


Freie Universität Berlin
Fachbereich Geowissenschaften
Institut für Geographische Wissenschaften
Modul 401 : Geoinformatik
Abschlussarbeit
bei Herrn Dr. Hans-Peter Thamm

Gemeinschaftsgärten in Berlin

Implementierung eines WebGIS und räumliche Analyse

von

GODT, MAX	429 48 60	*max.godt@fu-berlin.de*
RICHTER, JON	429 59 51	*jon.richter@fu-berlin.de*
WIESIOLEK, LENNART	429 32 36	*lennart.wiesiolek@fu-berlin.de*

Inhaltsverzeichnis

1 Einleitung 3
1.1 The Commons 5
1.2 Zum Aufbau dieser Arbeit 6
2 Anwendungsszenario : Urbanes Gärtnern in Berlin 6
2.1 Zivilgesellschaftliche Stadtentwicklung 7
2.2 Daten 8
3 Einführung in internetfähige Geographische Informationssysteme 9
3.1 Entwicklung von InternetGIS 10
3.2 Lizenzen 13
3.3 Mashups 15
3.4 Standards und Datenformate 18
3.5 Geodateninfrastrukturen 19
3.6 Geospatial Free and Open Source Software (GFOSS) 20
4 Implementierung des WebGIS 22
4.1 Marktschau 22
4.2 Implementierung 23
4.3 Kartengrundlagen 25
4.4 Gehversuche 26
4.5 UMN Mapserver 27
4.6 Mashup Verwendung 29
4.7 OpenLayers 30
5. Datenaufbereitung und Anwendungsbereiche im Zusammenhang mit Gemeinschaftsgärten in Berlin 32
5.1 Datenaufbereitung 32
5.2 Druckkarten 34
5.3 Forschungsfragen mit R 36
6. Fazit und Ausblick 39

Literaturverzeichnis 42
Abbildungsverzeichnis 45
Codeverzeichnis 46
gartenkarte.r 46
gartenkarte.js 49
tile_merger.py 56
tile_scraper.sh 58

„Geo-Informationen sind allgegenwärtig. Sie ziehen sich quasi wie ein roter Faden quer durch Wirtschaft, Forschung und den Alltag der Menschen. Ein Dienst, der relevante ortsbezogene Informationen sinnvoll und zielführend anbietet, ist heute schlicht und einfach noch nicht verfügbar.“

Kai Krause im Mai 2008 (PROVE 2008)

1 Einleitung

Das interdisziplinäre Fachgebiet der Geoinformatik stellt eine Funktion aus Informatik, Geographischen Informationssystemen sowie Raumwissenschaften dar (DE LANGE 2006:1). Uns als Studenten der Geographischen Wissenschaften stellte sich in diese Arbeit die Frage, wie wir die Kompetenzen dieses Faches nutzen können, um eine von uns gewählte Thematik zu bearbeiten. Als Grundlagen dabei dienten drei Grundpfeiler, welche ein erstes abstraktes Ziel und den Weg zu unserer Forschungsfrage umreißen:

- Als thematisches Feld dient jenes des *Urban Gardening.*
- Alle Arbeiten werden mit Free and Open Source Software (FOSS)-Lösungen erfolgen.
- Mit der Arbeit soll ein konkreter Anwendungsbezug hergestellt werden.

Die Thematik des *Urban Gardening* erlangte besonders im Berlin der letzten Jahren eine immer größere Bedeutung. Zwar gibt es schon seit sehr viel längerer Zeit Projekte zum städtischen Obst- und Gemüseanbau. Ein regelrechter Boom, einhergehend mit der Aufmerksamkeit der Medien, ist allerdings erst seit jüngster Zeit zu beobachten. In Folge dessen ist zu beobachten, dass in Berlin auf dem versiegelten oder aus dem nicht versiegelten Boden immer mehr Gemeinschaftsgärten sprießen. Aufgrund der Dynamik, der Vielfalt und der rasch wachsenden Größe dieses Themenfeldes erachteten wir es als äußerst realistisch hier zahlreiche Ansatzpunkte und sinnvolle Anwendungsfelder finden zu können.

Da die Bewegung sehr auf partizipatorischen Grundlagen fußt, stellte sich uns zunächt die Frage, inwieweit auf digitaler Ebene bereits Plattformen zur Vernetzung, zum Informationsaustausch und zur öffentlichen Repräsentation bereits vorhanden sind. Nach Recherchen in diesem Gebiet stellte sich heraus, dass zwar seit einigen Jahren eine Internetplattform und Projektarchiv namens *Urbanacker*[1] für die Berliner Gemeinschaftsgärten existiert. Dabei fiel uns aber auf es allerdings völlig an einer ästhetisch anspruchsvollen und aussagefähigen Karte fehlte.[2] Die mittlerweile inaktiv gewordene Seite wurde während unserer Arbeit dann vom *Stadtacker*[3] beerbt. Die Redaktion des Portals, das Forschungsprojekt *Innovationsanalyse Urbane Landwirtschaft* (INNSULA) des ZALF (Zentrum für Agrarlandschaftsforschung) zusammen mit dem Allmendekontor / Workstation Ideenwerkstatt, kam im Zuge unserer ursprünglichen Kooperationsanfrage an den Urbanacker auf uns zurück und bat uns, für das neue Portal *Stadtacker* eine neue Karte zu entwerfen.

Mit der günstigen Datengrundlage der INNSULA Forschung ausgestattet, verfestigte sich die Idee ein WebGIS für dieses Themenfeld zu erstellen, als eine brauchbare und präsentable Verortung der Gärten im Internet. Mit dem Anspruch zu einem möglichst großen Teil FOSS Lösungen (nähere Definitionen in Kapitel 3) zu nutzen, bewegen wir uns im Feld der sog. *Neogeographie,* ein Begriff, der etwa 2006 als Reaktion auf traditionelle und proprietäre Ansätze für einen offenen Datenaustausch und die Entwicklung von FOSS entstand (WARREN 2006 : 18).
Zudem passt der FOSS-Gedanke sehr gut zu den oben erwähnten Anliegen und Vorsätzen der meisten Gemeinschaftsgärten, denen die freie Wissensweitergabe ein Hauptanliegen ist.

[1] *http://urbanacker.net* (04.01.2013)
[2] Es gab lediglich ein kleines GoogleMaps Mashup.
[3] *http://stadtacker.net* (04.01.2013)

1.1 The Commons

Die Ideen von kollektiver Intentionalität (vgl. TOLLEFSEN : 2004), kollaborativer Wissenskonstruktion (ZUMBACH, RAPP : 2001) und, vereinfacht subsumiert, der Schwarmintelligenz[4], sickern langsam aus den Pionierkreisen von Wissenschaftlern und Ingenieuren vermehrt in das allgemeine Gedankengut. Wir sprechen zunehmend von den *Commons*. Einer Idee, welche bspw. auch von einigen der Berliner *Urban Gardening* Aktivisten adaptiert und übersetzt als *Allmende* bezeichnet wird. Gemeint ist eine neue Aufmerksamkeit für gemeinschaftlich geteilte Güter, seien es - nur um einige zu nennen - im globalen Maßstab: Luft, Wasser und Boden, im regionalen konkrete Rohstoffvorkommen oder im lokalen die Liegenschaften. Aber auch das Wissen selbst lässt sich mit einem erweiterten, abstrahierten Verständnis des Begriffes erfassen.

Im Sinne der *Commons* kann die Weitergabe einer Softwareanwendung an die AkteurInnen der Gemeinschaftsgärten ohne die inheränten Einschränkungen kommerzieller und proprietärer Lösungen wesentlich barrierefreier erfolgen. Dies war besonders deutlich, als im Laufe der Kooperation mit dem INNSULA Projekt Schwierigkeiten dabei auftraten eine Schnittstelle an das dort verwendete proprietäre *Content Management System* (System zum Austausch von beliebigen Inhalten) *SharePoint* zu bekommen.

Eine ausschließliche FOSS-Architektur würde gleichzeitig die zukünftige Pflege der WebGIS–Infrastruktur und -Daten nach Abschluss des Projektes durch die Nutzer möglich machen. Hinzu kommen Vorteile projektökonomischer Natur: die Implementierung kann größtenteils kostenschonend und ohne Abhängigkeit von externen Dienstleistern erfolgen, bspw. durch Arbeitsleistung ehrenamtlicher Mitarbeiter ersetzt werden. Wir bewegen uns in einem Feld neuer Wirtschafts- und Organisationsformen. Aus dem Dargelegten kann unsere folgende Fragestellung zusammengefasst werden:

> ***Wie kann unter ausschließlicher Verwendung von FOSS die Thematik des Urbanen Gärtnerns bearbeitet werden und dabei mit und für die AkteurInnen ein WebGIS aufgebaut werden, das mit Nutzwert und Nachhaltigkeit den der vormals verfügbaren Angebote übersteigt?***

[4] "Die Bewältigung der auf die Menschheit zukommenden Probleme verlangt eine effektivere Ausschöpfung menschlicher Denkpotenziale weltweit. Das Internet erleichtert dies." (ebd.)

1.2 Zum Aufbau dieser Arbeit

Als ein erster Schritt wird in dieser Arbeit auf die Grundlagen zum Anwendungsszenario eingegangen, d.h. auf den Rahmen des Urban Gardening als anwendungsbezogener Gegenstand dieser Arbeit, etwaige Kooperationsverhältnisse die sich im Laufe der Arbeit aufgetan haben sowie die Datengrundlage. Anschließend wird auf die für die hier angewendete Arbeitsweise relevanten Grundlagen (internetfähiger) Geoinformationssysteme eingegangen. Hier soll es von der Einführung in die Entstehung von internetfähigem GIS über Lizenzen, Standards und Datenformate bis hin zu der Beschreibung von Geodateninfrastrukturen sowie Geographische FOSS (GFOSS) gehen. Aufbauend auf diesen Grundlagen wird im Hauptteil dieser Arbeit die Implementierung des WebGIS mit all seinen Bestandteilen erläutert um anschließend die Auf- und Weiterverarbeitung des Datensatzes zu beschreiben.

2 Anwendungsszenario : Urbanes Gärtnern in Berlin

Die Urbane Landwirtschaft, als theoretische Basis des Urbanen Gärtnerns, ist kein neuartiges Phänomen, sondern gehörte bereits viele Epochen zum Stadtbild. Erst mit der Industrialisierung und der neoliberalen Neuordnung der Gesellschaften wurde die regionale Selbstversorgung, in der das Umland die Stadt versorgte und Städte zu einem nicht geringen Teil aus Agrarflächen bestanden, von einem zunehmend globalisierten Lebensmittelmarkt abgelöst (MEYER-RENSCHHAUSEN 2011, BAIER 2010). Die Subsistenzwirtschaft und damit einhergehende Ernährungssouveränität (vgl. ALLMENDEKONTOR, ORANGOTANGO : 2012) spielte damit in der Großstadt nur noch eine untergeordnete Rolle.

Erst mit den ökologischen und ökonomischen Krisen der letzten Jahrzehnte rückte diese Fragen wieder in den Fokus der Stadtbevölkerung. Träger dieser Rückkehr der Landwirtschaft in die Stadt waren, sowohl im globalen Norden als auch im globalen Süden, die ökonomisch benachteiligten Stadtbewohner und Stadtbewohnerinnen (MÜLLER 2007). In den Gärten wird der Versuch unternommen Pflanzen auf unterschiedlichsten Arten auf sehr limitiertem und meist unkonventionellem Raum zu kultivieren (SCHARF 2011).

Den zu bepflanzenden Orten in der Stadt sind dabei kaum Grenzen gesetzt: Diese reichen vom einfachen Balkon über Hausdächer und Industriebrachen bis hin zu mobilen Behältnissen. Gepflanzt werden hauptsächlich Nutzpflanzen welche der Lebensmittelproduktion dienen. Im Gegensatz zu der bisherigen Interpretation des Gartens als „weltabgewandtes Refugium im Privaten“ (MÜLLER 2011:9ff), wird der Garten in Form urbaner Landwirtschaft auch als so genannter *Community Garden,* also Gemeinschaftsgarten, bezeichnet (MEYER-RENSCHHAUSEN 2011 : 4). Aus ihrer Entstehungsgeschichte heraus, welche sie als Orte der gemeinsamen Nahrungsmittelbeschaffung oder des gemeinsamen politischen Protests geprägt haben, wurden die Gärten zu Räumen des öffentlichen Lebens und Austausches.

Auch wenn, wie das Beispiel Berlin zeigt, schon länger verschiedene Formen urbaner Gärten z.B. in Form von Kleingartenkolonien oder in alternativen Wohnprojekten, bestehen, prägten sie die Innenstädte bisher weniger als öffentliche Plätze des gesellschaftlichen und kulturellen Austausches (ebd.). Ende der 1990er Jahre findet diese Entwicklung Einzug in die europäischen Großstädte und aktiviert Menschen sich diese Form des kulturellen Austausches und der Begegnung mit der Natur zu Nutze zu machen (SCHARF 2011 : 9, MÜLLER 2007 : 3). Der Fokus liegt hier aber vornehmlich auf karitativen, sozialen und ästhetischen als auf selbstversorgenden Zwecken (SCHARF 2011 : 3).

2.1 Zivilgesellschaftliche Stadtentwicklung

Die urbanen GärtnerInnen dürfen also durchaus als Teil einer zivilgesellschaftlichen und/oder partizipatorischen Stadtentwicklung betrachtet werden. Somit geht es mit diesem Projekt auch darum herauszufinden, wie Methoden der Geoinformatik insbesondere im Bezug auf *Open Source* (im Sinne von FOSS, aber auch als erweiterter Begriffsnexus) in einem sozialen Kontext einen Beitrag leisten können. Diesen anstrebend, begaben wir uns auf die Suche nach Kooperationspartnern, um die Wahrscheinlichkeit einer nachhaltigen Nutzung des Projektes zu erhöhen, um unsere begrenzten Ressourcen zu erweitern und um notwendige Daten zur Verfügung gestellt zu bekommen. Nach vielseitigem Interesse an unserer Idee seitens der AkteurInnen als auch von verschiedenen Forschungsvorhaben, kristallisierten sich schnell – zumindest anfänglich – sinnvolle Kooperationen heraus.

So verfolgt das Forschungsprojekt *Innovationsanalyse Urbane Landwirtschaft* (INNSULA) am Zentrum für Agrarlandschaftsforschung (ZALF) ein ähnliches Ziel, allerdings in größerem und

institutionalisiertem Rahmen. Es soll eine partizipative, deutschlandweite und zentrale Plattform für alle AkteurInnen von Gemeinschaftsgärten entstehen. Was in den Planungen dieser Plattform allerdings fehlte, war eine räumliche Darstellung der Gärten mit den entsprechenden Informationen. An diesem Punkt konnten wir mit unserer Initiative gut anknüpfen. Des weiteren existiert an der HU Berlin in Zusammenarbeit mit der Deutschen Bundesstiftung Umwelt ein Projekt über Wissenslandschaften im Kontext Urbaner Landwirtschaft. Auch hier besteht im Rahmen eines Dissertationsprojektes Bedarf an kartographischen Darstellungen.

Diese Arbeit beschränkt sich zunächst auf die Implementierung des WebGIS, welche im Anschluss sowohl dem ZALF, den AkteurInnen des Urbanen Gärtnerns in Berlin als auch dem Dissertationsprojekt zur Verfügung gestellt wird. Einzelne Aspekte der Kooperation und damit verbundene Herausforderungen werden im Verlauf dieses Textes stets am Rande präsent bleiben.

2.2 Daten

Das Kalkül der Kooperation mit zivilgesellschaftlichen Akteuren des Feldes “Urbane Landwirtschaft in Berlin” ist schließlich aufgegangen: Auf Basis eines Werkvertrages wurden wir gebeten zunächst einen technischen Prototypen der Karte anzufertigen, welcher als Grundlage für das weitere Vorgehen dienen sollte.

Die Beteiligten des Projektes INNSULA gestatteten uns daraufhin frühzeitig erweiterten Zugriff auf die im Aufbau bestehende Datenbasis des Stadtackers. Dieser erlaubt uns direkt auf unsere Bedürfnisse angepasste Tabellen aus der Datenbank zu generieren, um in konvertierter Form für die Weiterverarbeitung zur Verfügung zu stehen. Unser konzeptionell später Einstieg in die Entwicklung der Plattform und die technisch-administrativen Bedingungen verhinderten zuletzt jedoch die Integration der Karte.

Durch unsere beratende Funktion im INNSULA-Projekt konnten wir aber die Verwendung einer offenen Lizenzierung der publizierten Daten anregen. Dies garantiert je nach gewählter Lizenz eine breite Verwendung der angebotenen Datensätze, ohne in Konflikt mit dem geltenden Urheberrecht zu gelangen. Eine Quellenangabe und die Fortschreibung der Lizenz reichen meist aus, um Ableger oder sogar kommerzielle Anwendungen zu gestatten. Neben der verwendeten *Creative Commons (CC)* Lizenz (siehe auch 3.2) existieren weitere Modelle zur

Lizenzierung von Medien- und Forschungsdaten. Dank dieser Innovation bei der Freigabe von Forschungsdaten war es schließlich möglich die Entwicklung der Karte weiter voranzutreiben und konfliktfrei sowie produktiv mit dem Stadtacker an der Visualisierung der Daten zusammenzuarbeiten.

3 Einführung in internetfähige Geographische Informationssysteme

Geographische Informationssysteme (GIS) und internetfähige GIS (auch: *WebGIS*, *InternetGIS*, *OnlineGIS* oder *web mapping*; hier synonym verwendet) sind wie alle Technologien von soziokulturellen sowie technologischen Entwicklungen beeinflusst (SCHUURMAN 2004). Das bedeutet im Besonderen, dass GIS - so wie viele andere digitale Arbeitsweisen - von der Entwicklung hin zur *Informationsgesellschaft* sehr stark beeinflusst worden sind.
In den letzten Jahren fand dies durch die vermehrte Verwendung des Internets in der breiten Öffentlichkeit, außerhalb von akademischen oder technophilen sozialen Gruppierungen statt. Demnach können auch GIS und WebGIS heute nicht mehr wirklich getrennt gedacht werden.

Einem Überbegriff wie GIS oder WebGIS kann erst einmal nur eine sehr allgemeine Bedeutung zugeschrieben werden. So soll er auch hier nur die vielen verschiedenen Anwendungsbereiche und Funktionalitäten umschreiben, welche „Internet-Technologien verwenden und räumliche Informationen darstellen, verteilen und bearbeiten." (vgl. KORDUAN & ZEHNER 2008 : 7). Dies kann beispielsweise die weltweite Verfügbarkeit von Geodaten im Internet meinen, oder aber auch die Kombinationen von internetbasierten GIS-Werkzeugen in sog. *Mashups* (siehe 3.3). Deren Präsentation und Nutzbarmachung, bspw. in browserbasierten Anwendungen, ist zudem bereits für ein breites Publikum ausgerichtet. Diese Arbeit zeigt ein konkretes Anwendungsbeispiel in diesem Bereich auf. Es soll exemplarisch vorgeführt werden, was Internet-Geoinformationsysteme gegenwärtig leisten können und wie relevant sie für die weitere Entwicklung von GIS sind.

3.1 Entwicklung von InternetGIS

Inzwischen haben so gut wie alle weit verbreiteten Desktop GIS-Anwendungen eine Schnittstelle an das Internet bekommen, entweder für die direkte Einbindung von Geodaten aus dem Internet aber natürlich auch für die Programmentwicklung und das installieren von Updates (insbesondere bei Open Source-Tools).
Die heutige weltweite Verbreitung von einfachen InternetGIS Anwendung wurde vor allem durch die Annahme der Technologien von großen Unternehmen wie z.B. *Google*, *Yahoo* oder *Microsoft* ermöglicht. Diese stammen zwar ursprünglich nicht direkt aus der GIS-Community, aber deren inzwischen äußerst weit verbreiteten Kartenportale machen weltweite Kartendaten für jeden Internetnutzer – mit Einschränkungen – benutzbar und in einem gewissen Maße durch eine Programmierschnittstelle (API) auch programmierbar. Die Bedeutung dieser Dienste ist allerdings mitunter nicht unumstritten und es muss erwähnt bleiben, dass sie zunächst nur eine Kartendarstellung anbieten und keine vollständigen WebGIS-Werkzeuge oder -Datenquellen darstellen (BATTY et al. 2010 : 2).

Getrieben wird diese Entwicklung dennoch vor allem durch den verstärkten Einsatz von Karten in sozialen Netzwerken und anderen *Web 2.0* Anwendungen. Insbesondere durch Geokodierung, der Zuordnung von Straßenadressen oder Örtlichkeiten und Koordinaten, und mobile Geräte, wie Smartphones, Tablets o.ä., bekommt das Netz eine räumliche Komponente (KORDUAN & ZEHNER 2008 : 1).

Durch diese Entwicklungen wurde somit auch der Anwenderkreis der einfachen WebGIS-Operationen deutlich vergrößert und in einem gewissen Grade demokratisiert, da die gesunkenen technischen Anforderungen den Zugang für Nicht-GIS-Experten erleichtern. Dies hat auch das Ausmaß an verschiedenartigen Webanwendungen erhöht, weshalb heutzutage auf diesem wachsenden Markt sehr viele Anbieter von einfachen bis komplexen Anwendungen, sog. *Rich Internet Applications*, zu finden sind (vgl. KORDUAN & ZEHNER 2008 : 5). Es kann bereits von einer Art Paradigmenwechsel in der IT-Branche gesprochen werden, der sich im Übergang von der Desktop- (zurück) zur Thin-Client-Architektur zeigt: *Cloud Computing* und *Content Delivery Networks* haben zu einer weiteren Dezentralisierung der Internetressourcen geführt (ENDRAI et al. 2004 : 18*f*).

Kollaboration & Neogeographie

Zudem hat sich auch die kollaborative Arbeitsweise der SoftwareentwicklerInnengruppen durchgesetzt, die weg von einer hierarchischen Organisationsweise hin zu einer Heterogenisierung und selbstständigeren, aber synchronisierten Arbeitsweise verläuft. Symptomatisch spiegelt sich dies auch in den verwendeten Werkzeugen bzw. den implementierten Client-Server-Architekturen wieder. Dabei kann ein Übergang von einem eher geschlossenen, monolithischen GIS hin zu einem flexibleren, offeneren und interoperableren GIS beobachtet werden. Die damit einhergehende Ausrichtung auf ein (potentiell) breiteres Publikum eröffnet ein Feld, das inzwischen ideologisch aufgeladen ist und in dem Partizipationsmöglichkeiten, Privatsphäre, Urheberrecht und der freie Informationsfluss miteinander abgewogen werden müssen (LI et al. 2011 : 7–8).

Als wichtiger Teil dieses Paradigmenwechsels und als deren Umsetzung kann die Implementierung von (mittlerweile sehr leistungsfähigen) *Open Source* Werkzeugen und der Entwicklung von verschiedenen neuen Standards angesehen werden (siehe 3.4). So finden Geodaten und die Methoden der Verarbeitung in den neuartigen GIS der letzten zehn Jahre eine immer weitere Verbreitung und neue Anwendungsgebiete.

Gleichzeitig bringt diese Entwicklung in der Kartographie und der Geographie, welche auch als „Neogeographie“ bezeichnet wird (HAKLAY 2008 : 2020ff), einige Herausforderungen mit sich. Diese können zum Teil sozialer Natur sein, wie die Fragen der Sicherheit, des Datenschutzes (sprich des Schutzes von Personen und ihrer Privatsphäre gegenüber dem Missbrauch von Daten), Urheberrechten und Informationsfreiheit, oder aber auch technische, wie die der Kompatibilität und Programmentwicklung (LI & GONG 2008 : 7). Der Fokus liegt in dieser Arbeit zunächst auf der technischen Umsetzbarkeit und Architektur einer WebGIS-Applikation, weshalb auf die Sicherheit und Urheberrechtsproblematik in dieser Arbeit nicht eingegangen werden kann. Hierbei kommt den Spezifikation und Standardisierung von Anwendungskonzepten und Austauschformaten eine zentrale Rolle zu (ALIPRANDI 2011).

Standardisierung

Mit der Gründung des Open Geospatial Consortium (OGC) im Jahre 1994 wurde bereits eine Internationale Institution für interoperable Geoinformationsdienste geschaffen, die es zum Ziel hat Geoinformationen und Dienste für verschiedenste Anwender nutzbar zu machen. Dafür arbeitet die Organisation an dem Ziel „Grundlagen für einheitliche und interoperable

Schnittstellenspezifikationen zu erstellen, die weltweit frei verfügbar und kostenfrei nutzbar sind“ (vgl. BERNARD et al. 2005 : 9). Das Konzept der *Freien Software,* oft fälschlicherweise mit *Open Source* synonym verwendet, legt in diesem Sinne wert auf die essentiellen Freiheiten (im sozialen Sinne) aller Benutzer Veränderungen am Programm vorzunehmen und an eigene Bedürfnisse anzupassen. *Open Source* dagegen ist zunächst nur eine Variante der Softwareentwicklung, die nicht zwangsläufig in *Freier Software* mündet. Aufgrund der inzwischen sehr leistungsstarken (und damit häufiger benutzten) Programme und des etwas griffigeren Begriffs, hat sich *Open Source* trotzdem als allgemeine Bezeichnung weitestgehend durchgesetzt.

Die ideelle Basis im Bezug auf die individuellen Freiheiten bei der Nutzung und Entwicklung von Software wird besser mit dem Begriff *Freie Software* erfasst (vgl. STALLMAN 2010 : 3–7). Der Aufschwung, den diese Entwicklungssparte erfahren hat, ist zu einem Teil der Konkurrenzfähigkeit gegenüber proprietären Programmen geschuldet, zum anderen Teil aber auch mit der zu Grunde liegenden libertären Ideologie zu erklären. Die besondere lizenzrechtliche Absicherung sowie die Kollaboration und Kooperation internationaler Entwickler fördert eine kreative und respektvolle Zusammenarbeit.

Dabei konnte auch gezeigt werden, wie unter Ausnutzung der vorhandenen Netzwerktechnologien komplexe Entwicklungsprozesse (geographisch und zeitlich) verteilt gesteuert und durchgeführt (vgl. HEINRICH & HETTEL 2012) werden können:

> *“Contributions such as edits to a wiki page, or “commits” to a version control system, cannot exist outside of the context in which they are made. A relationship to this context requires a coordinating mechanism that is an integral part of the initial production process. These mechanisms of coordination and governance can be both technical and social.”*
>
> Collaborative Futures. Coordinating Mechanisms create Contexts. (ZER-AVIV, LINKSVAYER, MANDIBERG, PEIRANO, TONER, HYDE, KANARINKA, TARKA, TAYLOR : 2010)

Offen vs. Proprietär

Diese ideologische Fundierung der mit *Freier und Open Source Software* verbundenen Wirtschaftsformen ist an sich nicht selbstverständlich, denn die o.g. prägenden kommerziellen WebGIS Anbieter sind traditionell bspw. sehr daran interessiert ihre Produkte und Daten möglichst gewinnbringend zu vermarkten. Ein gesamtgesellschaftlicher Mehrgewinn im Sinne von Informationsfreiheit ist dagegen nicht unbedingt das vorrangige Ziel dieser monetär orientierten Strategie.

Diesen kommerziellen Angeboten ist gemein, dass sie dem Nutzer nur sehr eingeschränkte Rechte einräumen. Schon der Ausdruck eines Kartenausschnittes für ein öffentliches Prospekt kann rechtlich problematisch werden, ebenso wenn die API außerhalb der Lizenz für externe Projekte verwendet wird. Das macht deutlich, dass die zu Grunde liegende Technik und die Rechtslage dabei von zentraler Bedeutung sind und deren Nutzung maßgeblich beeinflussen.
Trotz dieser Problematiken, aber auch auf Grund ihrer Vorteile, hat die Entwicklung von FOSS-Entwicklungsmodellen stark an Zulauf gewonnen. So gibt es beispielsweise mit dem Kartendienst OpenStreetMap (OSM) inzwischen ein durchaus konkurrenz- und leistungsfähiges Angebot, das ausschließlich auf benutzergenerierten Kartendaten basiert und diese kostenfrei unter einer offenen Lizenz veröffentlicht (OPEN DATA COMMONS 2013b).

Diesen Ansätzen ist die Ausrichtung auf kooperative Gestaltung öffentlicher, gemeinschaftlicher Angelegenheiten und Güter gemein. Die Chancen in dieser betreffen ebenso jene *Open* Begriffspaare, welche sich bereits in ihrer lexikalische Form explizit an die Idee von *Open Source* anlehnen: *Open Access*, *Open Data*, *Open Content* oder auch abstrakter *Open Knowledge*, *Open Governance* oder *Open Society*.
Aber auch verwandte und ähnliche Diskurse wie solche um *Liquid Democracy*, *Peer to Peer* und *Crowdfunding* oder *Basisdemokratie*, *Partizipation* und *Nachhaltigkeit* spielen mit hinein (DEMAREST, MINOD, RICHTER 2012 : 23).

3.2 Lizenzen

Es gibt mittlerweile neben den klassischen Copyright Lizenzen und Standards der proprietären Programme, die deren Bearbeitung und Weiterverbreitung zum allergrößten Teil ausschließen, die sog. *offenen* Lizenzen und Standards. Da dies ein sehr weites Feld ist, kann hier nur eine Auswahl vorgestellt werden, im Besonderen jene der für das Anwendungsbeispiel relevanten.

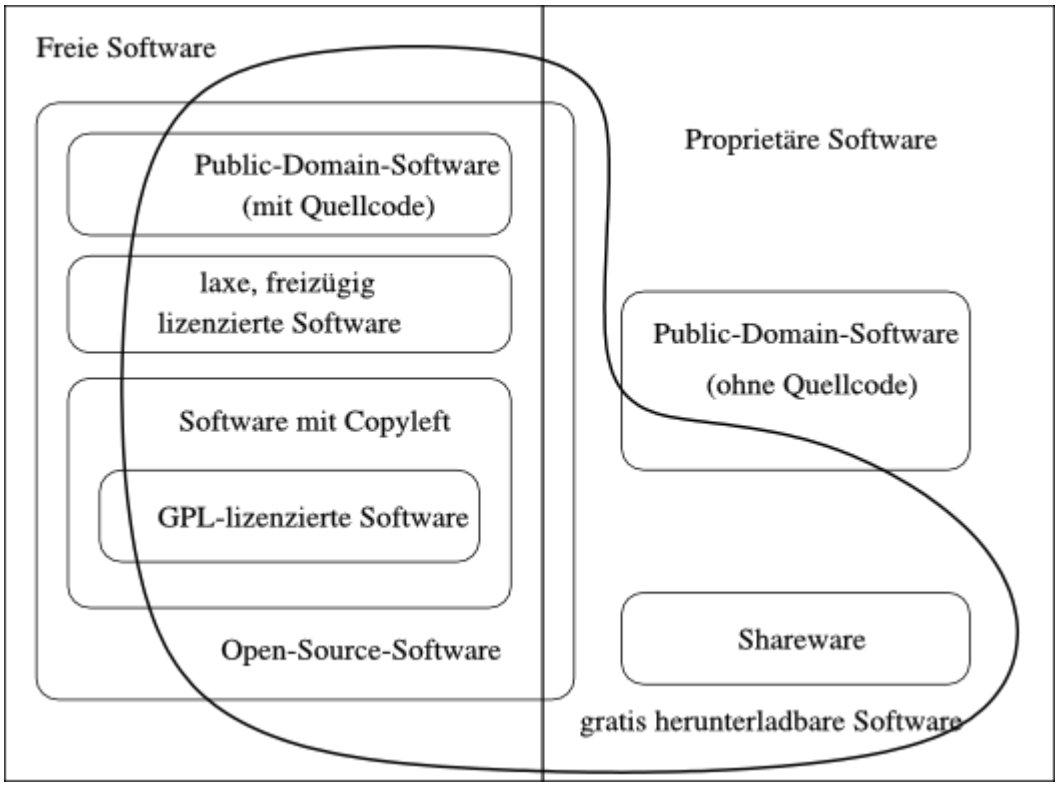

Abb. 1: Diagramm (ursprünglich von Chao-Kuei und seitdem von mehreren anderen aktualisiert) zeigt verschiedene Softwarekategorien. *http://www.gnu.org/philosophy/categories.html* (02.01.2013)

Wie aus dem Diagramm hervorgeht, ist besonders die Freie Software sehr vielfältig im Bezug auf Lizenzen. So zählt die *Open Source* Initiative (OSI) bereits 70 verschiedene Lizenzen[5], die nach ihren Regeln als *Open Source* (also nicht unbedingt *frei*) gelten können. Im Allgemeinen wird darum dann eher von *Free and Open Source Software* (FOSS) gesprochen. Im Fall einer WebGIS-Infrastruktur betrifft dies genauer die Lizenzierung von Software, Medien und Datensätzen.

Im Idealfall wird die zu Grunde liegende Datenbank eines WebGIS unter der ODbL 1.0 (Open Database License) und die Inhalte dieser unter der DbCL 1.0 (Database Contents License) lizenziert. Diese erlauben die Weitergabe der Datenbestände unter gleichen Bedingungen bei Berücksichtigung bestimmter Rechte des ursprünglichen Erzeugers. Analog zur *Creative Commons* Lizenz, welche in den frühen 2000er Jahren als Antwort auf das Bedürfnis nach *frei*[6] verteil- und verfügbaren Medien (Text, Ton, Bild, Video, etc.) entstand, wurde die ODbL in

[5] *http://opensource.org/licenses/alphabetical* (04.01.2013)

[6] "To understand the concept, you should think of 'free' as in 'free speech' not as in 'free beer'." (FREE SOFTWARE FOUNATION 2013)

Abgrenzung von bereits verfügbaren Lizenzen gestaltet. Durch die besondere Anwendungssituation, bspw. die Trennung von Inhalt und Behälter ebenso wie die zu erwartenden häufigen Wiederverwendungen von Daten(banken), mussten beim Verfassen der Lizenz diese Besonderheiten mit beachtet werden (OPEN DATA COMMONS 2012a).

Ursprünglich durch Open Source Softwarelizenzen inspiriert, es seien lediglich die bekanntesten wie BSD, GPL, MIT oder Apache License erwähnt, übernehmen diese Lizenzen die Idee des Teilens und Verteilens von digitalen Produkten und bewahren bestimmte Rechte des Originalautors. Diese sind zumeist *Namensnennung, Weitergabe unter gleichen Bedingungen, Nicht Kommerziell* oder *Keine Derivate* die außerdem unterschiedlich kombiniert sein können. Somit befähigt und ermutigt die ODbL Wissenschaftler dazu, sich dem weltweiten Publikum zu öffnen und Abwandlungen und Bearbeitungen der eigenen Arbeit zuzulassen.

Im Falle eines WebGIS - würden unserer Ansicht nach - idealerweise die dargestellten Webseiten, Bilder, Texte und Videos, aber auch Kartenkacheln oder andere Derivate, bspw. in diesem Projekt unter einer *Creative Commons* Lizenz stehen. Die darunter liegenden Programme stehen jeweils unter einer spezifischen Lizenz, die zunächst auf Kompatibilität mit den verwendeten Lizenzen aller anderen verwendeten Programme überprüft werden muss. Das bedeutet, dass die Lizenzen individuell ausgewählt werden, je nachdem wie viel vom Produkt freigegeben werden soll. Nicht zuletzt spielt dabei auch eine Rolle, welche Werkzeuge bereits verwendet wurden, da diese oftmals die Art der Weitergabe vorschreiben (z.B. Weitergabe unter gleichen Bedingungen). Auch wenn die meisten Lizenzen untereinander kompatibel sind, sollte ein immer besonderer Augenmerk auf den Lizenzen liegen, vor allem wenn mehrere Werkzeuge in einer WebGIS-Architektur verbaut sind und diese mit unterschiedlichen Lizenzen arbeiten (STALLMAN 2006).

3.3 Mashups

Der Begriff Mashup ist als Beschreibung für die Kombination von verschiedenen Werkzeugen und Datenquellen in einer Softwarearchitektur zu verstehen und besteht in diesem Zusammenhang erst seit 2004, wobei die Karten-Mashups dabei mit die wichtigste Rolle spielen. Die Idee dahinter korrespondiert dabei mit den neuen Werten des sog. *Web 2.0*, die sich auf das interaktive Teilen von Informationen, Interoperabilität, nutzerorientiertes Design und Kollaboration beziehen (BATTY et al. 2010 : 1). Der Begriff stammt an sich aus der

Remix-Kultur der Musik, wo verschiedene Musikstücke und -stile unterschiedlicher Machart zu einem neuen Stück zusammengesetzt werden.

Die Idee ist auch nicht neu. SoftwareentwicklerInnen sind ständig damit konfrontiert verschiedene externe Funktionen und Datenquellen mit einzubeziehen oder zu referenzieren. Der Grund warum Mashups neuerdings an Popularität gewonnen haben, daraus sogar eine Art Hype entstanden ist, liegt wohl darin, dass auch nicht-technisch-versierte Menschen Zugang zu solchen Werkzeugen ermöglicht wurde. So kann jeder Mensch mit Internetzugang heute eigene Mashups erstellen ohne z.B. gleich eine gesamte Programmiersprache lernen zu müssen (KOSHMIDER et al. 2009 : 1). Gleichzeitig ist der Begriff aber sehr heterogen zu verstehen und beschreibt ähnlich dem Begriff der *Neogeographie* ein Feld von Techniken und Methoden, das sich von traditionellen GIS-Praktiken abhebt, indem es unprofessionellen Nutzern ermöglicht durch Kombination verschiedener Bausteine selber Karten zu erstellen (TURNER 2006 : 3, MCCONCHIE 2008 : 29).

Diese Entwicklung ist darauf zurückzuführen, dass die Möglichkeiten und der Datenaustausch im Internet immer einfacher und schneller funktionieren. Des weiteren sind große kommerzielle Internetunternehmen den Trend, mit einem besonderen Fokus auf Karten-Mashups mitgegangen, welche 2008 etwa 40% aller Mashupvarianten darstellten (LI & GONG 2008 : 640) . Als Grundlage für diesen Trend dienen zusammengefasst kostenfrei verfügbare Kartengrundlagen sowie teilweise offene APIs, die browser- und dienstbasiert funktionierten.

Trotzdem ist das Potential in dieser Sparte noch nicht ausgeschöpft und es gibt weiterhin noch offenen Raum für Kombinationen, den besonders private, nutzerorientierte (z.B. FOSS) oder auch staatliche Initiativen (Stichwort Open Data, GDI - Kapitel 3.5) füllen sollten und können (vgl. LI & GONG 2008 : 639). Einen Überblick über die aktuell meist verwendeten Mapping APIs gibt die Abbildung 2.

Sehr augenfällig ist dabei die marktbeherrschende Position von Google und anderen großen Internet- und Softwareunternehmen. FOSS-Projekte wie OpenStreetMap stellen im Vergleich einen verschwindend kleinen Teil dar.

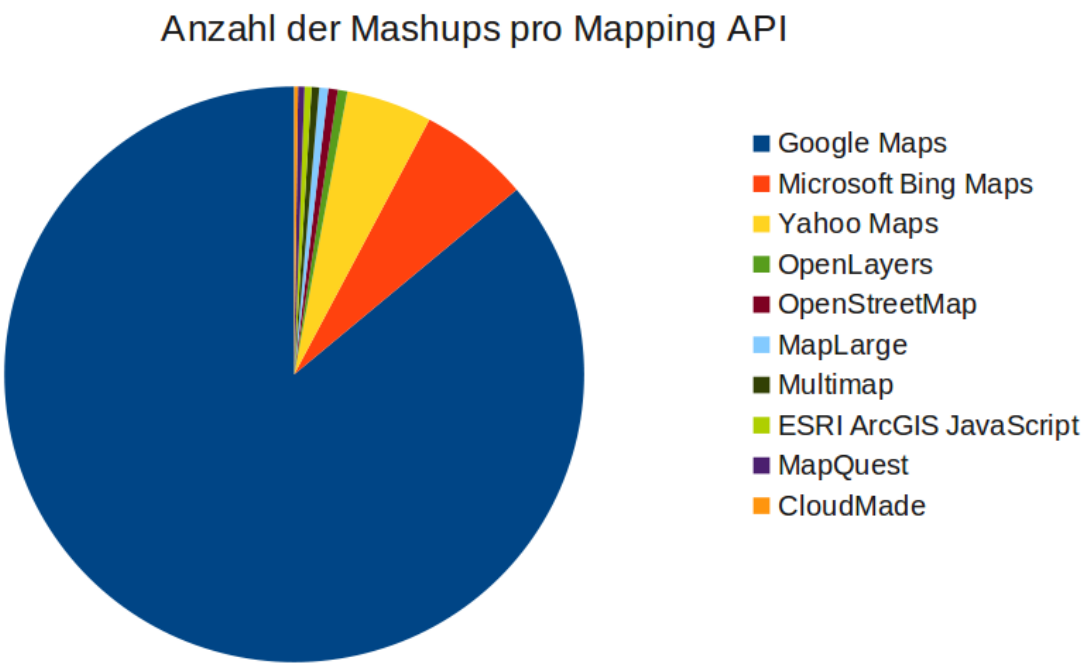

Abb. 2: API Auswahl auf Grundlage des Verzeichnisses auf *programmableweb.com* (29.12.2012). Eigene Abbildung.

Ein klassisches Karten-Mashup besteht typischerweise zunächst aus einer Geodatengrundlage, einem Geocoder, bzw. einer Suchfunktion, und einem Webinterface. Die vorrangige Verwendung der Karten-Mashups ist die Präsentation für ein breites Publikum. Der Wiederverwendung der Daten bzw. der Anbindung an den eigenen Server durch professionelle EntwicklerInnen mit Hilfe einer API kommt dabei trotzdem eine Schlüsselrolle zu (LI & GONG 2008). Das Potential darin, besonders im FOSS Bereich, ist besonders in der plattformübergreifende Interoperabilität zu sehen.

Wie bereits erwähnt, ist das Feld der Mashups sehr vielfältig und flexibel, sodass der Begriff nicht einfach zu fassen ist und weiterer Klassifikation bedarf (KOSHMIDER et al. 2009 : 2). Als Klassifikation könnte man z.B. anhand der Funktionen für den Endnutzer unterscheiden in *Informative Mashups* für hauptsächlich Präsentationsfunktionen, *Partizipative Mashups*, die Interaktionen der Nutzer zulassen und *Kollaborative Mashups*, die sogar neben der zwei-wege Interaktion auch noch weiteren Austausch und Einbindung erlauben (LI & GONG 2008 : 641).

Es bleiben bei jedem Mashup der Grad der Interaktion, die Teilhabe an den Daten und Möglichkeiten zur Weiterverwendung sehr wichtig. Dies wird auch an unserem Beispiel

Stadtacker deutlich. Im Folgenden soll der Mashupbegriff ohne Festlegung auf ein bestimmtes Modell als theoretisches Grundgerüst dienen, während der Fokus weiterhin auf der Anwendung einzelner, beispielhafter Technologien und der Client-Server-Architektur liegt.

Zum Verständnis sollte man sich dabei klarmachen, dass Webseiten nicht analog zu Buchseiten gesehen werden können, sondern im Prinzip kleine Programme darstellen. Diese können wiederum Funktionen ausführen, wovon die Anzeige auf dem Bildschirm nur eine davon, in einer Kette von Operationen und Abfragen, ist. Mashups machen sich dies zu Nutze, indem sie die Abfragen mit denen von anderen Websiten bzw. Datenbanken verknüpfen. So sind heutzutage die meisten Mashups browserbasiert. Im Gegensatz dazu hat die Serverseite als *gestaltende Kraft* eher abgenommen und übernimmt die nicht weniger wichtige Aufgabe der *Bereitstellung*. Eine treibende Kraft dafür ist das reichhaltige Angebot an Daten und Dienstleistungen und die Popularität der in Browsern benutzten relativ einfachen Skriptsprache JavaScript (FU & SUN 2011 : 94) und ihre Verwendung für AJAX (Asynchronous JavaScript and XML).

Des weiteren basieren die Webdienste in einem Mashup auf einer Server-Orientierten-Architektur (SOA). Die Nutzeranfragen können dabei über das Simple Object Access Protokoll (SOAP) koordiniert werden, einem sprachen- und plattformunabhängigen Protokoll, basierend auf HTTP (HyperText Tranfer Protocol) und XML (eXtensible Markup Language). D.h. in SOAP-Anwendungen bietet ein Server Dienstleitungen mit XML Spezifikationen an und der Nutzer schickt (über den Browser) Anfragen für einen Dienst oder eine, ebenso in XML kodierte, Informationsabfrage. Im Kontext einer heterogenen interoperablen Softwarearchitektur wird klar, dass es wichtig ist solche und ähnliche multikompatible Standards zu schaffen, damit dienstübergreifende Interaktionen fehlerfrei und möglichst schnell ablaufen können (vgl. LI et al 2011 : 17).

3.4 Standards und Datenformate

Der WFS (Web Feature Service) beispielsweise ist ein vom Open Geospatial Consortium (OGC) erstellter Standard, um geographische Merkmale abzufragen der SOAP- sowie auch konventionelle HTTP-Abfragen unterstützt. Ähnlich sind die ebenfalls vom OGC entwickelten Standards WMS und WCS, die es ermöglichen nicht-editierbare Kartendaten (Raster) abzufragen. In diesem XML-orientierten Zusammenspiel stehen dann die APIs der Geodatenanbieter und eröffnen so ein weites neues Anwendungsfeld für den Nutzer (LI et al 2011 : 17).

Insbesondere die XML-orientierten, offenen Standards erfreuen sich hier großer Beliebtheit, da sie auch die Integration in DesktopGIS-Anwendungen ermöglichen. Ein Beispiel hierfür sind GML und KML, beides vom OGC empfohlene Formate, die es ermöglichen Geodaten im Raster- und Vektor-Format (und Informationen zur 3D-Visualisierung) auszutauschen. Jedoch besteht auch hier noch Koordinierungsbedarf, weshalb das OGC im Jahr 2009 eine Arbeitsgruppe eingesetzt hat, um zukünftige Spezifikationsproblemen vorzubeugen (LI et al 2011 : 19). Der Unterschied zwischen GFOSS und unternehmensgeleiteten WebGIS Applikationen wird in einem Beispiel besonders deutlich: So scheinen OpenLayers (siehe *Abschnitt 4.7*) und die Google Maps Bibliothek auf den ersten Blick sehr ähnlich und beide sind auch mit KML kompatibel. Wenn aber Vektordaten als Overlay über eine Google Karte gelegt werden sollen und diese von einer anderen Website stammen, kann diese nicht im Browser angezeigt werden. Der Grund dafür ist ein KML Proxy, der keine anderen Datenquellen als die eigenen zulässt. Solche Einschränkungen gibt es bei der auf OGC Standards beruhenden FOSS API von OpenLayers nicht (BATTY er al 2010:4).

Ein weiterer erwähnenswerter Standard ist GeoJSON, der nicht auf XML basiert, sondern an JSON (JavaScript Object Notation) angelehnt ist. Im Unterschied zu XML-basierten Standards ist dieses Dokumentenformat von Maschinen und Menschen deutlich intuitiver zu lesen, kompatibel mir sehr vielen Skriptsprachen und plattformunabhängig. Als relativ neues, einfaches und effizientes Format wird GeoJSON als sehr zukunftsfähig angesehen, weshalb es in unserem Beispiel auch versuchsweise implementiert wurde (LI et al 2011 : 23).

3.5 Geodateninfrastrukturen

Ein weiterer Begriff aus dem Feld des WebGIS sind die Geodateninfrastrukturen (GDI) (eng.: spatial data infrastructure - SDI). GDI sind inzwischen ein wichtiges Infrastrukturelement der Informationsgesellschaft geworden. An sich dem des Mashups ähnlich, beschreibt der Begriff ein weites Feld von Nutzern, Netzwerken, Regeln, Standards und (Geo-)Daten, allerdings weniger mit direktem Anwendungsbezug, als aus strukturell-planerischer Sicht. Es gibt bereits auf verschiedenen räumlichen Ebenen staatliche, kommerzielle und non-profit GDI-Initiativen mit dem Ziel der verbesserten Koordination und dezentralen Organisation von Geodaten.

Eine GDI ist demnach keine direkte Ablösung der bestehenden GIS- und WebGIS-Strukturen, sondern vielmehr eine Ergänzung und Umstrukturierung. Der Fokus liegt dabei auf der

räumlichen Erfassung von Geodaten, deren Modellierung und Auswertung mit Hilfe von kollaborativen Vernetzungsmöglichkeiten und WebGIS-Architekturen. Das besondere dabei ist die neue Ausrichtung nicht mehr nur auf ausgewiesene GIS-Experten, sondern auch an Anwender aus diversen Bereichen, um diese mit Werkzeugen und Plattformen bei der Nutzung und Publikation von Geodaten zu unterstützen (vgl. BERNARD et al. 2005 : 3–5).

> *Die Qualität der Geodateninfrastruktur Deutschland (GDI-DE) hängt vor allem vom Grad der Zugänglichkeit vorhandener Geodaten ab. Aus technischer Sicht ist der Schlüssel für die Zugänglichkeit die interoperable Bereitstellung der Geodaten durch die Einhaltung offener Standards.*
>
> BUNDESAMT FÜR KARTOGRAPHIE UND GEODÄSIE 2013

Im Mittelpunkt der GDI steht der Versuch neue interoperable Systeme zu schaffen, die auch die Kooperation und Entwicklung in verschiedenen Gruppen ermöglichen. Auch wenn das in der Realität oft komplexer ist als erwartet und die Standardisierungsgremien sich gegenseitig die Arbeit schwer machen (oftmals zur Stärkung der eigenen Marktposition – siehe proprietäre Datenformate), gibt es ebenfalls FOSS-Bemühungen, beispielsweise beim Open GIS Consortium oder der ISO, „eine Imformationswelt zu schaffen, in der jedermann Geoinformationen und Geodienste über Netzwerk-, Applikations- und Plattformgrenzen hinweg nutzen kann“. (vgl. BERNARD et al. 2005 : 9)

3.6 Geospatial Free and Open Source Software (GFOSS)

Das Feld an GFOSS (Geospatial Free and Open Source Software) Werkzeugen ist im Laufe der Jahre auf Grund der Vielzahl und weltweiten Verteilung der Mitarbeiter der unterschiedlichen Projekte sehr breit aufgestellt. Es gibt Ansätze von Desktop GIS über kleinere, spezifische Programmbibliotheken bis hin zu vollständigen Client-Server GIS mit Datenbankanbindung.
Diese Diversität an Möglichkeiten ist zu begrüßen. So lassen sich für unterschiedliche Anwendungsszenarien, bei genügender Kenntnis der Angebote, schnell passende Programmpakete finden, welche eine zielführende Unterstützung in der Entwicklung einer GFOSS Anwendung sind.

Die Auswahl kann dabei immer nur eingeschränkt sein, da allein die Vielzahl an möglichen Programmiersprachen bei der konkreten Entwicklung eine Entscheidung bewirkt. Schließlich stehen in realweltlichen Arbeitsumgebungen nur begrenzte Ressourcen zur Verfügung. Die zur Verfügung stehende Menge der *gesprochenen* Programmiersprachen einer Entwicklerin kann eben auch nur begrenzt sein.

So lässt sich in der Szene der GFOSS EntwicklerInnen eine professionelle Ausrichtung erahnen. Einzelne Projekte leisten sich Vollzeitprogrammierer und werden mitunter von kommerziellen Anbietern finanziert. Dies muss jedoch kein Widerspruch zum FOSS Gedanken sein, da viele in diesem Bereich engagierte Firmen ihren Umsatz nicht mit dem Verkauf der Software, sondern mit der Implementierung bestimmter Anwendungsszenarien und anschließenden Supportverträgen verdienen. Auch finden sich häufiger Hostingangebote, bei denen ein Anbieter die gesamte Infrastruktur eines WebGIS zur Verfügung stellt und ausschließlich personalisierte Zugänge vermarktet.

Ebenso findet Crowdfunding seine Anwendung. Aus der FOSS Welt bekannt sind auch sogenannte *Code Sprints* in welchen sich einzelne Programmierer für einen bestimmten Zeitraum, üblicherweise eine Woche, physisch treffen und gemeinsam an vorher spezifizierten Aufgaben arbeiten. Dies erhöht die Arbeitsmoral und fördert den sozialen Zusammenhalt in der Gruppe.

Bedeutsam sind daher auch die Konferenzen FOSSGIS (für den deutschsprachigen Raum) und FOSS4G (für die internationale Community), welche den EntwicklerInnen jährlich Anlaufstellen und physische Diskussionsräume zur Aquise, gegenseitigem Wissensaustausch und zusammen Spaß haben bieten[7]. Diesen gemeinschaftlichen Entwicklungsvorgang im Kleinen bei der Entwicklung der Gartenkarte abzubilden, erlaubte es einen Einblick in die Welt der Softwareentwicklung zu erhaschen. Welchen Einfluss die Wahl der WebGIS-Software schließlich auf unser Endprodukt hatte, wird im nächsten Abschnitt deutlich.

[7] "Neu ist, dass die FOSSGIS-Konferenz im Jahr 2013 von Mittwoch bis Freitag stattfinden wird. So soll der Open-Source- und der OpenStreetMap-Community ein Besuch der Konferenz erleichtert werden. Zudem ist geplant am nachfolgenden Wochenende weitere Community-Treffen zu organisieren." FOSSGIS 2013.

4 Implementierung des WebGIS

Der Implementierung des *Urbane Gärten in Berlin WebGIS* ging eine umfassende Recherche zu GFOSS Werkzeugen voraus. Diese war notwendig um Anforderungen und Möglichkeiten der technischen Bedingungen auszuloten. Denn WebGIS existieren heutzutage in einer schier unüberschaubaren Vielfalt und für die verschiedensten Anwendungsgebiete: GDI Verwaltungsdienste stehen neben integrierten Desktopanwendungen, welche das schnelle und unkomplizierte Veröffentlichen überschaubarer Datenmengen gestatten; öffentliche kollaborative Karten stehen neben den Kartenportalen der großen kommerziellen Anbieter.

Unsere Aufgabe bestand nun darin das für unseren Anwendungsfall richtige Werkzeug zu finden. Zur Durchführung stand ein privater, gemieteter, virtueller Server zur Verfügung und konnte zunächst als Experimentier- und Veröffentlichungsplattform verwendet werden.

Dieser Abschnitt beschreibt den Arbeitsablauf von der Konzeption bis zum Endprodukt Da das Feld aber auch für uns einiges an Neuland bedeutete, gingen den ersten Experimenten auf dem Server eine umfassende Marktschau und darauf aufbauend Überlegungen zu Implementierung und Kartengrundlage voraus. Diese Vorbereitungen führten schließlich zur Installation eines Mapservers, der Verwendung von externen Datenquellen und abschließend der Konfiguration der Benutzerschnittstelle.

4.1 Marktschau

Diese soll hier kurz umrissen werden (in Klammern Erklärungen oder Beispiele) und gliedert sich grob in die folgenden vier Felder:

1. **Grundlagenberich :**
   Datenquellen, Bibliotheken, Datenformate, semantische Abfragesprachen, OGC Standards und OSGeo Bibliotheken
2. **Serveranwendungen :**
   Datenbanken, integrierte Anwendungen, auf dem OpenGIS Web Services (OWS) OGC Standard basierende Raster- und Vektordatendienste sowie Geolokalisierungs- und Katalogdienste

3. **Clientanwendungen :**
   JavaScript Benutzerschnittstellen (User Interfaces / UI) / Clients für OWS
4. **Implementierungen :**
   Cloud Computing Angebote, ungewöhnliche Mashups, Visualisierungen und 3D

Abschnitt eins, *Grundlagenbereich*, beschäftigt sich mit den zu Grunde liegenden Technologien von WebGIS. Besondere Erwähnung finden hierin Open Data Quellen, Umrechnungs- und Datenzugriffsbibliotheken (Proj.4, GDAL/OGR), einzelne freie Datenformate (GML, GeoJSON) sowie die Open Geospatial Consortium (OGC) Web Service Framework (WSF) Spezifikation.

Abschnitt zwei, *Serveranwendungen*, listet fortgeschrittene Umsetzungen der OGC WSF Spezifikation auf. Dies sind Datenbanken nach dem SQL (PostGIS) und NoSQL (GeoCouch) Schema, integrierte WebGIS Anwendungen (OpenGeo Suite, OSGeo Web Map Services), Ergänzungsdienste zum WMS (QGIS Server, Pyramidencaches und kartographische Styles) als auch Geolokalisierungs- und Katalogdienste, welche im Besonderen für die Umsetzung von GDI-DE- oder INSPIRE-konforme Angebote benötigt werden.

Abschnitt drei, *Clientanwendungen*, behandelt auf dem Markt verfügbare JavaScript-Bibliotheken, welche einen komfortablen Zugriff auf OWS bieten. Sie gliedern sich in jene mit erweiterter Bearbeitungsfunktionalität, reine Darstellungsbibliotheken und welche für reine (SVG basierte) Vektorkarten. Außerdem finden Datenanalyse (GeoTrellis, R) und Desktop GIS (GRASS, QGIS) Erwähnung.

Abschnitt vier, *Implementierungen*, erwähnt konkrete Onlineanwendungen im GFOSS Umfeld. Dazu gehören Cloud Computing Dienste zum ‘web mapping’ (MapBox, Cloudmade), spannende Mashup Beispiele (Ushahidi, Shareabouts) sowie kleinere Produktivitätswerkzeuge für den GFOSS Alltag (Slippy Maps Kartenkachelspezifikation, Google-, TMS- und QuadTree-Koordinaten und -Zoomstufe in WGS84/Spherical Mercator Umrechnung, GeoTIFF/-JPG/-PNG/-GIF World File Erstellung).

4.2 Implementierung

Auf Basis der aus der Recherche gewonnenen Erfahrungen entschieden wir uns für eine möglichst simple Architektur des Grundsystems. Es wurden eine Daten-, Dienst- und Präsentationsschicht definiert:

In der Datenschicht werden Dateien jeglicher Art im Dateisystem abgelegt. In unserem Beispiel handelt es sich Skripte, Dokumente, (georeferenzierte Raster- und Vektordaten, ein sog. MAPFILE sowie die Datenbankdateien. Auf der Dienstschicht bieten Serverprogramme normgerechte Schnittstellen zur Abfrage der einzelnen Dienste. Dies sind ein Terminalserver, um auf der lokalen Konsole des Servers arbeiten zu können, der Webserver, welcher die Beantwortung von Hypertextanfragen übernimmt und gleichzeitig für Vermittlung zum WMS verantwortlich ist. Letzteres besitzt weiterhin eine Anbindung an die räumliche Datenbank, die zur dynamischen Datenablage dient., Der UMN Mapserver würde diesen WMS Dienst bereitstellen, welcher unsere Geodaten publiziert.

Eine OpenLayers (s.u.) Instanz im Browser des Besuchers koordiniert abschließend in der Präsentationsschicht alle notwendigen Abfragen und stellt das Ergebnis dar. Kommandozeile und Datentransfertools dienen ferner der umfassenden Manipulierung des WebGIS. Eine vereinfachende Darstellung findet sich in Abbildung 3:

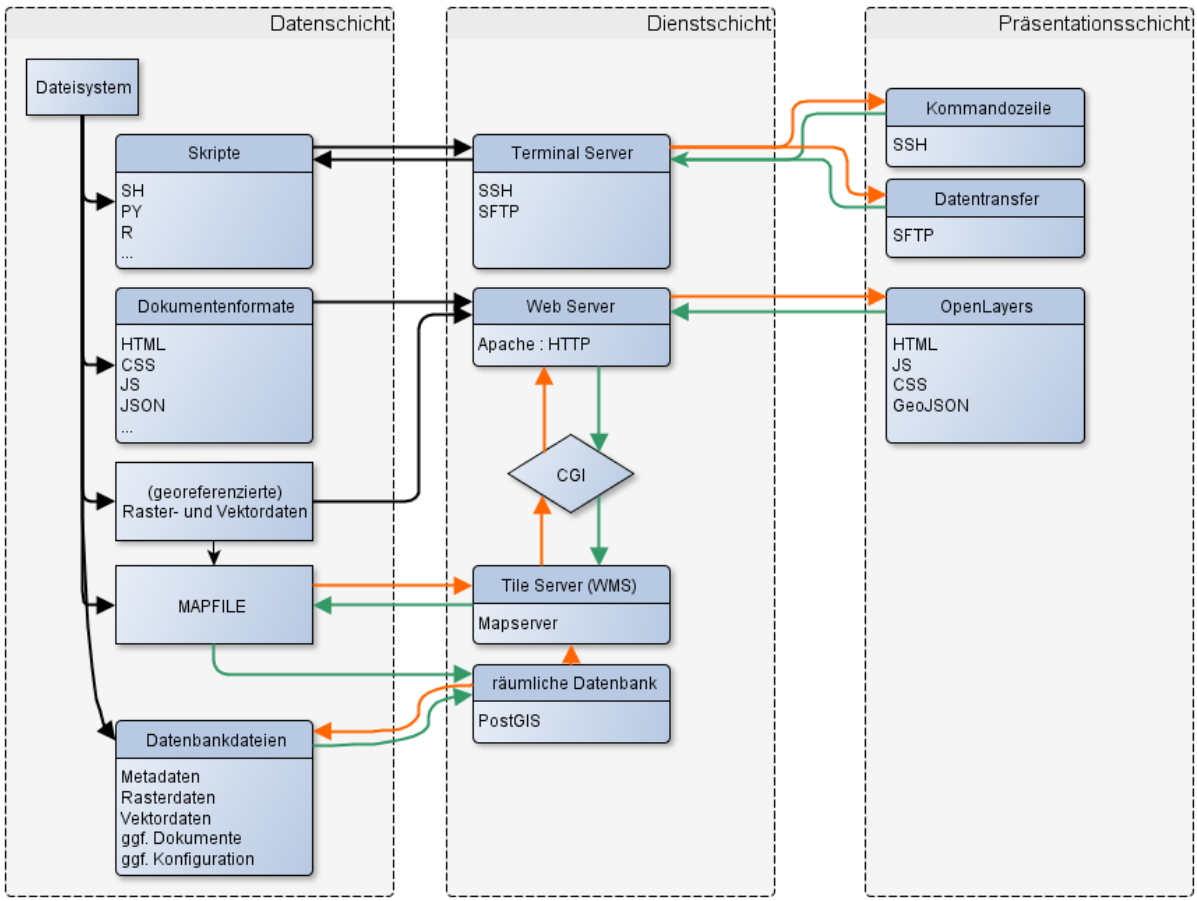

Abb. 3: Abstraktion der Daten-, Dienst- und Präsentationsschicht von *http://gartenkarte.de*. Eigene Darstellung. Nach ALESHEIKH, HELALI, BEHROZ 2002; BUTLER, FAWCETT, MCKENNA 2009; AMIRIAN, ALESHEIKH, BASSIRI 2010; NÁRODNÁ INFRAŠTRUKTÚRA PRIESTOROVÝCH INFORMÁCIÍ.

4.3 Kartengrundlagen

Als Basislayer für den Raum Berlin verwendeten wir die Datensätze des OpenStreetMap Projekts. Diese standen zunächst unter einer *Creative Commons Namensnennung-Weitergabe unter gleichen Bedingungen* Lizenz zur Verfügung, sind seit 12. September 2012 aber unter der *Open Database License* zu erhalten.

Solche Datensätze lassen sich im kleinen Maßstab direkt aus der Karte von OpenStreetMap erstellen, stehen für zusammenhängende Räume in größerem Maßstab aber bereits umgewandelt in gängige Formate auch auf Spiegelservern zur Verfügung. Wir bezogen diese Daten in Form von *Shapefiles* aus der Geofabrik[8].

In einer späteren Mashup-Variante unserer Gartenkarte verwendeten wir Kartenkacheln von Stamen Design (CC BY 3.0)[9], OpenStreetMap / Mapnik (CC BY SA 2.0) und OpenStreetMap / MapQuest (CC BY SA 2.0).

Die Punktdaten für die einzelnen Gärten erhielten wir im Verlauf vom INNSULA Projekt über deren Wissens-, Projekt- und Aktivitätensammlung zu Urbaner Landwirtschaft, eingangs über einen privaten Zugang und nun im Anschluss über die Webseite *Stadtacker*. Während beide Projekte, der Stadtacker und die Gartenkarte, noch in der Entwicklungsphase waren, erhielten wir aus der Datenbank lediglich Projekttitel und -adresse, woraus wir selbst durch manuelle Geokodierung die Koordinaten generierten.
Dabei unterstützten uns einmal mehr verschiedene Webdienste (Google Geocoder, Yahoo! Geocoder), die jedoch kostenfrei nur eingeschränkte Funktionalität boten. Es blieb festzustellen, dass keine FOSS-basierten verfügbaren, anspruchsvollen Geokodierer als Webservice zu finden waren. In einer späteren Version der INNSULA Datenbank lagen die Gärten direkt geocodiert vor. Der Schritt der Geokodierung wurde somit aus unserer Architektur ausgelagert.

Sobald die Daten einmal für uns handhabbar in Tabellenform vorlagen, konnten wir sie, ursprünglich mit Hilfe einer Tabellenkalkulation, später mit R, einlesen und modifizieren. Eine erste Version unseres Arbeitsablaufs fügte die Daten in eine *Tab Seperated Value* (TSV) Textdatei, welche bei Einhaltung einer bestimmten Syntax durch OpenLayers gelesen und zur

[8] *http://www.geofabrik.de* (04.01.2013)
[9] *http://maps.stamen.com* (04.01.2013)

Anzeige der Gärten genutzt werden konnte. Im späteren Verlauf diente uns R zum Einen zur Analyse und außerdem zum Export in das standardisierte GeoJSON Format, welches ebenfalls durch OpenLayers interpretierbar ist.

4.4 Gehversuche

Erinnern wir uns: Es existiert eine Vielzahl an Anwendungen, welche dem OGC OWS Standard konform sind und sich zur Implementierung einer solchen Aufgabe eignen. Auf Grund unseres Neulingsstatus im WebGIS Bereich war es daher anfänglich unumgänglich, verschiedene Möglichkeiten der Implementierung durchzuspielen.

Noch ohne das Feld an GFOSS ausreichend zu verstehen, wurden Mapbender (OSGeo) und GeoExt (OpenGeo) als erste zu überprüfende Kandidaten ausgewählt. Learning-by-Doing das es war, zeichnete sich schnell ab, dass beide Pakete zu umfassend für unseren definierten Anwendungsfall waren:

Was Mapbender als Verwaltungswerkzeug für ganze Geodateninfrastrukturen auszeichnet und ihn für die Verwendung als Bibliothek von *Geoportal*[10] der GDI-DE, betrieben vom Bundesamt für Kartographie und Geodäsie, prädestiniert, ist gleichzeitig auch eine Überfülle an Möglichkeiten für den kleinen Gebrauch. Es gelang uns trotz einer Vielzahl investierter Stunden nicht einmal eine Karte auch nur zu sehen.

Teil der ersten Experimente war auch die Software GeoExt, welche eine auf der JavaScript Kartenbibliothek OpenLayers (s.u.) aufbauende, umfassendere Darstellungsbibliothek ist. Leider war es uns auf Grund mangelnder Kenntnisse ebenfalls nicht möglich, damit aussagekräftige Ergebnisse zu produzieren, sodass die Versuche als gescheitert angesehen mussten.

Ein erneuter Schritt in Richtung Konzeptionalisierung und die fortwährende Auseinandersetzung mit dem breiten GFOSS Feld spezifizierte dann auch die Ansprüche an unsere Client-Server-Architektur. Auf Clientseite würde die vielfältig verwendete und gut dokumentierte Bibliothek OpenLayers zum Einsatz kommen, während serverseitig unter Verwendung des *Common Gateway Interface* (CGI) eine Kombination aus Web- und Mapserver für die Auslieferung der Anwendung und der Kartenkacheln diente.

[10] *http://www.geoportal.de* (04.01.2013)

4.5 UMN Mapserver

Der "MapServer ist eine Open Source-Software um raumbezogene Daten und interaktive Kartenanwendungen im Web zu präsentieren."[11] Er steht unter einer MIT-ähnlichen Lizenz[12] und ist für viele Betriebssysteme verfügbar (ebd.).

Wie bei der ersten Installation von komplexen Anwendungen üblich, förderten die Voraussetzungen des UMN Mapservers erneut das Verständnis der Funktionsweise moderner *nix-Systeme. Nach weiteren gescheiterten Versuchen den Mapserver über die Betriebssystemschnittstelle zu installieren, musste der Weg über die Übersetzung der Quelltexte in Maschinensprache gewählt werden. Die regelmäßige Konsultation einer Suchmaschine und die ausgezeichnete Dokumentation zusammen führten schließlich zum Erfolg, sodass der Mapserver bei Direktaufruf in der Lage war, eine Fehlermeldung zu produzieren:

"No query information to decode. QUERY_STRING is set, but empty."[13]

Da beim Direktaufruf der Anwendung keine Parameter übergeben wurden, konnte keine Abfrage auf die geographische Datenbasis ausgeführt werden. Dieses Verhalten ist bei der Installation erwünscht und stellt das erste Lebenszeichen nach der Übersetzung in Maschinensprache dar.

Was nun folgte war die Aufbereitung unserer geographischen Daten für die Präsentation im Mapserver mit Hilfe von QGIS. Denn der Mapserver ist zwar in der Lage mit einer Vielzahl an geographischen Vektor- und Rasterdaten umzugehen, benötigt jedoch eine Datei, welche das geographische Koordinatensystem und die Projektion für einzelne Ebenen angibt. Diese MAPFILE genannte Datei konnte bequem über einen QGIS Export und anschließende manuelle Nachbearbeitung eingesetzt werden.

Nach einer weiteren Recherche der WMS Abfragespezifikation konnte, nach den langwierigen Prozessen der Installation und Konfiguration des Mapservers, die erste in einer Kartendarstellung (siehe Abb. 3) mündende Abfrage ausgeführt werden:

[11] *http://mapserver.org/de* (07.01.2013)
[12] *http://mapserver.org/de/copyright.html* (04.01.2013)
[13] *http://geosemantik.de/cgi-bin/mapserv* (03.01.2013)

http://geosemantik.de/cgi-bin/mapserv?map=/var/www/vhosts/geosemantik.de/cgi-bin/shapefile/berlin/berlin.map&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&LAYERS=berlin_osm&STYLES=&SRS=EPSG:4269&BBOX=13.0628,52.3279,13.764,52.68&WIDTH=1024&HEIGHT=768&FORMAT=image/png

Der erste Teil der Abfrage ist von der erfolgreichen Fehlermeldung her bekannt. Hinter dem Fragezeichen folgen nun Variablen, welche der Mapserver zur ordnungsgemäßen Beantwortung benötigt:

- **map** : Lage des MAPFILEs im Dateisystem des Servers
- **service** : angefragter Dienst. Mapserver ist überdies dazu in der Lage neben WMS auch weitere OGC OWS anzubieten.
- **version** : gewählte Spezifikation des Dienstes
- **request** : Anfrageart
- **layers** : gewählte Ebenen aus dem MAPFILE
- **styles** : Gestaltungselemente
- **srs** : Koordinatenreferenzsysteme, Referenzellipsoide oder Projektionen in EPSG (European Petroleum Survey Group Geodesy : einer Quelle dafür) Notation
- **bbox** : Bounding Box, d.h. der darzustellende Kartenausschnitt, dargestellt über die Koordinaten der Punkte oben links und unten rechts
- **width** : Ausgabebreite (in Pixeln)
- **height** : Ausgabehöhe (in Pixeln)
- **format** : gewünschtes Dateiformat

Solcherlei Abfragen würden auf Grund ihrer Komplexität in Zukunft automatisiert durch die Darstellungsbibliothek in der Präsentationsschicht erfolgen.

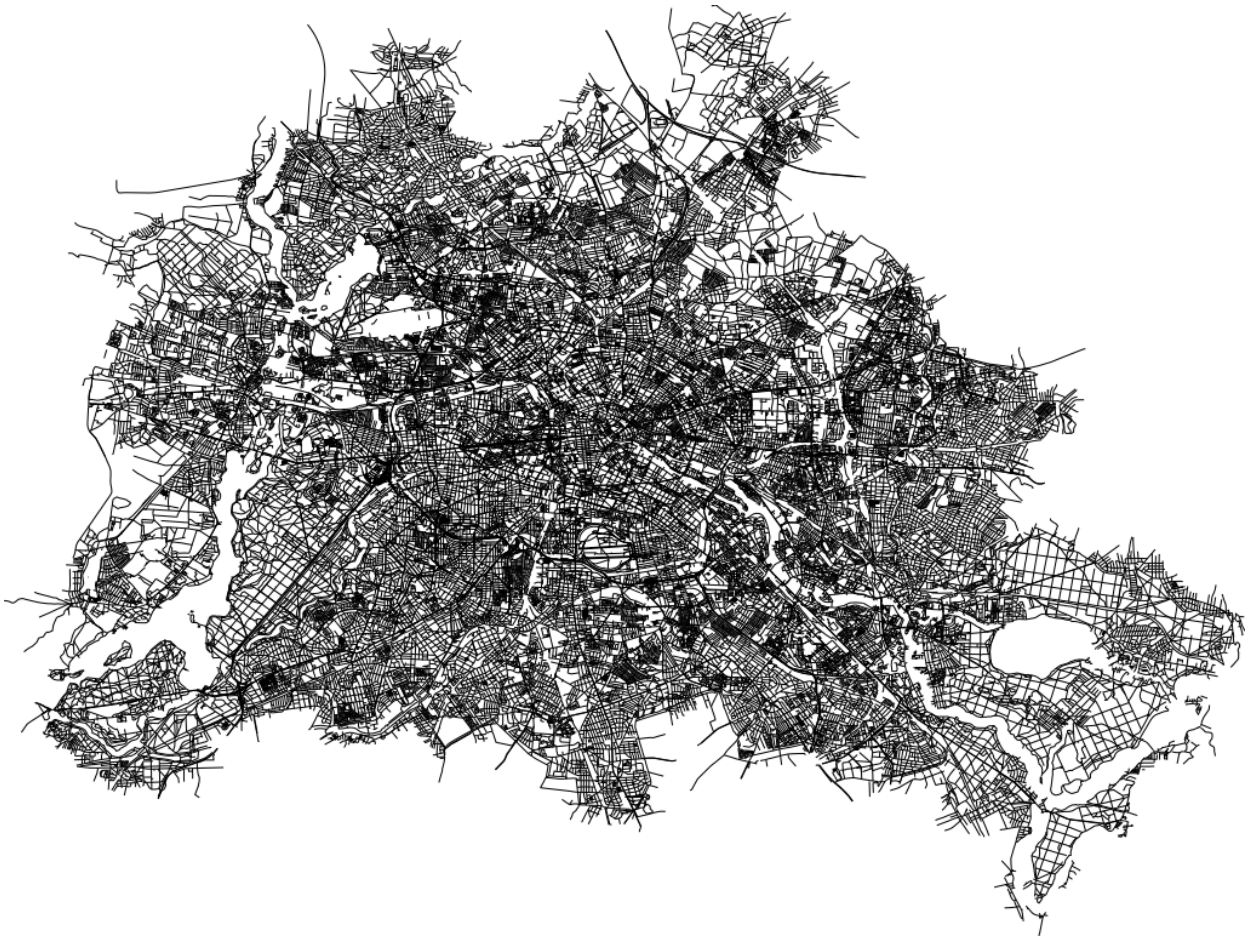

Abb. 4: Ergebnis der ersten erfolgreichen Abfrage des Mapservers. Eigene Darstellung.

4.6 Mashup Verwendung

Auf Grund der interoperablen Architektur des OWS Standards stellt unser WMS / Mapserver am Ende auch nur eine der vielfältigen Quellen für Kartenmashups dar. Dies erlaubte uns die ursprünglich als rein studentisches Forschungsprojekt geplante WebGIS-Anwendung hochzuskalieren und an einem bestimmten Punkt in der Kooperation mit INNSULA den eingesetzten WMS Dienst zu wechseln.

Im Rahmen der Implementierung eines Kartenprototypen für *Stadtacker* wählten wir daraufhin die auf OSM Daten beruhenden Kartenkacheln des Anbieters Stamen. Obwohl uns durchaus bewusst war, dass wir uns damit in die Abhängigkeit von einem bestimmten Anbieter begaben, was in den Anfängen auch häufiger zu Darstellungsproblemen führte, entschieden wir uns auf Grund der hohen ästhetischen Qualität dafür.

Als Alternativen blieben weiterhin die Mapnik- und MapQuest-Ebenen als Kartengrundlage auswählbar. Ein infrastrukturelles Ziel könnte es an diesem Punkt sein, vermehrt in eigene Hostingarchitekturen zu investieren, um damit Unabhängigkeit beizubehalten. Somit ist unsere Karte am Ende eine Kombination von Daten zweier unterschiedlicher Quellen. Stünde von Seiten der INNSULA Forschung eine offene, interaktive Schnittstelle zum Austausch der Geodaten zur Verfügung, könnten wir im Mashup auf die manuellen Datenaufbereitungsschritte (hauptsächlich Geokodierung) verzichten.

4.7 OpenLayers

Für die interaktive Darstellung der Karte entschieden wir uns für die *Open Source* JavaScript Bibliothek OpenLayers. Sie dient der Veröffentlichung dynamischer Karten in jeder Art von Webseite und versteht sich auf alle Arten von geographischen Informationen. Die Bibliothek steht unter einer FreeBSD Lizenz[14]. Der eigentliche Code der *Berliner Gartenkarte* umfasst lediglich vier Textdateien:

- ./index.html : Grundgerüst der Webseite und Referenz von Stil- und Skriptdefinitionen
- *./gartenkarte.js : OpenLayers Initialisierungsskript*
- ./gartenkarte.css : Stildefiniton
- ./gartenkarte.json : GeoJSON Datengrundlage

Hinzu kommen OpenLayers' JavaScript und Stildefinition und Skript von Stamen und OpenStreetMap, welche deren Karten OpenLayers bekannt machen sowie einige wenige Bilddateien, welche für die Benutzerschnittstelle verwendet werden. Von entscheidender Bedeutung für OpenLayers ist das Initialisierungsskript *gartenkarte.js*. Dieses ist im Codeverzeichnis aufgeführt und wird im Folgenden kurz erklärt.

Das Skript wird im Browser eines Besuchers der Seite *http://gartenkarte.de* aufgerufen und steuert die grundlegende Logik dieser WebGIS Implementierung. Es kontrolliert die Popups, welche beim Überfahren der Gartensymbole angezeigt werden (*Zeile 21ff.*), initialisiert die Karteneigenschaften wie Projektionen und Benutzerschnittstellenelemente (*Zeile 60ff.*) und initialisiert die Kartenebenen (*Zeile 91ff.* - dieser Teil ähnelt in seiner Funktion sehr stark dem MAPFILE). Anschließend zeichnet es die Karte und zoomt sie auf den von uns festgelegten Kartenausschnitt (*Zeile 177ff.*, siehe Abbildung 4). Der Anwender erhält eine kontextbezogene Darstellung von Urbanen Gärten und verwandten Projekten in Berlin.

[14] *http://openlayers.org* (07.01.2013)

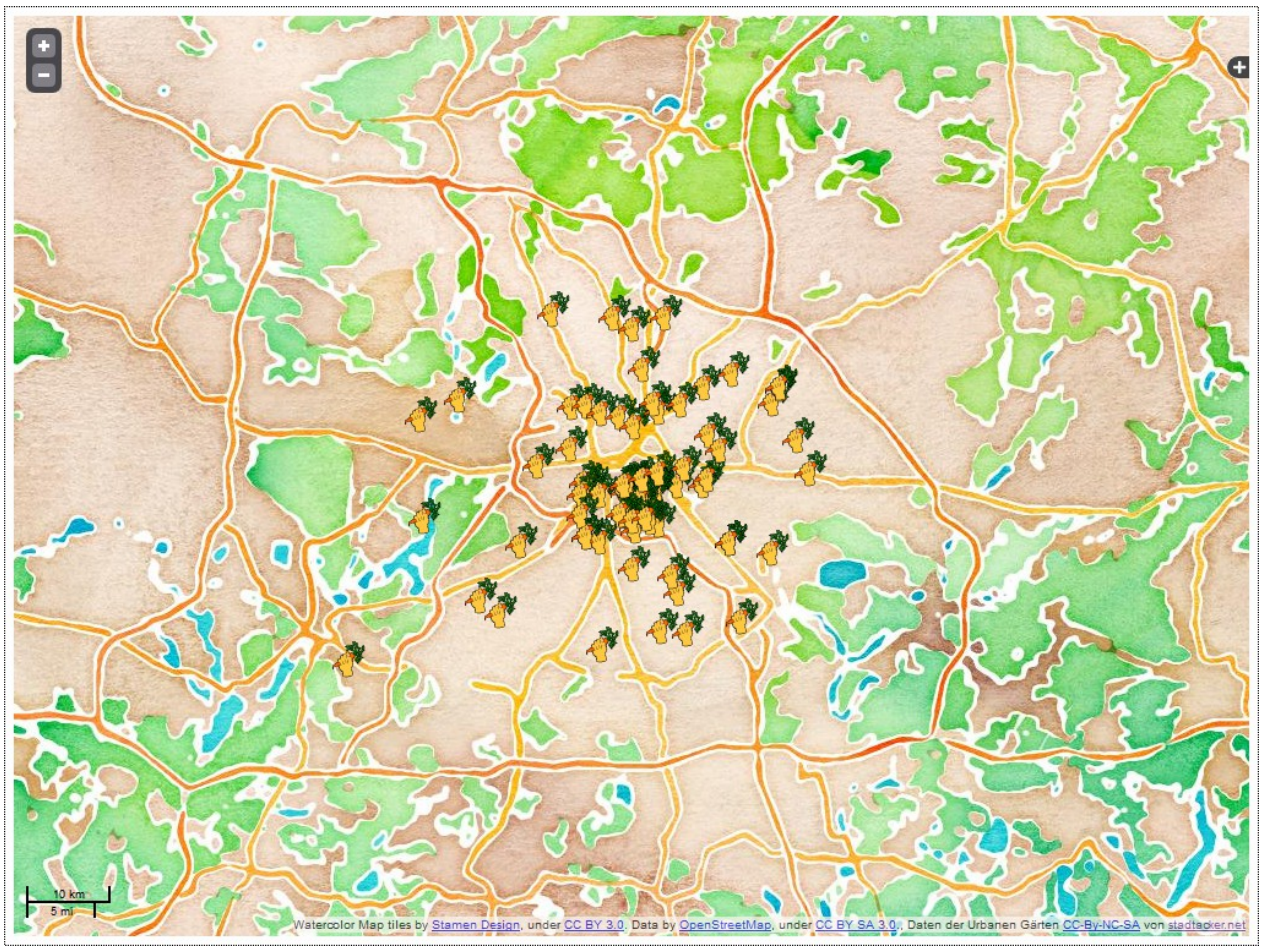

Abb. 5: Ergebnis : Berliner Gartenkarte : *http://gartenkarte.de* (02.01.2013)

5. Datenaufbereitung und Anwendungsbereiche im Zusammenhang mit Gemeinschaftsgärten in Berlin

Aus der Zusammenarbeit mit den Gartenaktivisten des Allmendekontors[15] ergaben sich weitere Anwendungsfelder über die entworfene Webseite *gartenkarte.de* hinaus. So wird einerseits auf der vorhandenen Datenbasis noch eine großformatige Karte mit Informationen über die verschiedenen Gärten entworfen, welche den Gärten als Informationstafel für Besucherinnen zur Verfügung gestellt wird. Da sich das Konzept von *gartenkarte.de* aus dargelegten Gründen letztendlich nicht entsprechend unseres eingehenden Anspruches realisieren ließ, sehen wir hierin einen Kompromiss, um der Berliner *Gardening Community* die Zusammenstellung der Informationen auf diesem Wege zur Verfügung zu stellen. Andererseits ist im Rahmen des erwähnten Projektes an der HU Berlin ein Informationsband in Bearbeitung, welcher Anfang 2014 nach Beendigung des Projektes veröffentlicht werden soll. Auf Anfrage werden hierfür auf Grundlage der Gartenkarte zwei Karten entworfen. Auch bei der Erstellung dieser Karten bildet der FOSS - Gedanke die Grundlage, d.h. es werden ausschließlich offene Programme zur Erstellung verwendet sowie das Ergebnis unter einer CC-Lizenz veröffentlicht.

In diesem Kapitel soll es darum gehen einerseits die Datenaufbereitung der zur Verfügung gestellten Geodaten der Gärten für alle Schritte, d.h. die *gartenkarte.de*, die Druckkarten sowie weitere Analysen, zu automatisieren. Weiterhin wird auf die Erstellung der Druckkarten sowie Möglichkeiten einer weiteren Analyse der eingegangen.

5.1 Datenaufbereitung

Die Datengrundlagen für all unsere Weiterverarbeitungen bildet die Datenbank von *stadtacker.net*. Leider konnte auf Grund des verwendeten SharePoint CMS und beschränkter Zugriffsrechte unsererseits bisher keine Automatisierung des Datenzugriffs realisiert werden. Jedoch können wir zur Weiterverarbeitung manuell auf die Daten zugreifen. Für die Aufbereitung dieses Datensatzes wurde ein R-Skript (Skript siehe Codeverzeichnis) geschrieben um:

[15] *http://www.workstation-berlin.org/stadt-und-garten5/161-allmende-kontor* (07.01.2013)

- aktuelle Geodaten der Gärten in Berlin *gartenkarte.de* in passendem Format zuzuführen.
- einen passenden Datensatz für die Erstellung der Druckkarten zu haben.
- eine exemplarische Forschungsfrage zu bearbeiten.

5.1.1 R zur Bearbeitung von Geodaten

R „is a free software environment for statistical computing and graphics" (BIVAND 2008 : 2). Mit einer *GNU General Public License* (GPL) Lizenz ausgestattet, reiht sich R also wunderbar in den Anspruch dieser Arbeit ein[16]. Durch die Einbindung zahlreicher Pakete (z.B. SP, RGDAL, MAPTOOLS) eignet sich R über statistische Anwendungen hinaus sehr gut zur Aufbereitung von Geodaten, deren Analyse und Visualisierung. R ist ein kommandozeilenbasiertes Programm und eine Skriptsprache, wobei zahlreiche spezifische Editoren sowie IDEs (Integrated Development Environments) vorhanden und weiter im Entstehen sind. Als eine solche Umgebung stellt sich insbesondere Rstudio - ebenfalls FOSS - nicht zuletzt wegen der Ähnlichkeit zu anderen Umgebung wie MatLab als einsteiger- und benutzerfreundlich dar (ebd.; TORFS 2012). R ist eine simple und effektive Programmiersprache und eignet sich insbesondere aufgrund der schnellen Weiterentwicklung sowie enormer Möglichkeiten der Erweiterung für viele wissenschaftliche Anwendungen. Wissenschaftlerinnnen und andere Anwenderinnen haben bereits tausende spezialisierte Methoden implementiert, welche als Pakete direkt in R integriert werden können (BIVAND 2008 : 2ff.).

5.1.2 Datenbereinigung und Bereitstellung für gartenkarte.de

Ein erster Schritt besteht darin, die Daten nach den Berliner Adressen zu filtern, zu sortieren und als neuen Datensatz bzw. *subset* zur weiteren Verarbeitung zu speichern. Für weitreichendere statistische Kontexte steht das Paket *subselect* bereit (CADIMA 2012), in diesem Fall genügt allerdings die einfache *subset* - Funktion (*gartenkarte.r*, *Zeilen 11—27*).
Für *gartenkarte.de* wird das an JSON (JavaScript Object Notation) angelehnte GeoJSON verwendet. Dank der Einbindung des GDAL - Paketes ist ein Export mit R ohne weiteres möglich. Ist der entsprechende Treiber für das Format definiert, kann exportiert werden (*gartenkarte.r*, *Zeilen 57—59*). Die hierfür verwendete *OGR Simple Feature Library* stellt einen Zugang zu vielen Vektorformaten bereit und ist in *gdal* enthalten (BIVAND 2008 : 94ff.).

[16] *http://www.r-project.org* (03.01.2013)

5.2 Druckkarten

Die Datenaufbereitung wird weitergeführt um für alle Karten zum jeweiligen Veröffentlichungstermin die aktuellen Daten zu verwenden und auf Grundlage des immer gleichen Datenformats die vorgefertigte Karte neu erstellen zu können (bzw. die gärtenbezogenen Geodaten und Attribute zu ersetzen).

5.2.1 Wissenslandschaften

Für das erwähnte Projekt zum Thema Wissenslandschaften im Kontext urbaner Landwirtschaft an der HU Berlin werden zwei Karten benötigt: eine Überblickskarte aller Gärten in Berlin und eine Wissenslandkarte ausgewählter Gärten. Letztere ist inhaltlich noch nicht weiter besprochen und kann hier daher nicht weiter beschrieben werden. Als Grundlage dienen die aus der INNSULA-Datenbank generierten Daten sowie das Stamen-Design der Gartenkarte. Der gegenüber uns formulierte Anspruch an die Karte bezieht eine durchnummerierte Darstellung der Gärten auf Grundlage des hier verwendeten Kartendesigns sowie eine alphabetische Ausgabe der Gartennamen in der Legende mit ein. Die alphabetische Sortierung wurde bereits oben durchgeführt, eine Spalte zur Durchnummerierung der Gärten wird erzeugt (*gartenkarte.r*, *Zeile 30*).

Um die Karte mit QGIS zu erzeugen, bietet es sich an ein komplettes *Shapefile* zu exportieren. Zu diesem Zweck werden die Daten in die entsprechende *Spatial Class* umgewandelt. R kann nicht nur genutzt werden um Nummern, Vektoren und Matrizen zu berechnen, sondern kann um zahlreiche weitere Klassen erweitert werden, um andere Datentypen zu analysieren. Klassen werden von den jeweiligen Paketen bereit gestellt. So sind alle *Spatial Classes* in dem Paket *sp* (classes and methods for spatial data)[17] enthalten. Der hier verwendete *SpatialPointsDataFrame* ist eine Unterklasse der Klasse *SpatialPoints*. Die Klasse *Spatial* vereint alle raumbezogenen Daten im Paket *sp*. Ein *DataFrame* wird deshalb verwendet, da darin Daten jeglicher Art (numerisch, logisch, Text etc.) gespeichert und kombiniert werden können (BIVAND 2008 : 3; ZEILEIS 2006) (*gartenkarte.r*, *Zeilen 36—40*).
Anschließend werden die Daten als *Shapefile* exportiert. Desweiteren wird eine zweispaltige Textdatei mit Gartennummer und Gartenname erzeugt, welche später zum Erzeugen der Legende dient (*gartenkarte.r*, *Zeile 54f.*, *Zeile 62f.*).

[17] *http://cran.r-project.org/web/packages/sp/sp.pdf* (06.01.2013)

Abbildung 6 zeigt eine Vorabversion der Überblickskarte, wobei das endgültige Layout noch redaktionell abgestimmt werden muss. Der weitere Arbeitsablauf zum Erstellen der Karte in QGIS soll hier nicht weiter erläutert werden. Wichtig ist lediglich, dass die raumspezifischen Daten inklusive Legende dank der automatisierten Datenaufbereitung jederzeit durch Ausführen der R-Skripte aktualisiert werden können.

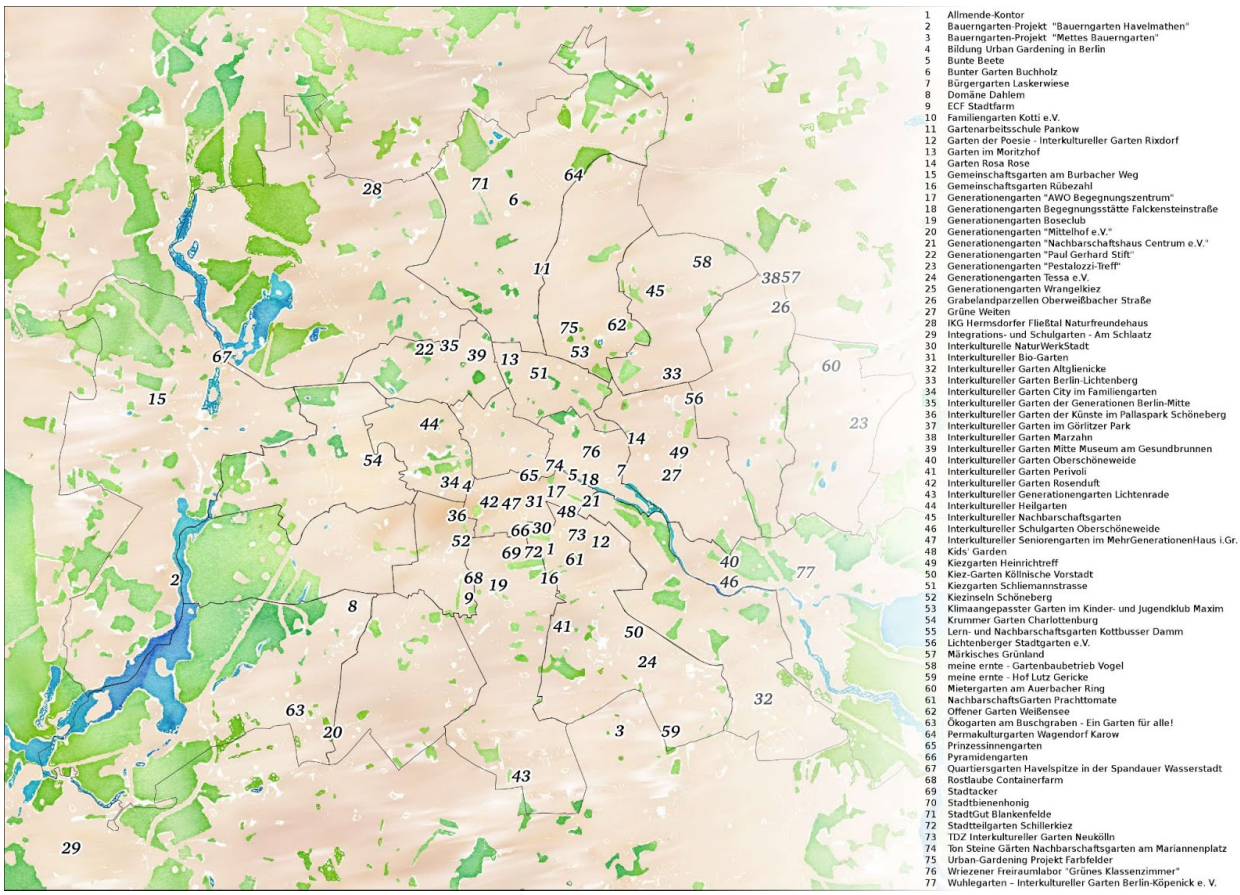

Abb. 6: Eine erste Version der Gartenkarte als Druckversion. Eigene Abbildung.

5.2.2 Karte für die Gärten

Die Karte für die GärtnerInnen welche einigen Gärten zum Aushängen zur Verfügung gestellt werden soll, wird ähnlich aufgebaut sein wie diejenige für die Wissenslandschaften. Zusätzlich soll diese allerdings noch Informationen über den Standort (Adresse) sowie die Bezirkslage enthalten. Um die geplante großformatige Karte entwerfen zu können, bedarf es hochauflösende Rasterdaten der verwendeten Stamen Kartenkacheln. Dazu wurde einerseits ein Bash-Skript geschrieben, das die Kacheln im Bereich Berlins vom Stamen WMS durchnummeriert herunterlädt. Anschließend können diese mit Hilfe eines Python-Skriptes zusammengefügt werden (siehe *tile_scraper.sh* und *tile_merger.py* im Codeverzeichnis).

Da jeder Garten in der Legende zur Orientierung mit dem jeweiligen Bezirk gekennzeichnet werden soll, bedarf es zunächst einer räumlichen Zuordnung. Die räumlichen Daten der Gärten sind bereits vorhanden, Daten zu den Bezirken werden geladen in R (*gartenkarte.r*, *Zeile 45*).

Die Polygone der Bezirke entstammen einem OSM-Datensatz und wurden lediglich mit QGIS bereinigt und aufbereitet. Wenn herausgefunden werden soll, ob die Punktdaten (Gärten) in Polygonen (Bezirke) liegen und darüber hinaus eine Zuordnung der Punkte zu den jeweiligen Polygonen erreicht werden soll, geschieht dies Mithilfe eines *map overlay*. Ein numerischer *map overlay* "combines spatial features from one map layer with the attribute (numerical) properties of another" (PEBASMA 2012 : 1). Für eine solche Operation eignet sich in R die Methode *over* (sp-Paket).

Beide Datensätze haben die gleichen Projektionen und die jeweiligen Bezirke werden dem Datensatz in einer extra Spalte zugeordnet (*gartenkarte.r*, *Zeilen 48—50*). Um die Genauigkeit des Datensatzes zu erhöhen, könnte dieses Verfahren auch bereits bei der Datenaufbereitung und Auswahl der Berliner Gärten angewandt werden.

5.3 Forschungsfragen mit R

Neben der erstellten Gartenkarte Webseite sowie den Druckkarten solle im Folgenden, angelehnt an die hier behandelte Thematik, eine kleine Forschungsfrage entworfen werden, um die Möglichkeit einer wissenschaftlichen Bearbeitung des Themas mit Hilfe der GIS-fähigen FOSS-Software R zu illustrieren.

Der Mangel an naturnahen Grünräumen zählt allgemein zu den größten Defiziten in benachteiligten Stadtbezirken. So lässt sich feststellen, dass diejenigen Gebiete mit einer hohen „sozialen Problemdichte" im Mittel die wenigsten Anteile von Grün- und Freiflächen aufweisen (Bunge 2011:14). Nachbarschafts-, und Gemeinschaftsgärten können ein hoher Synthesgrad von sozialer Leistung, partizipativer Stadtentwicklung sowie nachhaltiger Schaffung von Grünflächen zugesprochen werden. Gerade in Berlin sind es diese recht zahlreich, auch aufgrund eines Senatsbeschlusses welcher zwei Gemeinschaftsgärten pro Stadtbezirk festlegt (ebd: 15). Um zu prüfen welcher Bezirk z.Zt. wie viele Gemeinschaftsgärten beheimatet, kann die These geprüft werden in R mit den Befehlen:

```
y <- gaerten_bezirk$gaerten_nach_bez
y > 2
```

oder es wird mithilfe eines *subset* eine Abfrage generiert

```
Bez_kl_2 <- subset(gaerten_bezirk, gaerten_nach_bez < 2)
```

und folgendes Ergebnis ausgegeben:

```
Charlottenburg              1
Reinickendorf               1
Treptow                     1
Mitte                       0
Steglitz                    0
Wilmersdorf                 0
```

Eine komplette “Versorgung” Berlins im Sinne des Senatsbeschlusses ist z.Zt. also noch nicht gewährleistet. Um die weitere Entwicklung beobachten zu können, soll nun eine Textdatei ausgegeben werden, welche mit der Bearbeitung jeden neuen Datensatzes inklusive Erstellungsdatum erweitert wird:

```
gaerten_nach_bez <- sort(table(gaerten_berlin$Bezirke), decreasing
    =TRUE)
gaerten_bezirk <- as.data.frame(gaerten_nach_bez)
x <- date()
write.table(gaerten_bezirk,file="Gaerten_nach_Bezirken.txt",sep="/t",ap
    pend=TRUE, col.names = x, quote = FALSE)
```

Ausgabe:

```
Sat Jan  5 15:13:30 2013
Kreuzberg               14
Neukoelln               12
Pankow                  4
Schöneberg              4
Friedrichshian          3
...
```

Das Ganze soll nun zur Veranschaulichung zusätzlich visualisiert werden. Nachdem die oben berechneten Häufigkeiten dem *SpatialPolygonDataFrame* zugeordnet werden

```
y <- gaerten_in_bezirk[order(gaerten_in_bezirk$name),]
Bezirke_sort <- Bezirke[order(Bezirke$name),]
Bezirke_sort$x <- y$gaerten_nach_bez
```

können diese im Plot farblich dargestellt werden. Mit *spplot* werden in *sp* grundlegende Methoden bereit gestellt um räumliche Daten und deren Attribute zu visualisieren. Es existieren weitreichender Methoden, mit denen Karten weitaus höheren Anspruchs generiert werden können wie z.B. das Plottingsystem *ggplot2*. Da *spplot* allerdings bereits eine Erweiterung "klassischer" Plotmethoden in R darstellt und speziell für räumliche Informationen entworfen ist, soll dieses Verfahren hier genügen (vgl. Bivand 2008:68*ff*):

```
plotvar <- Bezirke_sort@data$x
nclr <- 7
plotclr <- brewer.pal(nclr,"Greens")
class <- classIntervals(plotvar, nclr, style="equal")
colcode <- findColours(class, plotclr)
spplot(Bezirke_sort, "x", col.regions=plotclr, at=round(class$brks,
  digits=2), main =  "Häufigkeit von Gärten nach Bezirken", sub=date,
  lwd=.8, col="white")
```

Es sind zwei weitere Pakete nötig um eine Klassifizierung vornehmen zu können (*classInt;* vgl. ebd:77) sowie eines um die Daten (Häufigkeiten der Gärten) in Farben umzusetzen (*RColorBrewer;* vgl. ebd:76). Anschließend wird die darzustellende Variable, die Anzahl der Klassen, die Farbgebung sowie die Klassenbreite definiert. Ein Ausgabe der Karte mit Datum im Folgenden.

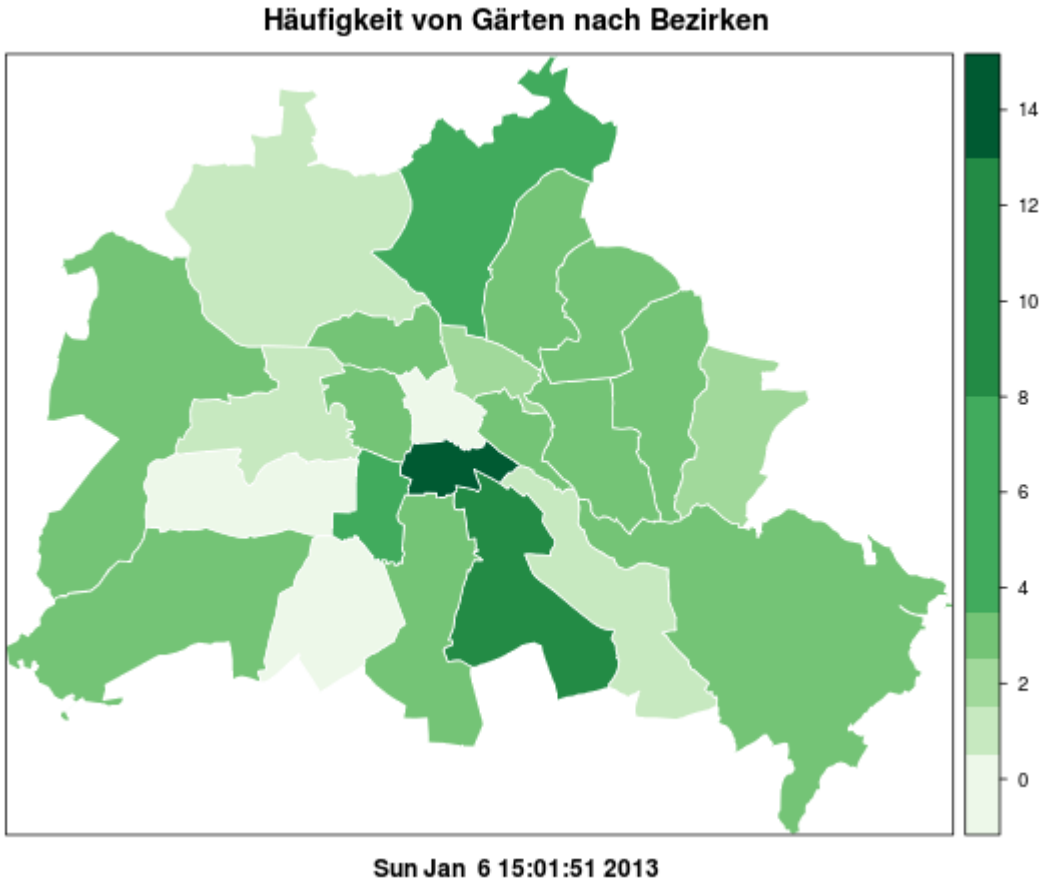

Abb. 7: Gemeinschaftsgärten in Berlin nach Bezirken. Eigene Abbildung.

Es ist also gelungen, die vorhandenen Daten für die weitere Verarbeitung aufzubereiten sowie eine Übersicht über die Häufigkeit der Berliner Gemeinschaftsgärten zu erstellen. In Zukunft könnte darüber hinaus noch eine Zeitreihenanalyse entworfen werden um die räumliche Entwicklung über einen längeren Zeitraum zu dokumentieren bzw. diese postum zu analysieren.

6. Fazit und Ausblick

Es konnten in dieser Arbeit Möglichkeiten aufgezeigt werden, wie das Thema des *Urban Gardening* mit FOSS - Lösungen zu bearbeiten und dazu ein WebGIS Mashup in einer Client-Server Architektur zu implementieren[18] ist. Daneben konnte dies noch als Grundlage für einige Druckkarten dienen, die mit Hilfe von FOSS Werkzeugen immer auf den neusten Stand gebracht werden können.

[18] *http://gartenkarte.de*

So konnte sogar auf verschiedenen Ebenen der *Urban Gardening* - Gemeinschaft ein Mehrwert zugeführt werden. Die Steigerung von Nutzwert und Nachhaltigkeit bezüglich des WebGIS gegenüber der vorher gegebenen Plattform *Urbanacker* kann auf Grund der gegebenen Änderungen[19] mit dieser Arbeit noch nicht abschließend beurteilt werden. Aber auch die Kooperation mit INNSULA hat sich als erfolgreich herausgestellt, auch wenn der *Stadtacker* dank der schlechten Kompatibilität der Datenbanksysteme jetzt ein Google Maps Mashup verwendet.

Es ist aber festzuhalten mit den Lösungen für die Gartenkarte einen sehr guten Grundstock gelegt zu haben, um die an uns selbst gestellten Ansprüche als Übung im Umgang mit GFOSS Werkzeugen zu realisieren. Dies bedeutet auch einen FOSS-Grundstein zu haben, wenn in zwei Jahren das INNSULA Forschungsprojekt ausläuft und die Wissenssammlung an die Zivilgesellschaft übergeben wird. Gegenwärtig existiert nun aber eine unabhängige FOSS-basierte Plattform für Berlin, auf der sich Interessierte ein Bild über die Lage der Gärten machen können.

Natürlich können weitere Verbesserungen in Benutzbarkeit und Layout gemacht werden. So soll mit einer abschließenden Offenlegung der Daten bzw. einer Implementierung einer Schnittstelle seitens des Stadtackers eine Automatisierung realisiert werden, um die Gartendaten auf der Gartenkarte auf dem aktuellen Stand zu halten. Weiterhin soll eine Verlinkung zu den vom Stadtacker bereit gestellten Steckbriefen stattfinden, um weiterführende Informationen zu den einzelnen Gärten anzubieten. Des weiteren stellt sich an der Darstellung z.Zt. die nicht vorhandene Clusterung der Icons auf bestimmten Zoomstufen als suboptimal heraus. Insbesondere bei der Überlegung einer Einbindung des deutschlandweiten Datensatzes könnte dies zu unangenehmen Effekten führen.

Bei Betrachtung unserer eigenen Arbeitsweise können wir feststellen, dass es eine herausfordernde und spannende Zeit war, sich dem Feld mit Online Kollaborationswerkzeugen zu nähern und sich davon unsere eigene Forschungspraxis formen zu lassen. Wir erkennen einen möglichen weitreichenden Beitrag der Geoinformatik an Diskursen der zivilgesellschaftlichen Stadtentwicklung hin zur *Open Society*.

Die Technologien der *Neogeography*, eingedacht in die Entwicklungen hin zum semantischen Web, zum Internet der Dinge und weiterer Verbreitung von mobilen Endgeräten, lassen viele

[19] Neuauflage der Urbanacker-Seite als *http://stadtacker.net*

Ideen hervortreten, u.a. diese : eine kollektive, kritische, geosemantische Grassroots Kartierung von Alltagsphänomenen in den Händen von vielen.

Es hat auch nicht die letzte Stunde geschlagen für DesktopGIS Anwendungen, denn diese werden noch immer für Aufbereitung der Daten zur Einpflege in (WebGIS) Datenbanken benötigt. Es konnte hiermit aber gezeigt werden, dass FOSS-Werkzeuge in ihrer Diversität sehr leistungsfähig sind und noch ein großes Potential und gesellschaftliche Bedeutung in sich tragen könnten.

Literaturverzeichnis

- ALESHEIKH, AA.; HELALI, H.; BEHROZ, HA. (2002): Web GIS: Technologies and Its Applications. Figure 1b. Teheran. In: ARMENAKIS, C.; LEE, Y.C. (Ed.). International Society for Photogrammetry and Remote Sensing. Commission IV. Symposium 2002. Geospatial Theory, Processing and Applications. Proceedings. Volume XXXIV Part 4.
- ALLMENDEKONTOR, ORANGOTANGO (2012): Allmende-Kontor. Gemeinschaftsgarten. Karte. In: ORANGOTANGO. Kollektives Kritisches Kartieren. Auflage 2. Berlin.
- ALIPRANDI (2011): Interoperability And Open Standards: The Key To True Openness And Innovation. In: International Free and Open Source Software Law Review. Vol. 3, Isue 1. S. 5-25. http://www.ifosslr.org (01.01.2013).
- AMIRIAN, P; ALESHEIKH, AA.; BASSIRI, A. (2010): Standards-based, interoperable services for accessing urban services data for the city of Tehran. In: Computers, Environment and Urban Systems. Volume 34. Issue 4. July 2010. Teheran. S. 309-321. *http://www.sciencedirect.com/science/article/pii/S0198971510000049* (03.12.2012)
- BAIER, A. (2010): Urbane Subsistenz als Teil nachhaltiger Gesundheitsförderung. IN: GÖPEL, E. (Hrsg.): Nachhaltige Gesundheitsförderung. S. 240-257. Frankfurt a.M.
- BATTY et al. (2010): Map mashups, Web 2.0 and the GIS revolution. In: Annals of GIS. Vol. 16, No. 1, März 2010. S. 1–13.
- BERNARD et al. (2005): Geodateninfrastruktur – Grundlagen und Anwendung.
- BIVAND, R.; PEBESMA, E.; GÓMES-RUBIO, V. (2008): Applied Spatial Data Analysis with R. New York.
- BUNGE, C.; HORNEBERG, C.; PAULI, A. (2011): Auf dem Weg zu mehr Umweltgerechtigkeit – Handlungsfelder für Forschung, Politik und Praxis. In: UMID – Umwelt und Mensch – Informationsdienst: Themenheft Umweltgerechtigkeit. Ausgabe 2, 2011. S. 9 – 17.
- BUTLER, H.; FAWCETT, D.; MCKENNA, J. (2009): Einführung zum MapServer. *http://mapserver.org/de/introduction.html* (03.01.2013)
- CADIMA, J. et al (2012): The subselect R Package. Erreichbar unter *http://cran.r-project.org/web/packages/subselect/vignettes/subselect.pdf* (02.01.2013)
- DE LANGE, de N. (2006): Geoinformatik in Theorie und Praxis. Heidelberg.
- DEMAREST, A., MINOD, R., RICHTER, J. (2012): Public Space Invaders. a collaborative research on collective urbanism. Berlin. Paris. *http://publicspaceinvaders.org/2012/igc-article/* (07.01.2013).

- EDZER, P. (2012): Map overlay an spatial aggregation in sp. Verfügbar unter: *http://cran.r-project.org/web/packages/sp/vignettes/over.pdf* (03.01.2013)
- FREE SOFTWARE FOUNDATION: What is free software? *http://www.gnu.org/philosophy/free-sw.en.html* (04.01.2013)
- FOSSGIS e.V. (2013): Veranstaltungsort für die FOSSGIS 2013. *http://www.fossgis.de/* (08.01.2013).
- FU & SUN (2011): Web GIS - Principles and Applications. Redlands (CA).
- BUNESAMT FÜR KARTOGRAPHIE UND GEODÄSIE (2013): GDI-DE Testsuite. *http://www.geoportal.de/DE/GDI-DE/Komponenten/GDI-DE-Testsuite/gdi-de-testsuite.html?lang=de* (03.01.2013)
- HAKLAY et al. (2008): Web Mapping 2.0: The Neogeography of the GeoWeb. In: Geography Compass 2/6: 2011–2039. *http://onlinelibrary.wiley.com/doi/10.1111/j.1749-8198.2008.00167.x/abstract*
- HEINRICH, M., HETTEL, T. (2012). Das Geheimnis der Echtzeit-Kollaboration. *http://www.heise.de/developer/artikel/Synchronisationsalgorithmen-verstehen-1562877.html (06.01.2013)*
- KORDUAN & ZEHNER. 2008: Geoinformation im Internet: Technologien zur Nutzung raumbezogener Informationen im WWW. Berlin.
- KOSCHMIDER et al. (2009): Elucidating the Mashup Hype: Definition, Challenges - Methodical Guide and Tools for Mashups. *http://mashup.pubs.dbs.uni-leipzig.de/files/paper14[1].pdf* (02.01.2013)
- LI & GONG (2008): Mashup: A new way of Providing Web Mapping/GIS Services. In: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII. Part B4. Beijing 2008. S. 639-647
- LI et al. (Hg.)(2011): Advances is Web-based GIS, Mapping Services and Applications. London.
- MEYER-RENSCHHAUSEN, E. (2011): Gemeinschaftlich betriebene Gemüsegärten in Berlin. Eine Studie. Berlin.
- MÜLLER, C. (2007): Interkulturelle Gärten – Urbane Orte der Subsistenzproduktion und der Vielfalt. In: Deutsche Zeitschrift für Kommunalwissenschaften 1/2007. S. 55-67.
- MÜLLER, C. (2011): Urban Gardening. Über die Rückkehr der Gärten in die Stadt. München.
- NÁRODNÁ INFRAŠTRUKTÚRA PRIESTOROVÝCH INFORMÁCIÍ [ohne Datum]: Tools for distribution of geospatial information: Creation of webmap services with use of environment MapServer – model locality “Bánovce nad Bebravou“. Bratislava. *http://geonet.fns.uniba.sk/en/research_11.htm* (03.01.2012)

- OPEN DATA COMMONS (2012a). Licences FAQ. *http://opendatacommons.org/faq/licenses/#db-versus-contents* (04.01.2013)
- OPEN DATA COMMONS: Open Database License 1.0. *http://opendatacommons.org/licenses/odbl/1-0/* (03.01.2013)
- OPEN SOURCE INITIATIVE (2012): Licences by Name. Palo Alto. 18.07.2012, *http://opensource.org/licenses/alphabetical* (02.01.2013)
- PROVE (2008): Zitat - Abgerufen am 04.12.2008 durch Matthias Müller-Prove, *http://delicious.com/mprove/KaiKrause*, abgerufen am (03.01.2013)
- SCHUURMAN, N. (2004): GIS – A short Introduction. Malden, Oxford, Victoria (Aus).
- STALLMAN, R. (2010): Free Software, Free Society - Selected Essays By Richard M. Stallman. Boston.
- STALLMAN, R. (2006): Free Software Foundation Europe Interview - Richard Stallman at the 2nd international GPLv3 conference ; 21st April 2006. *http://fsfe.org/campaigns/gplv3/fisl-rms-transcript#licence-compatibility* (05.01.2013)
- SCHARF, P. (2011): Temporaer-urbane Landwirtschaft. Gartenbau als Brachflaechenzwischennutzung. Diplomarbeit an der Hochschule Dresden zur Erlangung des akademischen Grads Diplomingenieur.
- TURNER, A. (2006): Introduction to Neogeography. *http://pcmlp.socleg.ox.ac.uk/sites/pcmlp.socleg.ox.ac.uk/files/Introduction_to_Neogeography.pdf* (05.01.2013)
- TORFS, P.; BRAUER, C. (2012): A (very) short introduction to R. *http://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf* (02.01.2012)
- WARREN, J. (2006): Grassroots Mapping: tools for participatory and activist cartography. Degree of Master of Science at the Massachussetts Institut fo Technology.
- ZEILEIS, A.; TÜCHLER, R. (2006): Dataframes in R. *http://statmath.wu.ac.at/courses/multverf1/DatenVK2.pdf*
- ZER–AVIV, M.; LINKSVAYER, M.; MANDIBERG, M., PEIRANO, M., TONER, A., HYDE, A., KANARINKA, TARKA, S., TAYLOR, A. (2010): Collaboarative Futures. A Book About the Future of Collaboration, Written Collaboratively. Berlin. New York.
- ZUMBACH, J. & RAPP, A. (2001): *Hypermedien und Wissenskonstruktion*, Osnabrücker Beiträge zur Sprachtheorie, Heft 63 ([8]), darin: : Wissenserwerb mit Hypermedien. Eine kognitionswissenschaftliche Betrachtung, S. 27-44

Abbildungsverzeichnis

Abbildung 1 : Diagramm verschiedener Softwarekategorien.
Abbildung 2 : API Auswahl.
Abbildung 3 : Abstraktion der Daten-, Dienst- und Präsentationsschicht.
Abbildung 4 : Ergebnis der ersten erfolgreichen Abfrage des Mapservers.
Abbildung 5 : Ergebnis : Berliner Gartenkarte.
Abbildung 6 : Eine erste Version der Gartenkarte als Druckversion.
Abbildung 7 : Gemeinschaftsgärten in Berlin nach Bezirken.

Codeverzeichnis

gartenkarte.r

siehe *Abschnitt 5*

```
1.  # Pakete
2.      library("sp")
3.      library(rgdal)
4.      library(maptools)
5.      library(RColorBrewer)
6.      library(classInt)
7.      library(ggplot2)
8.      library(plyr)
9.
10. # DATENAUFBEREITUNG
11. # Laden der Gartenadressen als .csv
12.     adressen <- read.csv(file="Gartenadressen.csv",head=TRUE,sep=",")
13.
14. # Datensatzbereinigung
15.     adressen_sort <- adressen[order(adressen$Titel),] # sortieren für Karte
16.     adressen_aufb <- subset(adressen_sort, select = c(Titel, ProjektCode, lat,
    lon)) # Var auswählen
17.
18.
19. # Auswahl aller Berliner Adressen nach KOS
20.     adressen_lat <- subset(adressen_aufb, lat >= 52.373922)
21.     adressen_lat2 <- subset(adressen_lat, lat <= 52.675549)
22.     adressen_lon_berlin <- subset(adressen_lat2, lon <= 13.757400)
23.     adressen_lon2_berlin <- subset(adressen_lon_berlin, lon >= 13.092728)
24.
25.     write.csv (adressen_lon2_berlin, file = "Gartenadressen_berlin.csv")
26. # speichern
27.     adressen_berlin <-
    read.csv(file="Gartenadressen_berlin.csv",head=TRUE,sep=",")  # berliner adressen
    laden
28.
```

```
# Nummerierung
    adressen_berlin$Nr <- row.names(adressen_berlin)
    write.csv (adressen_berlin, file = "Gartenadressen_berlin.csv")            # speichern


# GEODATEN UND EXPORT
# Umwandeln in Spatial Class
    sp_point <- cbind(adressen_berlin$lon, adressen_berlin$lat)
    colnames(sp_point) <- c("LONG", "LAT")

    proj <- CRS("+proj=longlat +datum=WGS84")
    gaerten_berlin <- SpatialPointsDataFrame(coords=sp_point, data=adressen_berlin, proj4string=proj)


# ZUORDNUNG DES BEZIRKES
# shp- laden (Bezirke)
    Bezirke=readOGR(".", layer="01_Bezirke_Polygon_ALL", head=TRUE)

# point in polygon
    proj4string(Bezirke) <- proj4string(gaerten_berlin) # beide dataframes haben das geiche lat/lon Referenzsystem

    gaerten_berlin$Bezirke <- over(gaerten_berlin, Bezirke)$name # Bezirk den Gärten hinzufügen

# EXPORT
# als shp exportieren
    drv <- "ESRI Shapefile"
    writeOGR(gaerten_berlin, dsn = "gaerten_berlin.shp", layer = "gaerten_berlin", driver = drv)

# als geoJSON exportieren
    drvJson <- "GeoJSON"
    writeOGR(gaerten_berlin, dsn = "gaerten_berlin_json.geojson", layer = "gaerten_berlin_json", driver = drvJson)

# Textdatei mit Nr und Name für die Karte
```

```
62.     txt_Nr_Name <- subset(adressen_berlin, select = c(Nr, Titel))
63.     write.table(txt_Nr_Name, file="Nr_Name.txt",sep="\t",append=TRUE, col.names =
        FALSE, quote = FALSE, row.names = FALSE)
64.
65.  # ANALYSE
66.  # ZEITREIHE: wieviel Gärten im Bezirk + Ausgabe in .txt mit Datum
67.     gaerten_nach_bez <- sort(table(gaerten_berlin$Bezirke), decreasing = TRUE)
68.     gaerten_bezirk <- as.data.frame(gaerten_nach_bez)
69.     date <- date()
70.     write.table(gaerten_bezirk,
71.  file="Gaerten_nach_Bezirken.txt",sep=",",append=TRUE, col.names = x, quote =
        FALSE)
72.
73.  # BEZIRK > 2: Abfrage: in welchem Bezirk (administrativ) Gärten < 2?
74.
75.     y <- gaerten_bezirk$gaerten_nach_bez
76.     y > 2
77.
78.     Bez_kl_2 <- subset(gaerten_bezirk, gaerten_nach_bez < 2)
79.
80.
81.  # Karte, Fraben nach Häufigkeit/Gärten
82.     # Berechnete Gärten in SpatialPolygon Bezirke einfügen
83.     write.table(gaerten_berlin, file="Gaerten_nach_Bezirken2.csv")
84.     gaerten_in_bezirk <-
        read.csv(file="Gaerten_nach_Bezirken2.csv",head=TRUE,sep=",")
85.     y <- gaerten_in_bezirk[order(gaerten_in_bezirk$name),]
86.     Bezirke_sort <- Bezirke[order(Bezirke$name),]
87.     Bezirke_sort$x <- y$gaerten_nach_bez
88.
89.     # Plotten
90.     plotvar <- Bezirke_sort@data$x
91.     nclr <- 7
92.     plotclr <- brewer.pal(nclr,"Greens")
93.     class <- classIntervals(plotvar, nclr, style="equal")
94.     colcode <- findColours(class, plotclr)
95.
96.     spplot(Bezirke_sort, "x", col.regions=plotclr, at=round(class$brks,
        digits=2), main = "Häufigkeit von Gärten nach Bezirken", sub=date, lwd=.8,
```

```
        col="white")
```

gartenkarte.js

siehe *Abschnitt 4.7*

```
1.  /*
2.
3.      * gartenkarte.de - berliner gartenkarte
4.      *
5.      * This script initializes and configures OpenLayers to display a map of
6.      * Urban Gardening Projects in Berlin.
7.      *
8.      * This script may contain parts or ideas originating from different authors.
9.      * They keep their respective rights.
10.     *
11.     * Placed in the Public Domain by
12.     * Jon Richter, Lennart Wiesiolek, Max Godt
13.     *
14.     * Berlin, 06.01.2013
15.     *
16.
17. */
18.
19. var map;
20.
21. //  * <popup_control>
22. function showPopup(evt) {
23.     feature = evt.feature;
24.
25.     var lonlat = new OpenLayers.LonLat(feature.geometry.x,feature.geometry.y);
26.     var contentSize = new OpenLayers.Size(800,800);
27.     var contentHTML = feature.attributes.Titel;
28.     var anchor = new OpenLayers.Icon("./icon.png",contentSize,feature.geometry);
29.
30.     popup = new OpenLayers.Popup.Anchored("popup",
```

```
31.         lonlat,
32.         contentSize,
33.         contentHTML,
34.         null,
35.         false,
36.         null);
37.     popup.autoSize = true;
38.     popup.maxSize = new OpenLayers.Size(400,400);
39.     popup.fixedRelativePosition = true;
40.     feature.popup = popup;
41.     popup.feature = feature
42.     map.addPopup(popup, true);
43. }
44. 
45. function hidePopup(evt){
46.     feature = evt.feature;
47.     if (feature.popup){
48.         popup.feature = null;
49.         map.removePopup(feature.popup);
50.         feature.popup.destroy();
51.         feature.popup = null;
52.     }
53. }
54. // * </popup_control>
55. //  * <openlayers_initialization>
56. function init(){
57. 
58.     // * <initialize_map_properties>
59.         // * <configure_bounding_box />
60.         var gakaBounds = new OpenLayers.Bounds(
61.             13.0628,52.3279,
62.             13.764,52.68
63.         );
64.     // * <declare_map_option>
65.         var options = {
66.             theme: null,
67.             projection: new OpenLayers.Projection("EPSG:900913"),
68.             displayProjection: new OpenLayers.Projection("EPSG:4326"),
69.             controls:[
```

```
70.                 new OpenLayers.Control.TouchNavigation({
71.                     dragPanOptions: {
72.                         enableKinetic: true
73.                     }
74.                 }),
75.                 new OpenLayers.Control.Navigation(),
76.                 new OpenLayers.Control.ZoomPanel(),
77.                 new OpenLayers.Control.LayerSwitcher({roundedCornerColor :
78. '#48474C' }),
79.                 new OpenLayers.Control.Attribution(),
80.                 new OpenLayers.Control.ScaleLine(),
                    ],
81.         };
82.         // * </declare_map_options>
83.      // * <draw map />
84.         map = new OpenLayers.Map('map', options);
85.     // * </initialize_map_properties>
86.     // * <initialize_map_layers>
87.
88.         // * <initialize_mapquest_layer>
89.             // * <initialize_OpenLayers.Layer.MapQuestOSM_WMS_driver>
90.             OpenLayers.Layer.MapQuestOSM =
91. OpenLayers.Class(OpenLayers.Layer.XYZ, {
92.                 visibility:false,
93.                 isBaseLayer:true,
94.                 name: "OpenStreetMap MapQuest",
95.                 attribution: "MapQuest Layer Data CC-By-SA by <a
    href='http://openstreetmap.org/'>OpenStreetMap</a>",
96.                 sphericalMercator: true,
97.                 url: '
98. http://otile1.mqcdn.com/tiles/1.0.0/osm/${z}/${x}/${y}.png',
99.                 clone: function(obj) {
                        if (obj == null) {
100                         obj = new OpenLayers.Layer.OSM(
101.                        this.name, this.url, this.getOptions());
                        }
102.                    obj = OpenLayers.Layer.XYZ.prototype.clone.apply(this,
103. [obj]);
104.                    return obj;
```

```
105.                 },
106.                 CLASS_NAME: "OpenLayers.Layer.MapQuestOSM"
107.             });
                 // * </initialize_OpenLayers.Layer.MapQuestOSM_WMS_driver>
108.
109.             // <assign_mapquest_layer />
110.             var layerMapQuest = new OpenLayers.Layer.MapQuestOSM();
111.         // * </initialize_mapquest_layer>
112.
113.         // * <initialize_OSM_default_layer />
114.             var layerMapnik = new OpenLayers.Layer.OSM.Mapnik("OpenStreetMap",
115.                 {
116.                     visibility:false,
117.                     isBaseLayer:true,
118.                 } );
119.
120.         // * <initialize_stamen_watercolor_layer>
121.             var layerStamen = new OpenLayers.Layer.Stamen("watercolor",
122.                 {
123.                     isBaseLayer:true,
124.                     attribution: "Watercolor Map Tiles by <a
125. href='http://stamen.com' target=_blank>Stamen Design</a>, under <a
126. href='http://creativecommons.org/licenses/by/3.0' target=_blank>CC BY 3.0</a>.
127. Data by <a href='http://openstreetmap.org' target=_blank>OpenStreetMap</a>,
128. under <a href='http://creativecommons.org/licenses/by-sa/3.0' target=_blank>CC
129. BY SA 3.0</a>.",
                 });

                 // * <fix_stamen_layer_title />
                 layerStamen.setName('Stamen Watercolor');
         // * </initialize_stamen_watercolor_layer>
130.
131.         // * <initialize_geojson_marker_layer>
132.
133.             // * <style_geojson_vector_features />
134.             var styleGeoJSON = new OpenLayers.StyleMap({
135.                 pointRadius: 16,
136.                 externalGraphic: './icon.png',
137.             })
```

```
138.
139.            // * <initialize_geojson_layer>
140.            var layerGeoJSON = new OpenLayers.Layer.Vector("Urbane Gärten in
141. Berlin", {
142.                // * <activate_popup_event_listeners />
143.                eventListeners:{
144.                    'featureselected': showPopup,
145.                    'featureunselected': hidePopup,
                    },
146.                // * <configure_geojson_layer />
147.                projection: new OpenLayers.Projection("EPSG:4326"),
148.                attribution: "Daten der Urbanen Gärten <a
149. href='http://creativecommons.org/licenses/by-nc-sa/3.0/deed.de' target=_blank>CC
150. BY NC SA</a> von <a href='http://stadtacker.net'
151. target=_blank>stadtacker.net</a>",
152.                styleMap: styleGeoJSON,
153.
                    // * create_vector_layer />
                    strategies: [new OpenLayers.Strategy.Fixed()],
                    // * <add_feature_to_layer />
154.                protocol: new OpenLayers.Protocol.HTTP({
155.                    url: "./gartenkarte.geojson",
156.                    format: new OpenLayers.Format.GeoJSON()
157.                }),
158.            });
159.            // * </initialize_geojson_layer>
160.
161.            // * <create_selectFeature_control />
162.            var selectorGeoJSON = new
163. OpenLayers.Control.SelectFeature(layerGeoJSON,{
164.                hover:true,
165.                autoActivate:true,
166.            });
167.
                // * <add_layer_popup_control_to_map />
168.            map.addControl(selectorGeoJSON);
169.
170.        // * </initialize_geojson_marker_layer>
171.
```

```
172.          // * <add_layers_to_map />
173.          map.addLayers([layerStamen, layerMapQuest, layerMapnik, layerGeoJSON ]);
174.      // * </initialize_map_layers>
175.
176.      // * <display_map_extent />
177.      map.zoomToExtent(
178.          gakaBounds.transform(map.displayProjection, map.projection)
179.      );
180.  }
181.  // * </openlayers_initialization>
```

tile_merger.py

siehe *Abschnitt 5.2.2*

```
1.  ### Merge tiles into one image ###
2.  import sys, os
3.
4.  print "Usage: tile_merge.py zoomlevel xMin xMax yMin yMax filename"
5.  print
6.  print "This utility merges tiles."
7.
8.  zoom = None
9.  xMin, xMax, yMin, yMax = None, None, None, None
10. filename = None
11.
12. argv = sys.argv
13. i = 1
14. while i < len(argv):
15.     arg = argv[i]
16.
17.     if zoom is None:
18.         zoom = int(argv[i])
19.     elif xMin is None:
20.         xMin = int(argv[i])
21.     elif xMax is None:
22.         xMax = int(argv[i])
23.     elif yMin is None:
24.         yMin = int(argv[i])
25.     elif yMax is None:
26.         yMax = int(argv[i])
27.     elif filename is None:
28.         filename = argv[i]
29.     else:
30.         Usage("ERROR: Too many parameters")
31.
32.     i = i + 1
33.
34. import Image
35. tileSize = 256
```

```
36. tileDir = os.path.join(os.curdir,"tiles",str(zoom))
37.
38. out = Image.new( 'RGB', ( (xMax - xMin + 1) * tileSize, (yMax - yMin + 1) *
    tileSize) )
39.
40. imx = 0
41. for x in range(xMin, xMax+1):
42.     imy = 0
43.     for y in range(yMin, yMax+1):
44.         tileFile = os.path.join(tileDir,str(x),str(y)+".jpg")
45.         tile = Image.open(tileFile)
46.         out.paste( tile, (imx, imy) )
47.         imy += tileSize
48.     imx += tileSize
49.
50. out.save(os.path.join(os.curdir,filename))
51. print "done"
```

tile_scraper.sh

siehe *Abschnitt 5.2.2*

```
#! /bin/bash

if [ $# -lt 5 ]; then
        echo "Usage: zoom topleft_x topleft_y bottomright_x bottomright_y"
        exit
fi

#echo -n "Herunterladen der watercolor tiles für die gaka, zoomstufe: "
ZOOM=$1
#read -e ZOOM
echo Zoomstufe ist $ZOOM

TOPX=$2
echo -n "topleft_x: $TOPX"
#read -e TOPX

TOPY=$3
echo -n "topleft_y: "
#read -e TOPY
BOTTOMX=$4
echo -n "bottomright_x: "
#read -e BOTTOMX
BOTTOMY=$5
echo -n "bottomright_y: "
#read -e BOTTOMY

let DELTAX=BOTTOMX-TOPX
echo "deltax ist $DELTAX"
let DELTAY=BOTTOMY-TOPY
echo "deltay ist $DELTAY"

if [ ! -d "$ZOOM" ]; then
    mkdir $ZOOM
fi
```

```
35. cd $ZOOM
36. 
37. for i in `seq 0 $DELTAX`
38. do
39.     let X=TOPX+i
40. 
41.     if [ ! -d "$X" ]; then
42.         mkdir $X
43.     fi
44.     cd $X
45. 
46.     for j in `seq 0 $DELTAY`
47.     do
48.         let Y=TOPY+j
49. 
50.         echo "$ZOOM / $X / $Y"
51.         if [ ! -f "$Y.jpg" ]; then
52.             wget http://d.tile.stamen.com/watercolor/`printf $ZOOM`/`printf
    $X`/`printf $Y`.jpg
53.         else
54.             echo "Tile $ZOOM/$X/$Y ist schon gespeichert."
55.         fi
56.     done
57.     cd ..
58. done
59. 
60. cd ..
61. echo fertig.
```