem definierten Ablauf, zum Beispiel:

1. Neue Aufgabe erstellen
2. Aufgabe überprüfen und bewerten
3. Aufgabe durchführen und dokumentieren
4. Aufgabe abschließen

Mit Blick auf Optimierung von Arbeitsabläufen und Tourenplanung und basierend auf verfügbarer Open-Source-Software und offenen Standards, haben wir eine Task-Management-Software entwickelt, die im sich momentan in der Testphase befindet. Die dazu verwendete Open Source Software besteht aus folgenden Komponenten:

- Redmine Issue Tracker [2]
- PostgreSQL/PostGIS als Datenbank

- OpenLayers zur Kartendarstellung

Redmine ist ein in der Softwareentwicklung erfolgreich eingesetztes Tool zum Projektmanagement. Angepasst an die Anforderungen von ortsbezogenen Dienstleistungen und Aufgaben, kann es auch für Nicht-Softwareentwickler verwendet werden und dazu beitragen, Dienstleistungen vor Ort zu erleichtern, die Servicequalität zu verbessern und die Effizienz zu steigern. Redmine kann dabei an nahezu beliebige Anforderungen von Arbeitsabläufen angepasst werden, und kann bei Bedarf mittels Plugins erweitert werden. Eine ReST-API erlaubt außerdem die Integration mit vorhandener Software oder die Entwicklung speziell angepasster mobiler Clients.

Kontakt zum Autor:

Daniel Kastl
Georepublic (Deutschland)
Salzmannstr. 44, 81739 München
+49 (089) 4161 7698-1
daniel.kastl@georepublic.de

Links:

[1] Demo: http://beta.sugurepo.jp

[2] Redmine: http://www.redmine.org

# Schatzsuche in OpenStreetMap

Roland Olbricht

Mit der Overpass API lassen sich auch ungewöhnliche Daten in OpenStreetMap finden - und bewundern, ihnen nachspüren oder sie korrigieren. Neben einer Präsentation der Ergebnisse des Vergleiches von Bonn und Münster werden dabei auch ausführlich die verwendeten Overpass-API-Abfragen vorgestellt, so dass jeder die Abfragen leicht für seine Stadt wiederholen oder inhaltlich auf seine Bedürfnisse anpassen kann.

Mit der Overpass API lassen sich auch ungewöhnliche Daten in OpenStreetMap finden - und bewundern, ihnen nachspüren oder sie korrigieren.

Wir untersuchen zwei Großstädte, Bonn und Münster, mit verschiedenen Ansätzen auf ungewöhnliche Daten, z.B.: Was bleibt, wenn man alle Objekte weglässt, die sich anhand ihrer Tags einer klaren Kategorie zuordnen lassen? Wo fehlen wahrscheinlich noch Briefkästen? Welche Brücken haben keine Höhenangabe? Wo steht in den Tags eines Elements vermutlich Spam? Welche Restaurants (und sonstige Points-of-Interest) sind verdächtig alt?

Neben einer Präsentation der Ergebnisse für die beiden benannten Städte werden dabei auch ausführlich die verwendeten Overpass-API-Abfragen vorgestellt, so dass jeder die Abfragen leicht für seine Stadt wiederholen oder inhaltlich auf seine Bedürfnisse anpassen kann.

# Cesium - der 3D-Globus im Web

## Open Source Technologie in drei Dimensionen

Elisabeth Leu

Cesium ist ein performantes und interoperables Tool für die Visualisierung von Daten im dreidimensionalen Kontext. Stichworte zum Vortrag: 3D - JavaScript - Open Source - WebGL - Zeitabhängige Darstellung - OGC Standards - Openlayers 3 API - Demos und Beispiele.

Mit der JavaScript Programmbibliothek Cesium kann ein 3D-Globus für das Web erstellt werden, ohne dass für die Visualisierung Plugins gebraucht werden. Cesium benutzt WebGL und unterstützt ausserdem OGC-Standards wie WMS oder WMTS. Dies macht es zu einem performanten und interoperablen Tool für die Visualisierung von Daten im dreidimensionalen Kontext.

Die Präsentation stellt das Cesium.js Projekt vor und möchte folgende Fragen beantworten:

- Ein Opensource 3D Globus - was kann Cesium?
- Performante mehrdimensionale Visualisierung im Web - was steckt dahinter?
- 3D ist überall - wo wird Cesium eingesetzt?

Ausserdem wird die Kombination von Openlayers3 mit Cesium vorgestellt und ein Ausblick über die nächsten Entwicklungen gegeben.

### Links

- cesiumjs.org
- OL3-Cesium Open Source release: http://www.camptocamp.com/de/actualite/ol3-cesium-open-source-release/

# MapFish Print V3: Printing maps like a boss

## The next generation of printed maps

Tobias Sauerwein

Dieser Vortrag stellt die neue Version von MapFish Print vor und spricht Themen an wie die neuen Features und deren Nutzung, die Erstellung von Templates mit dem Report-Designer von JasperReports, Upgrade von der vorherigen Version, Skalierbarkeit und Erweiterung durch eigene Module.

Das Projekt MapFish Print besteht aus einer Bibliothek und einer Web-Anwendung zum Druck von Reports mit Karten, wobei eine Vielfalt von Quellen unterstützt wird, zum Beispiel WMS, WMTS, OpenStreetMap-Kacheln, WFS oder GeoJSON.

Mittlerweile besteht das Projekt seit fast einem Jahrzehnt und wird in einer Vielzahl von Websites erfolgreich eingesetzt. Allerdings ist es etwas in die Jahre gekommen, so dass die ursprüngliche Architektur zum Hindernis wurde. Beispielsweise ist die Formatierung der Reports über eine Konfigurationsdatei nicht sehr flexibel, das Ausgabeformat ist prinzipiell auf PDF beschränkt, es ist nicht möglich mehr als eine Karte auf einer Seite anzuzeigen und andere Limitationen.

MapFish Print V3 ist das Ergebnis eines kompletten Rewrites der Implementierung. Dank der Integration mit der weit verbreiteten Reporting-Engine JasperReports und der neuen, erweiterbaren Architektur ist MapFish Print flexibler, mächtiger und einfacher zu Skalieren als je zuvor.

Diese Präsentation richtet sich an Web-Entwickler und Projektleiter, und spricht folgende Themen an:

- Die neuen Features der neuen Version
- Wie können diese Features genutzt werden?
- Die Verwendung des Report-Designers
- Beispiele für komplexe Reports
- Upgrade von der vorherigen Version
- Die Architektur, wie sie die Skalierbarkeit erleichtert und wie sie es ermöglicht MapFish Print durch eigene Module zu erweitern

# FlatMatch: Online-Wohnungssuche mit OSM-Daten

### Können wir tausende deutsche Vermieter zu OSM-Mitwirkenden machen?

Robert Buchholz

Interaktive 3D-Wohnungsbesichtigung im Browser auf Basis von OSM-Daten ermöglicht ein besseres Bewerten freier Wohnungen und deren Umfeld. Und sie macht Vermieter zu OSM-Stakeholdern. Die Mitarbeit kommerzieller Nutzer an OSM hängt jedoch allein davon ab, ob sie dadurch einen wirtschaftlichen Mehrwert erhalten. Diese Vortrag stellt ein Projekt vor, dass potentiell zehntausende kleine und mittlere Unternehmen in Deutschland diesen Mehrwert bietet.

Kontinuierliche Beiträge zu OSM erfolgen vor allem durch die Community und kommerzielle Nutzer. Letztere leisten durch ihr strukturiertes Vorgehen und umfangreiches Zeitbudget einen wichtigen Beitrag zu Datenqualität, und -umfang von OSM. Die Mitarbeit kommerzieller Nutzer an OSM hängt jedoch allein davon ab, ob sie dadurch einen wirtschaftlichen Mehrwert erhalten. Diese Vortrag stellt ein Projekt vor, dass potentiell zehntausende kleine und mittlere Unternehmen in Deutschland diesen Mehrwert bietet: FlatMatch ist eine Web-Anwendung, um Mietwohnungen direkt im Webbrowser virtuell zu besichtigen und so eine geeignete neue Wohnung zu finden. Hierzu wird nicht nur die Wohnung interaktiv dreidimensional dargestellt. Auch die Aussicht aus der Wohnung wird - auf Basis von OSM-3D-Gebäuden und TileMaps - dargestellt, um einen besseren Einblick in das Wohnumfeld zu geben. Dies erlaubt Mietern, sicherer und auch ohne langwierige Vor-Ort-Besichtigungen eine für sie passende Wohnung zu finden. Und es erlaubt Vermietern, den Personalaufwand für Besichtigungen zu reduzieren, und ihre Wohnungen auch auf Basis deren Umfelds zu vermarkten. Damit haben erstmals zehntausende Vermieter einen Mehrwert davon, OSM-Daten in der Umgebung ihrer Wohnungen mit möglichst großen Detailgrad beizutragen, zu aktualisieren und Fehler zu korrigieren. Dies kann einen wichtigen Beitrag zu OSM leisten, und – gerade bei größeren Vemietern – OSM-Community-Mappern eine Verdienstmöglichkeit für das Mappen dieser Regionen eröffnen.

**Links**

- Demo von FlatMatch: http://rbuch703.de/flat3
- GitHub-Seite für den Quellcode der StreetView-Komponente: http://github.com/rbuch703/osm-streetview
- Demo allein der StreetView-Komponente: http://rbuch703.de/osmsv

# Intermodales ÖPNV/Rad/Fuss Routing

Dr. Arndt Brenschede

Die klassische Fahrplanauskunft erwartet als Eingabedaten eine Start- und eine Zielhaltestelle und damit bereits einen Teil der Lösung. Diesen Widerspruch haben die Anbieter erkannt und bieten zunehmend auch „intermodale“ Auskunftssysteme an, bei der die eigentlichen Start- und Ziel-Positionen angegeben werden – das Auskunftssystem berechnet dann nicht nur die eigentliche Fahrplanverbindung, sondern auch die Fuss- oder Radwege zu und von den Haltestellen. Entsprechende Angebote gibt es von der Bahn, den meisten Verkehrsverbünden, einigen der landesweiten Fahrradroutenplaner und auch, mit für Deutschland stark eingeschränktem Fahrplan, bei Google-Transit.

Bei genauerem Hinsehen arbeiten all diese System jedoch nicht „echt“ intermodal, sondern beschränken sich darauf, nur für den ersten und den letzten Abschnitt einer Reise individuelle Strecken zu berechnen. Wo es notwendig ist, in der Mitte einer Reisekette einen Abschnitt individuell zurückzulegen, etwa um von einem Bahnhof zu einer Strassenbahn zu gelangen, sind solche Verbindungen teilweise in die Fahrplandatenbank eingepflegt, teilweise nicht.

Dieser Kurzvortrag zeigt, welche Folgen diese Einschränkung bereits für die Qualität der klassischen Fahrplanauskunft hat, und dass etwa Fahrradmitnahme sich mit solchen Systemen überhaupt nicht sinnvoll rechnen lässt. Aber auch z.B. „Fitnessrouting“, bei dem eine Präferenz auf einen möglichst grossen individuellen Anteil gelegt wird, bei nicht oder nur gering erhöhter Gesamtreisezeit, ist mit solchen Systemen nicht möglich – ein Anwendungsfall, der für Millionen Büropendler von Interesse ist.

Anhand eines Prototyps, der mit Wegedaten aus Open-Street-Map und begrenzten, manuell eingepflegten Fahrplandaten arbeitet, wird gezeigt, was tatsächlich möglich ist und dass OSM und der ÖPNV eigentlich ein Dreamteam wären.

ÖPNV Daten sind in Deutschland jedoch in maschinen-lesbarer Form nicht zugänglich, was etwa von der Deutschen Bahn mit der Sorge um die Qualität der Fahrplanauskunft begründet wird [1]. Eine kleine Anfrage des Bundestages zu den tatsächlichen Hintergründen dieser Zurückhaltung einerseits und der exklusiven Kooperation mit Google andererseits wurde von der Bundesregierung kürzlich ausweichend beantwortet [2].

Kontakt zum Autor:

Dr. Arndt Brenschede
Arndt.Brenschede@web.de

Literatur

[1] *www.bahn.de/dbbahn/view/service-notizen/2012-september/offene-fahrplandaten-der-deutschen-bahn.shtml*

[2] *Bundestags Drucksache 18/3674 vom 30.12.2014*

# OSGeo-Live – Überblick über das erfolgreiche Open Source Projekt

Astrid Emde


OSGeo-Live ist ein Open Source Projekt, das mehr als 50 Softwareprojekte aus dem Bereich FOSS+GIS bündelt. Hier findet sich eine Sammlung aus den Bereichen Web Mapping Clients und Server, DesktopGIS, Datenbanken, Krisenmanagement, räumliche Tools und Bibiotheken sowie Daten.

Jedes halbe Jahr wird eine neue Version von OSGeo-Live erstellt, die dem Nutzer die verschiedenen Anwendungen, Daten und Informationen als sortierte Sammlung anbietet. Der Einstieg wird leicht gemacht, da die Installation und Konfiguration der Software entfällt. Alles liegt bereits vor und kann direkt getestet werden. Dokumentationen und Quickstart-Dokumente helfen dabei beim Einstieg.

OSGeo-Live ist ein erfolgreiches OpenSource-Projekt. Projektübergreifend wurden viele Freiwillige gefunden, die regelmäßig die Inhalte aktualisieren. So ist ein Produkt entstanden, das als globale Visitenkarte nicht nur der OSGeo-Projekte dient.

OSGeo-Live kann in Workshops und eigenen Veranstaltungen verwendet werden. Die FOSS- und GIS-relevante Software wird mehrsprachig und mit Dokumentation zur Verfügung gestellt. Auch zu den verschiedenen OGC Standards gibt es Informationen. Die Dokumentation liegt mittlerweile in dreizehn Sprachen vor (englisch, deutsch, indonesisch, italienisch, polnisch, griechisch, japanisch, französisch, catalanisch, chinesisch, koreanisch, russisch). Mit jeder neuen Version kommen neue Software und neue Übersetzungen hinzu.

Gerade für FOSS-GIS-Neueinsteiger bietet OSGeo-Live einen guten Überblick. Wenn Ihnen keine Geodaten zum Testen zur Verfügung stehen, können die enthaltenen Beispieldaten von Natural Earth und OpenStreetMap benutzt werden.

Die ehrgeizigen Ziele wie mehrsprachige Dokumentation, Benutzung von Beispieldatensätzen und Support in der Community stellen ganz unterschiedliche Anforderungen dar. Es gilt nicht nur technische Probleme zu lösen. Hinzu kommen terminliche Absprachen, damit zu bestimmten Anlässen wie wichtigen Konferenzen und Workshops aktuelle Versionen bereit stehen.


Aufbau der Softwaresammlung

Durch das OSGeo-Live Projekt werden bootfähige ISO-Images als auch virtuelle Maschinen zur Verfügung gestellt. Das gesamte System basiert auf dem Ubuntu-Derivat Lubuntu. Darin enthalten sind vollständig installierte und konfigurierte Anwendungen aus der gesamten GIS- Welt. Es muss nichts lokal installiert werden, der benutzte Rechner wird in keiner Weise verändert. Die ISO-Images können entweder als bootfähige DVD oder USB-Stick oder als virtuelles virtuelle Maschine genutzt werden.

Zu jedem Projekt gibt es eine Dokumentation. Dazu gehört eine Übersicht, ein Quickstart und ggf. die Originaldokumentation des Projektes. Die Übersicht enthält eine kurze Beschreibung des Projektes, Angaben zu den Features, benutzte Lizenz sowie Bezugsquellen und Links zur Projektseite. Mit Hilfe der Quickstarts werden dem Benutzer die ersten Schritte vereinfacht.

Folgende Komponenten finden sich auf OSGeo-Live 8.5:

Desktop GIS

(Generelle Geodaten Anzeige, Barbeitung und Analyse über den Desktop)

- Quantum GIS
- GRASS GIS
- gvSIG Desktop
- uDig (User-friendly Desktop Internet GIS)
- Kosmo Desktop
- OpenJUMP GIS
- Saga

Browser Clients

(Generelle Geodaten Anzeige, Barbeitung und Analyse über den Browser)

- OpenLayers - Browser GIS Client
- Leaflet - Interaktive Karten für mobile Geräte geeignet
- Cesium - 3D Globen und 2D Karten im Browser
- Geomajas - Browser GIS Client
- Mapbender - Geo-Portal-Lösung
- MapFish - Framework zum Aufbau von Web-Mapping Anwendungen
- GeoMoose - Web GIS Portal
- Cartaro- Geospatial CMS
- GeoNode - Geospatial CMS

Internet Dienste

(Veröffentlichung von Geodaten im Netz)

- GeoServer
- MapServer
- deegree
- ncWMS - Web Map Service
- EOxServer - Web Coverage Service
- GeoNetwork - Metadata Katalog und Katalogdienst für das Web
- pycsw - Metadata Katalog
- MapProxy - Proxy für WMS und Tile-Dienste
- QGIS Server - Web Map Service
- 52°North WPS - Web Processing Service
- 52°North SOS - Sensor Observation Service
- TinyOWS - WFS-T Service
- ZOO-Project - Web Processing Service

Datenbanken

(Speicherung von räumlichen Daten)

- PostGIS - Räumliche Datenbank
- SpatiaLite - Leichtgewichtige Datenbank
- Rasdaman - Multi-Dimensionale Datenbank für Rasterdaten
- pgRouting - Routing für PostGIS

Navigation und Karten

- GpsDrive - GPS Navigation
- GpsPrune - Anzeige, Bearbeitung und Konvertierung von GPS Tracks
- Marble – Virtual Globe
- OpenCPN - Darstellung von Seekarten und GPS
- OpenStreetMap - Open Street Map Werkzeuge
- Viking - GPS Datenanalyse und -anzeige

Spezielle GIS Software

- GeoKettle - ETL Tool (Extrahieren, Transformieren und Laden)
- GMT - Kartographisches Rendering
- Ipython -
- Mapnik - Kartographisches Rendering
- TileMill - Styling and Publishing
- MapTiler - Erzeuge Bildkacheln (Tiles)
- OSSIM Image Processing
- OTB - Bildprozessierung
- R for Spatial Data - Statistische Berechnungen

GIS Werkzeuge

- Sahana Eden - Katastrophenmanagement
- Ushahidi - Kartendarstellung und Zeitachsen für Ereignisse
- osgEarth - 3D Terrain Rendering
- MB-System - Sea Floor Mapping
- zyGrib - Wettervorhersagekarten

Daten

- Natural Earth - Globale Daten
- OSGeo North Carolina, USA Schulungsdatensatz

- OpenStreetMap - Beispiel-Extrakt von OpenStreetMap
- NetCDF Data Set

GIS Bibliotheken

- GDAL/OGR - Geospatial Data Translation Tools
- JTS Topology Suite (JTS) - Java Topology Suite
- GeoTools - Java GIS Toolkit
- GEOS - C/C++ Spatial Library
- MetaCRS - Coordinate Reference System Transformations
- libLAS - LiDAR Data Access

Geschichte von OSGeo-Live

Schon 2007 entstand auf der FOSS4G die Idee, eine DVD mit OSGeo Software zu erstellen. Im Jahr 2009 wurde die Version 2.0 auf der FOSS4G mit 21 Projekten vorgestellt und war ein großer Erfolg. Die DVD wurde von LISAsoft zusammengestellt und unter dem Namen Arramagong veröffentlicht.

2010 folgte dann die Version 3.0 mit 34 Projekten auf der FIG (International Surveyors Conference). Die DVD kam hier bereits in Workshops zum Einsatz.

Im September 2010 kam zur FOSS4G in Barcelona die Version 4.0 heraus. Diese Version trug dabei erstmals den Namen OSGeo-Live. Auch mit dieser Version ist die Anzahl der Projekte gestiegen. Mit 43 Projekten und einer englischen Dokumentation zu den Projekten und zu OGC Standards wurde die DVD auf der Konferenz verteilt und in den Workshops eingesetzt. Einen Monat später wurde die DVD auch auf der INTERGEO in Köln ausgegeben.

Im Jahr 2012 ist der Einsatz von OSGeo-Live beeindruckend. OSGeo-Live wurde weltweit auf mehr als 40 Veranstaltungen vorgestellt oder verwendet.

Auch 2013 und 2014 kam OSGeo-Live auf der FOSSGIS Konferenz (Rapperswil, Berlin) und auf der FOSS4G (Nottingham, Portland) in den Workshops zum Einsatz. Ab 2014 wurde das Erstellen von DVDs eingestellt, da immer mehr Anwender mittlerweile nicht mehr über DVD Laufwerke verfügen. Die Werbung über die Postkarten, soll dazu anregen, sich die neuste Version herunterzuladen und zu testen. Auch diesem Jahr wird OSGeo-Live auf der FOSSGIS 2015 in den Workshops genutzt.

Es gibt ein Kernteam aus 4 Personen, das das Projekt koordiniert. Das Packen der Projekte übernehmen über 80 Beteiligte, ähnlich viele erstellen die Übersetzungen und Tausende schreiben die Software oder erfassen die Daten.

OSGeo-Live macht OSGeo Software und die OSGeo Foundation greifbar. Das Projekt spiegelt die hohe Aktivität und zeigt anschaulich, in welchem rasanten Tempo sich die OpenSource Software und Daten entwickeln.


Herzlichen Dank an alle Beteiligten und viel Spaß mit OSGeo-Live!

Kontakt zur Autorin:

Astrid Emde
WhereGroup GmbH & Co. KG
Eifelstraße 7
53119 Bonn
+49 (0)228 909038-0
astrid.emde@wheregroup.com

Literatur

[1] Webseite http://live.osgeo.org

[2] Download http://live.osgeo.org/en/download.html

[3] Wiki: http://wiki.osgeo.org/wiki/Live_GIS_Disc

[4] IRC:  irc://irc.freenode.net#osgeolive