Diplomarbeit

Stand der Technik und Potenziale von *Smart Map Browsing* im Webbrowser

am Beispiel der Freien WebMapping-Anwendung *OpenLayers*

Vorgelegt von
Emanuel Schütze

unter der Betreuung von
Prof. Dr.-Ing. Heide-Rose Vatterrott
(Hochschule Bremen)

Dr. rer. nat. Dipl.-Systemwiss. Jan-Oliver Wagner
(Intevation GmbH, Osnabrück)

im Studiengang
Medieninformatik
Hochschule Bremen

Osnabrück, am 1. Juni 2007

Diese Diplomarbeit ist unter **http://smartmapbrowsing.org** erhältlich.
Kontakt: emanuel@smartmapbrowsing.org

Zusammenfassung

Die Nutzung von WebMapping-Anwendungen hat in den letzten Jahren stark zugenommen. Der wachsende Nutzerkreis macht es erforderlich, verstärkt die Gebrauchstauglichkeit von webbasierten Kartenanwendungen zu untersuchen. In dieser Entwicklung spielt Freie Software eine bedeutende Rolle.

Im Rahmen dieser Arbeit wird die Usability von WebMapping-Anwendungen thematisiert und in folgenden drei Schritten dargestellt:

Bestandsanalyse: Zunächst wird die Ausgangssituation analysiert. Dazu werden elf ausgewählte Freie WebMapping-Anwendungen und *Google Maps* auf ihre Gebrauchstauglichkeit untersucht.

***Smart Map Browsing*:** Auf Grundlage der Analyseergebnisse wird in dieser Arbeit der Begriff *Smart Map Browsing* als neuer Ausdruck für die Usability von WebMapping-Anwendungen geprägt. Die Eigenschaften von *Smart Map Browsing* werden nach dem aktuellen Stand der Technik abgeleitet und auf ihre Potenziale untersucht.

Realisierung: Im praktischen Entwicklungsteil dieser Arbeit wird das *Smart Map Browsing* Feature *animated zooming* ausgewählt und für die Freie JavaScript-WebMapping-Anwendung *OpenLayers* realisiert. Mit dieser Erweiterung wird die Karte beim Zoomwechsel bis zur neuen Zoomstufe skaliert, um die Orientierung des Nutzers beim Zoomvorgang zu verbessern.

Ein erster Teil der durchgeführten Software-Entwicklungsarbeit ist bereits im April 2007 in die *OpenLayers*-Version 2.4 RC2 eingeflossen und veröffentlicht worden. Das komplette *animated zooming* Feature wird voraussichtlich in die Version 2.5 von *OpenLayers* mit aufgenommen.

Abstract

The use of web mapping applications has increased considerably over the last number of years. The growing user base points to a need to investigate the user friendliness of web-based mapping applications. Free software also plays an important part in this development.

This thesis analyzes the usability of web mapping applications using the following three steps:

Analysis: First, the usability of eleven selected Free web mapping applications and Google Maps will be analysed.

***Smart Map Browsing*:** Based on the results of this analysis the term *Smart Map Browsing* will be used to indicate the usability of web mapping applications. The properties of *Smart Map Browsing* will be derived from the current state of technology and analyzed for their future potential.

Realization: Finally, for the practical part of this thesis, the *Smart Map Browsing* feature *animated zooming* is implemented into the Free JavaScript web mapping application *OpenLayers*. This extension allows the user to scale a map to a new zoom level during the zooming process.

A first phase of the software development is already integrated into *OpenLayers* 2.4 RC2 (released in April 2007). The entire *animated zooming* feature is expected to be included in *OpenLayers* 2.5.

Inhaltsverzeichnis

1 Einleitung

Der Gebrauch von webbasierten Karten verzeichnete in den letzten Jahren eine rasante Zunahme. Schätzungsweise 40 Millionen Karten pro Tag wurden im Jahre 1999 im Internet aufgerufen. Zwei Jahre später waren es bereits etwa 200 Millionen Karten – ein Wert, der sich gegenwärtig vervielfacht haben dürfte und die zunehmende Bedeutung der Visualisierung raumbezogener Daten im Internet verdeutlicht [Peterson, 2003; Dickmann, 2004].

Die Veröffentlichung von *Google Maps* im Februar 2005 war eine Sensation im WebMapping-Bereich. *Google Maps* beeindruckte mit einer hohen Geschwindigkeit und einer hochgradigen Interaktivität. Zusätzlich sorgten hochauflösende Satellitenbilder und ein attraktives Kartendesign dafür, dass die neuartige, webbasierte Kartenanwendung weltweit große Aufmerksamkeit erlangte [Ramsey, 2006]. *Google Maps* verstand es durch eine reine, clientseitige JavaScript-Implementierung die Benutzbarkeit von WebMapping-Anwendungen zu revolutionieren. Ein Aspekt, der bis dahin von vielen Applikationen vernachlässigt wurde. Dabei hat die Usability einen entscheidenen Einfluss auf die Zufriedenheit des Benutzers, und damit auf den Erfolg und die Verbreitung einer Anwendung.

In der Literatur gibt es aktuell nur sehr wenig Quellen, die sich mit der Usability von WebMapping-Anwendungen befassen. Analysen oder empirische Untersuchungen von Freien WebMapping-Lösungen unter Berücksichtigung ihrer Usability wurden bisher nicht veröffentlicht.
Die vorliegende Arbeit untersucht die Gebrauchstauglichkeit von WebMapping-Anwendungen und beschreibt diese Thematik mit dem Begriff *Smart Map Browsing*. Dieser Ausdruck wurde bisher in keiner Literatur gefunden und wird in dieser Arbeit geprägt. Die genaue Bedeutung des Begriffes wird im Kapitel 3.2 definiert. Diese Arbeit stellt damit einen Versuch dar, einen Beitrag zur fachlichen Weiterentwicklung im Bereich der Usability von WebMapping-Anwendungen zu leisten.

1.1 Zielsetzung

Die folgenden Zeilen beschreiben die ursprüngliche, unveränderte Formulierung der Zielsetzung, die zu Beginn dieser Arbeit verfasst wurde:

Das Ziel dieser Diplomarbeit ist es, den aktuellen Stand der clientseitigen WebMapping-Technik zu untersuchen. Hierbei sollen Freie Internet-Map-Applikationen mit ihren unterschiedlichen *Smart Map Browsing* Eigenschaften im Mittelpunkt stehen.

Ausgehend von den Untersuchungsergebnissen wird der Begriff *Smart Map Browsing* definiert.

Im praktischen Entwicklungsteil dieser Diplomarbeit soll ein *Smart Map Browsing* Feature ausgewählt und für reine JavaScript-WebMapping-Anwendungen realisiert werden. Das Ziel der Erweiterung ist es, die Benutzbarkeit von solchen Anwendungen in Webbrowsern zu verbessern. Die Lösung sollte nach Möglichkeit allgemeingültig sein, so dass eine problemlose Integration in vorhandene Freie WebMapping-Anwendungen gewährleistet ist.

1.2 Gliederung der Arbeit

Kapitel 2 (Grundlagen) beschreibt die wichtigsten Grundlagen dieser Arbeit. Hier werden die Begriffe *Freie Software*, *Usability* und *WebMapping* erläutert, das *Open Geospatial Consortium* vorgestellt sowie die Grundlagen von *Web Map Services* erklärt.

Kapitel 3 (Analyse) widmet sich zunächst einer Bestandsanalyse von ausgewählten WebMapping-Anwendungen, deren Ergebnisse anschließend den Begriff *Smart Map Browsing* definieren helfen. Es folgt eine Anforderunsanaylse für das *animated zooming* Feature. Die technische Untersuchung von *OpenLayers* bildet den Abschluss dieses Kapitels.

Kapitel 4 (Konzept) umfasst das konkrete Konzept für die Umsetzung der geplanten *OpenLayers*-Erweiterung. Neben technischen Details wird hier auch auf die Entwicklungsrichtlinien des *OpenLayers*-Projekts eingegangen, die für die Realisierung beachtet werden müssen.

Kapitel 5 (Realisierung) erläutert die einzelnen Schritte der Implementierung des *animated zooming* Features sowie das Vorgehen beim Testen und die insgesamt dabei aufgetretenen Schwierigkeiten.

Kapitel 6 (Schlussfolgerung) fasst die Ergebnisse dieser Arbeit zusammen, diskutiert die offen gebliebenen Probleme und schließt mit einem Ausblick.

Anhang A (Bestandsanalyse) beinhaltet die ausführlichen Analyseergebnisse der zwölf untersuchten WebMapping-Anwendungen aus Kapitel 3.

Anhang B (Klassendiagramm) stellt ein vollständiges Klassendiagramm von *OpenLayers* dar, wobei alle in der Implementierung geänderten Klassen und Funktionen farbig markiert sind.

Anhang C (Testplan) umfasst den erstellten Testplan für die unter Kapitel 5 durchgeführten Integrationstests.

Anhang F (CD-ROM) enthält eine CD-ROM mit den entwickelten *animated zooming* und *panning* Erweiterungen integriert in die *OpenLayers*-Version 2.4 RC5 vom 25.5.2007.

2 Grundlagen

Dieses Kapitel erläutert die Grundlagen, die für das Verständnis der vorliegenden Arbeit von Bedeutung sind.

2.1 Freie Software

2.1.1 Begriff

Das Wort »Frei« in dem Begriff *Freie Software (FS)* bezieht sich nicht auf den Preis, sondern ist im Sinne von »Freiheit« zu verstehen.

Der Begriff *Freie Software* lässt sich durch folgende 4 Freiheiten definieren [FSF Europe]:

1. Die Freiheit, das Programm für jeden Zweck auszuführen.
2. Die Freiheit, die Funktionsweise eines Programms zu untersuchen, und es an seine Bedürfnisse anzupassen.
3. Die Freiheit, Kopien weiterzugeben und damit seinen Mitmenschen zu helfen.
4. Die Freiheit, ein Programm zu verbessern, und die Verbesserungen an die Öffentlichkeit weiterzugeben, sodass die gesamte Gesellschaft profitiert.

Ein Programm ist dann Freie Software, wenn es Benutzern alle diese Freiheiten gewährt. Andernfalls wird die Software als proprietär oder unfrei bezeichnet. In dieser Arbeit wird ein großgeschriebenes »Frei« im Zusammenhang mit Software immer im Sinne der o. g. Freiheiten von Freier Software gebraucht.

1985 gründete Richard Stallman die *Free Software Foundation (FSF)* mit dem Ziel Freie Software zu fördern [Grassmuck, 2004]. Freie Software schließt (nach Freiheit 1) eine kommerzielle Nutzung nicht aus. Dass Geschäftsmodelle auf Basis Freier Software bzw. Open Source durchaus eine wirtschaftliche Bedeutung haben, zeigt eine Marktstudie der EU-Kommission von Januar 2007 [heise open]:

> »*Für das Jahr 2010 sagen die EU-Forscher Open-Source-bezogenen Dienstleistungen einen Anteil von 32 Prozent am gesamten IT-Dienstleistungssektor voraus. Der Open-Source-Anteil am Bruttoinlandsprodukt könnte 2010 bereits 4 Prozent betragen. Derzeit macht die gesamte IT-Infrastruktur 10 Prozent des europäischen BIP aus.*«

2.1.2 Freie Software versus Open Source

Freie Software wird oft irrtürmlich mit dem Begriff *Open Source* gleichgesetzt.

Hintergrund: Der Begriff *Freie Software* hat ein Problem der Zweideutigkeit. »Frei« wird fälschlicherweise oft im Sinne von kostenlos und nicht (wie nach der FSF-Definition) im Sinne von Freiheit verstanden. Dies ist *ein* Grund warum 1998 die *Open Source Initiative (OSI)*[1] damit begonnen hat, den Begriff *Open Source* anstelle von Freier Software zu benutzen. Das Ziel dieser Marketingkampagne war es, die Kommerzialisierung und Akzeptanz von Freier Software besonders im Unternehmensumfeld voranzutreiben. Dabei wurde auf die philosophischen, ethischen und gesellschaftlichen Ideen von Freier Software verzichtet und sich ausschließlich auf den technischen Vorteil der Quelloffenheit konzentriert [FSF Europe]. Open Source bezieht sich demnach nur auf den Zugang zum Quellcode. Im Vergleich zu Freier Software ist dies ein wesentlich schwächeres Kriterium [GNU-Projekt, a].

2.1.3 Lizenzkategorien

Software-Lizenzen beschreiben die Nutzungsrechte von Software. Es handelt sich um eine Freie Software-Lizenz, wenn in ihr alle unter Abschnitt 2.1.1 genannten vier Freiheiten ausdrücklich erlaubt werden. Freie Software-Lizenzen sind genauso bindend wie Lizenzen proprietärer Software.

Wird eine Software (mit oder ohne Änderungen) weitergegeben, greifen die Lizenzbedingungen. Abbildung 2.1 kategorisiert die zahlreichen Freien Software-Lizenzen auf Basis der Schutzwirkung der Freiheit nach [Reiter, 2004] in folgende vier Kategorien:

1. *Stark schützend*
   z. B. GNU General Public License (GPL): Der Lizenznehmer verpflichtet sich bei Weitergabe einer veränderten GNU GPL Software, diese wieder komplett unter der gleichen Lizenz mit Quelltext zur Verfügung zu stellen.
2. *Schwach schützend*
   z. B. GNU Lesser General Public License (LGPL): GNU LGPL Software darf in proprietärer Software verwendet werden. Bei Weitergabe muss nur der GNU LGPL Quelltext frei bleiben.
3. *Nicht schützend*
   z. B. X11 (modifizierte BSD): Die Software kann ohne Quelltext und Freiheiten weitergegeben und in proprietärer Software verwendet werden.
4. *Nicht GPL kompatibel oder unausgeglichen*
   alle Freien Software-Lizenzen, die nicht unter (1) bis (3) fallen; z. B. Netscape Public License

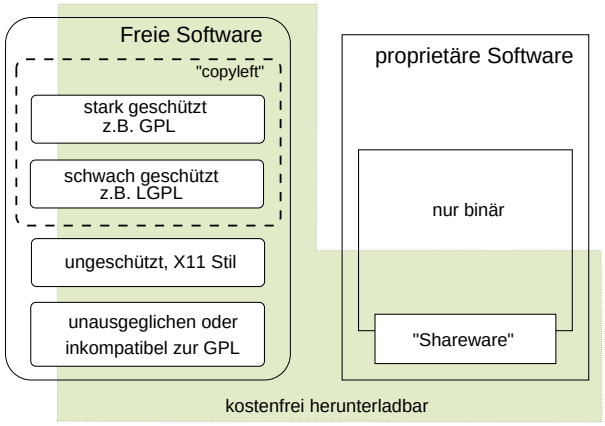


Abbildung 2.1: FS Lizenzkategorien (nach [Reiter, 2004])

Software mit mehreren unterschiedlichen Lizenzen aus (1) bis (3) fallen unter den Schutz der stärksten Lizenz [Reiter, 2004]. Freie Software-Lizenzen mit Schutz werden auch als *Copyleft* bezeichnet. Das Copyleft bedeutet, dass alle, die die Software (mit oder ohne Änderungen) weiter verteilen, auch die Freiheit zum Weitergeben und Verändern mitgeben müssen. Das Copyleft garantiert allen Benutzern diese Freiheit. Das GNU-Projekt beschreibt das Copyleft im Vergleich zum Copyright sehr anschaulich [GNU-Projekt, b]:

> »*Proprietäre Software-Entwickler verwenden das Copyright, um den Benutzern ihre Freiheit zu nehmen; wir verwenden es, um ihnen ihre Freiheit zu garantieren. Deshalb haben wir den Namen umgedreht und aus dem 'Copyright' das 'Copyleft' gemacht.*«

[1] http://www.opensource.org [Abruf: 15.05.2007]

2.2 Usability

2.2.1 Begriffe

Der englische Begriff *Usability* wird im Deutschen häufig mit einer Vielzahl von synonym verwendeten Wörtern gebraucht: Bedienbarkeit, Bedienerfreundlichkeit, Benutzbarkeit, Benutzerfreundlichkeit, Ergonomie, Gebrauchstauglichkeit, Handhabbarkeit, Nutzungsqualität, Software-Ergonomie, User Experience u. a. m. Der Begriff *Gebrauchstauglichkeit* hat sich allgemein als deutsche Übersetzung von *Usability* durchgesetzt. Das Wort »Gebrauch« steht hierbei für den Nutzungskontext in dem ein Produkt zum Einsatz kommt. Im Gegensatz zu einigen der oben genannten Synonyme (wie beispielsweise *Benutzerfreundlichkeit*) geht es hierbei nicht nur um den Benutzer selbst, sondern insbesondere auch um die Arbeitsaufgabe, die der Benutzer mit einem Produkt erledigen muss. Unterstützt ein Produkt eine Arbeitsaufgabe nicht im Sinne des Benutzers, so ist das Produkt also schlichtweg untauglich [FIT, 2005; Römeling, 2003].

1998 wurde der Begriff *Usability* erstmalig in der internationalen Norm ISO 9241-11 definiert und ein Jahr später ins Deutsche übersetzt [FIT, 2005]:

Englisch: »Usability« (ISO 9241-11:1998)

> *»Extend to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.«*

Deutsch: »Gebrauchstauglichkeit« (DIN EN ISO 9241-11:1999; dt. Übersetzung)

> *»Das Ausmaß in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen.«*

Jakob Nielsen gilt als einer der führenden Persönlichkeiten auf dem Gebiet der Software- und Webdesign-Gebrauchstauglichkeit und beschreibt Usability als den »Grad an Qualität, in welchem der Benutzer die Interaktion mit etwas erlebt« [Rampl].

In der vorliegenden Arbeit wird der Begriff *Usability* sowie die deutschen Begriffe *Gebrauchstauglichkeit* und *Benutzbarkeit* in gleicher Bedeutung verwendet.

2.2.2 Messbarkeit

Die Gebrauchstauglichkeit eines Produktes ist über die in der ISO-Norm definierten Kenngrößen – Effektivität, Effizienz und Zufriedenheit – messbar. Es können allerdings keine einheitlichen Richtwerte existieren, da die Bedeutung dieser Kenngrößen immer vom Nutzungskontext und dem Zweck der Gebrauchstauglichkeit abhängig ist. Usability sollte daher immer zweckbezogen betrachtet werden [Römeling, 2003]. Eine rein objektive Beurteilung der Gebrauchstauglichkeit ist aufgrund der unterschiedlichen Bedürfnisse der Nutzer, und den damit verbundenen subjektiven Empfindungen, kaum möglich.

Jakob Nielsen unterteilt die Gebrauchstauglichkeit in fünf messbare Usability-Faktoren und zeigt an einem interaktiven Anwendungssystem, wie facettenreich der Begriff Usability zu verstehen ist [Römeling, 2003; van der Vlugt u. Stanley, 2005]:

- **Learnability:** Das System sollte möglichst leicht zu erlernen sein, um schnellstens Arbeitsaufgaben zu bewältigen.
- **Efficiency:** Das System sollte zeitlich effizient zu nutzen sein und einen hohen Grad an Produktivität ermöglichen.
- **Memorability:** Die Bedienung des Systems sollte leicht erinnerbar sein, so dass nach einer späteren Rückkehr das System nutzbar ist, ohne sich wieder neu einarbeiten zu müssen.
- **Errors:** Das System sollte eine niedrige Fehlerrate besitzen.
- **Satisfaction:** Das System sollte angenehm zu benutzen sein, so dass sich Zufriedenheit bei der Benutzung einstellt.

Demnach kann die Usability als ein Maß für die Qualität der Benutzerinteraktion mit einem System betrachtet werden. Dieses Maß beeinflußt entscheidend die Akzeptanz des Nutzers und den Erfolg eine Systems.

2.3 WebMapping

2.3.1 Begriffe

Für die elektronische Kartendarstellung im Internet gibt es mittlerweile zahlreiche Begriffe: *Internet-Karte*, *Web-Map*, *netzbasierte E-Map*, *Cyber-Map* u. a. m.

Nach Jens Fitzke lassen sich drei verschieden Oberbegriffe für diese Karten definieren [Fitzke, 1999]:
GIS online dient als Sammelbegriff für alle Arten von GIS-Anwendungen, die über ein Netzwerk verfügbar sind. *Internet-GIS* schränkt das Betrachtungsfeld weiter ein und beschreibt stärker die auf bestimmten Protokollen basierende Netzwerktechnik. In den meisten Fällen lässt sich der Begriff noch enger fassen, in dem ein bestimmter Client (der Webbrowser) vorausgesetzt wird. In diesem Fall spricht man von einem *WebGIS*.

Frank Dickmann hingegen unterteilt die elektronischen Kartendarstellungen im Internet in zwei Begriffe: *WebMapping* und *WebGIS*.
WebMapping ist von dem Begriff *Web-Map* abgeleitet und bezieht sich demnach stärker auf den Herstellungsprozess der Karten. *WebMapping* umfasst neben der Visualisierung der Karte im Internet auch einfache Ansichtsmanipulationen, wie Zoomen, Verschieben sowie das Ein- und Ausblenden einzelner Ebenen (Layer). Diese Funktionen gelten in der Regel noch nicht als »GIS adäquat« [Dickmann, 2004].
In Abgrenzung dazu wird von einem *WebGIS* gesprochen, wenn der Nutzer über die *WebMapping*-Funktionalitäten hinaus Sachdaten selbständig ändern und weiterführende GIS-Operationen (wie themenbezogene Abfragen, Suchfunktionen, Flächen- und Streckenermittlungen) durchführen kann.

Eine einheitliche und klare Abgrenzung zwischen *WebMapping* und *WebGIS* existiert nicht. Beispielsweise werden vielfach grundlegende GIS-Analyse-Funktionalitäten, wie die Sachdatenabfrage, ebenfalls dem *WebMapping* zugeordnet [Dickmann, 2004].

In dieser Arbeit wird der Begriff *WebMapping* im weiteren Sinne nach Frank Dickmann verwendet und schließt dabei einfache analytische GIS-Funktionalitäten mit ein.

2.3.2 Klassifizierung

WebMapping-Anwendungen lassen sich nach dem Interaktionsgrad in *statische* und *interaktive* Kartendarstellungen klassifizieren [Mitchell, 2005].

Die sogenannten *static maps* oder *view only maps* sind vorgefertigte Karten, die sich unmittelbar als einzelne Grafiken in Webseiten integrieren lassen.
Mit Beginn der WebMapping-Entwicklung wurden Papierkarten gescannt und lediglich in Form von Bitmaps als statische Karten im Internet visualisiert. Es entstanden umfangreiche Online-Sammlungen von statischen Karten. Diese Art der Kartendarstellung ist noch heute sehr verbreitet [Dickmann, 2004; Mitchell, 2005].

Bei interaktiven Karten hat der Nutzer vielfältige Möglichkeiten den Kartenausschnitt individuell anzupassen oder weiterführende Informationen zu erhalten. Beispielsweise kann der Nutzer die Karte verschieben, hineinzoomen, weitere Kartenebenen aktivieren oder Abfragen zu einem Kartenobjekt anfordern. Der Kartenausschnitt wird dazu aktualisiert und neue Informationen werden dargestellt [Dickmann, 2004].

Technisch betrachtet ist die Grenze zwischen statischen und interaktiven Kartendarstellungen nicht ganz leicht zu ziehen. Es handelt sich auch dann noch um eine statische Karte, wenn die Karte verschoben (*panning*) bzw. in der Zoomstufe verändert wurde (*zooming*) und anschließend keine neuen Informationen dargestellt werden [Dickmann, 2004]. Demnach bekäme der Nutzer keinen »Mehrwert«, wenn beispielsweise beim Zoomen die Karte lediglich skaliert, weitere Details aber nicht ergänzt würden.

2.3.3 Architektur

Die Grundlage von WebMapping- und WebGIS-Anwendungen bildet das Client-Server-Modell. Als Client wird ein Webbrowser benutzt, der die Kommunikation mit dem Webserver übernimmt. Abbildung 2.2 veranschaulicht diesen Prozess: Der Client fordert eine Karte an, in dem er eine Anfrage formuliert und diese an den Webserver schickt. Der Webserver leitet die Kartenanfrage an den Mapserver weiter. Dieser wertet die Anfrage aus, greift auf die benötigten Geodaten zurück und generiert daraus eine entsprechende Grafik. Die fertige Karte wird nun an den Webserver gesendet, wo sie in eine HTML-Seite integriert und zurück an den Client-Browser geschickt wird [Klauer, 2002; Mitchell, 2005].

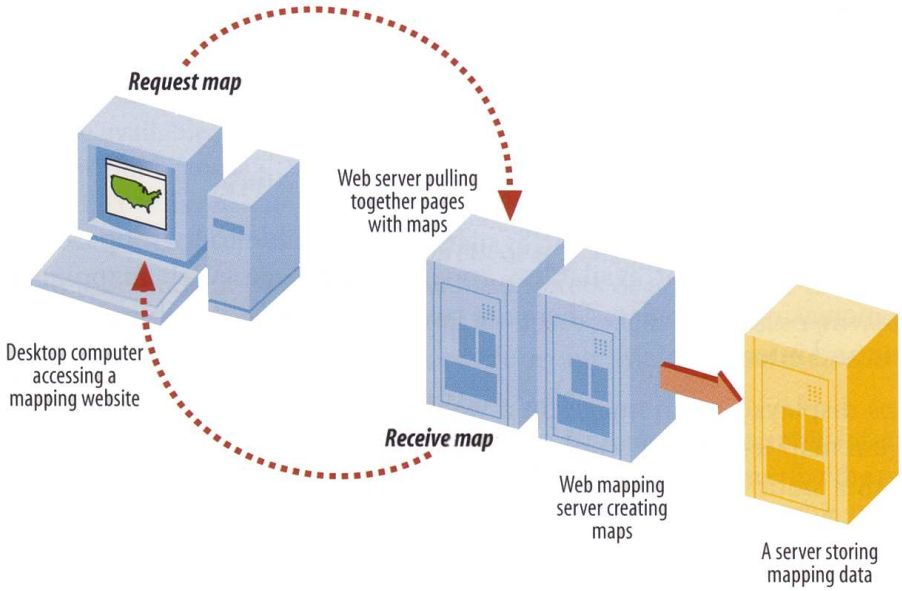


Abbildung 2.2: Client-Server-Architektur (nach [Mitchell, 2005])

Client-Technik

Für die Nutzung einer WebMapping-Anwendung wird auf der Clientseite lediglich ein Webbrowser vorausgesetzt. Dieser bietet standardmäßig die Möglichkeit Rasterdaten in den Formaten GIF, JPG und PNG darzustellen.
Vektordaten werden noch nicht von allen Browsern unterstützt. Mozilla Firefox[2] unterstützt beispielsweise in Version 2.0 nativ die SVG 1.1 Spezifikation[3]. Dieser noch nicht einheitlich durchgesetzte SVG-Standard in Browsern ist *ein* Grund, warum Vektordaten in WebMapping-Anwendungen derzeit noch eine eher untergeordnete Rolle spielen. Die Thematik WebMapping in Verbindung mit Vektordaten soll an dieser Stelle nicht weiter vertieft werden, da es für die vorliegende Arbeit nicht von Bedeutung ist. Es sei aber auf weiterführende Literatur [Kunze, 2006; Langfeld, 2006] verwiesen.

Server-Technik

Auf der Serverseite wird neben einem Webserver (z. B. *Apache*[4]) ein spezieller Mapserver benötigt, der, ebenso wie die Geodaten(bank), nicht zwingend auf dem gleichen Rechner installiert sein muss [Klauer, 2002]. Ein Freier und sehr verbreiteter Mapserver ist der *UMN MapServer*[5]. Dieser kommuniziert über ein *Common Gateway Interface (CGI)* mit dem Webserver. Viele WebMapping-Anwendungen basieren auf dem *UMN MapServer* (vgl. Kapitel 3.1). *Geoserver*[6] und *deegree*[7] sind zwei weitere bedeutende Freie Mapserver.

2.3.4 Eigenschaften

Was zeichnet eine WebMapping-Anwendung aus? Die wichtigsten Komponenten und Interaktionsmöglichkeiten einer WebMapping-Anwendung werden an dieser Stelle vorgestellt. Dabei werden die Eigenschaften *allgemein* betrachtet. Eine spezielle Analyse ausgewählter WebMapping-Anwendungen wird in Abschnitt 3.1 durchgeführt.

Komponenten

Die Benutzeroberfläche von WebMapping-Anwendungen besteht in der Regel aus nachfolgenden Komponenten (Abbildung 2.3 zeigt einige von ihnen am Beispiel von *OpenLayers*):

[2] http://www.mozilla-europe.org [Abruf: 15.05.2007]
[3] http://www.w3.org/TR/SVG11 [Abruf: 15.05.2007]
[4] http://www.apache.org [Abruf: 15.05.2007]
[5] http://mapserver.gis.umn.edu [Abruf: 15.05.2007]
[6] http://docs.codehaus.org/display/GEOS/Home [Abruf: 15.05.2007]
[7] http://deegree.org [Abruf: 15.05.2007]

- *Hauptkarte*
  Die elementarste Komponente jeder Anwendung. Sie enthält eine oder mehrere überlagerte Raster- oder Vektorgrafiken; teilweise werden weitere Komponenten auf der Karte platziert.
- *Übersichtskarte*
  Stellt die Hauptkarte (in der Regel in ihrer ganzen geografischen Ausdehnung) in einer navigierbaren, kleineren Referenzkarte dar und markiert den aktuellen Hauptkartenausschnitt. Sie dient dem Nutzer zur besseren Orientierung und Übersicht.
- *Pan-Zoom-Navigationsleiste*
  Bietet ein Panning-Steuerkreuz zum Verschieben der Karte und eine Zoombar mit Slider zum Verändern der Zoomstufe.
- *Ebenenübersicht*
  Listet alle verfügbaren Kartenebenen (Layers) auf und bietet die Option diese einzeln ein- bzw. auszublenden.
- *Werkzeugleiste*
  Enthält alle Buttons und Menüpunkte zur Aktivierung verschiedener (GIS-)Funktionen (wie z. B. Navigations-, Analyse-, Abfrage- und Druckfunktionen).
- *Maßstab/-sbalken*
  Stellt den aktuellen Maßstab als Balkengrafik, Text- oder Auswahlfeld dar.
- *Legende*
  Erläutert die in der Karte verwendeten Farben und Symbole. Sie ist häufig mit der Ebenenübersicht kombiniert.

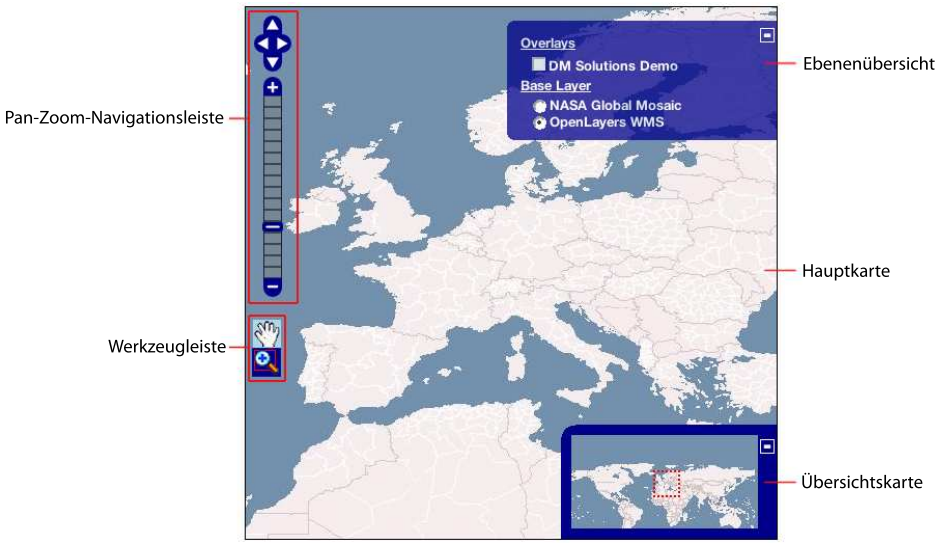


Abbildung 2.3: Komponenten der WebMapping-Anwendung *OpenLayers*

Interaktionsmöglichkeiten

Interaktivität ist mit der wachsenden Beliebtheit von WebMapping-Anwendungen und der rasanten Entwicklung des Internets in den 90er Jahren heute zu einem fundamentalen Bestandteil in der Entwicklung und Gebrauchstauglichkeit von Kartenanwendungen geworden [Ebinger u. Skupin, 2007].

Eine entscheidene Rolle spielt dabei die Navigation der Karte. Der Nutzer hat die Möglichkeit die Karte über die Maus per Drag&Drop stufenlos zu verschieben. In vordefinierten Schritten bewegt sich die Karte beim Drücken der Panning-Steuerkreuz-Buttons.
Das Verändern der Karten-Zoomstufe ist in der Regel durch folgende Interaktionen möglich: Klicken auf einen + bzw. – Button, Doppelklicken auf die Karte, Aufziehen einer Zoombox mit der Maus, Bewegen des Zoomsliders oder Benutzen des Mausrades.
Einige WebMapping-Anwendungen bieten zusätzlich eine Panning- und Zooming-Navigation über die Tastatur an (vgl. Kapitel 3.1).

Die Übersichtskarte bietet dem Nutzer oftmals die Möglichkeit, per Klick oder Drag&Drop, deren markierten Bereich - und damit die gesamte Hauptkarte - zu verschieben. Umgekehrt aktualisiert sich auch die Übersichtskarte sobald der Nutzer in der Hauptkarte navigiert.

Das Ein- und Ausblenden von Ebenen anhand von Checkboxen oder Radiobuttons sowie das Zusammenklappen von Unterebenen zu einer Hauptebene sind Interaktionsmöglichkeiten in der Ebenenübersicht.

Die Funktionen innerhalb der Werkzeugleiste stellen verschiedene Grade an Interaktivität bereit: Beispielsweise die erwähnten Pan/Zoom-Möglichkeiten sowie Funktionen zum Drucken, zur Sachdatenabfrage, Objektselektion, Entfernungsmessung oder Routenberechnung.

Eine aktuelle Cursor-Positionsangabe oder das individuelle Anpassen der Kartengröße sind zwei weitere interaktive Features in WebMapping-Anwendungen. Manche Map-Applikationen bieten dem Nutzer zusätzlich eine Suche über georeferenzierte Objekte oder weiterführende Sachdaten an. Die Kombination von Suchabfragen und der Umgang mit den Suchergebnissen ermöglicht dem Nutzer auch hier ein hohes Maß an Interaktivität.

Alle genannten Features tragen zum Interaktionsangebot einer WebMapping-Anwendung bei. Durch den zunehmenden Trend von Web 2.0 und AJAX sind auch im WebMapping-Bereich deutliche Verbesserungen in der Interaktivität spürbar. Einer der wohl größten Vorteile dabei ist die asynchrone Datenübertragung durch das *XMLHttpRequest*-Objekt, wodurch Webseiten im Hintergrund aktualisiert werden können und somit nicht für jede Änderung neu geladen werden müssen. Diese Technik ermöglicht es WebMapping-Anwendungen nach einer o. g. Benutzerinteraktion die gewünschte Funktion in einer vergleichsweise kurzen Wartezeit im Hintergrund durchzuführen. Dadurch entwickelt sich die Usability von interaktiven WebMapping-Anwendungen in die Richtung von klassischen Desktop(GIS-)Anwendungen [Dickmann, 2001; Kunze, 2006].

2.3.5 Kachelung

Die Kachelung (engl. *tiling*) ist ein recht junges Verfahren in der WebMapping-Entwicklung, um Karten in kleinere Kacheln zu unterteilen und damit die Usability von WebMapping-Anwendungen spürbar zu verbessern.

Im Folgenden wird anhand von Abbildung 2.4 das WebMapping-Tiling-Verfahren allgemein erläutert [Langfeld, 2006; Kunze, 2006]:

Um die WebMapping-Anwendung nur mit den wirklich notwendigen Daten zu belasten, werden zunächst alle Karten in Kacheln (engl. *tiles*) unterteilt (a). Jede Kachel ist für sich betrachtet eine eigene kleine Karte, die sich in Kombination mit den anderen zu einer Gesamtkarte fügt und optisch nicht mehr als einzelne Grafik wahrgenommen wird. Wird vom Anwender nun eine Region ausgewählt (b), müssen alle Kacheln geladen werden, die ganz oder teilweise in dieser Region sichtbar sind (c). Zusätzlich werden noch alle nicht sichtbaren Kacheln, die an diese Region angrenzen, geladen (d). Verschiebt der Anwender nun die Karte, werden die unter (d) bereits vorgeladenen Kacheln ohne Verzögerung angezeigt. Gleichzeitig werden alle neu angrenzenden Kacheln im Hintergrund geladen und alle nicht mehr angrenzenden Kacheln auf der anderen Seite entfernt.

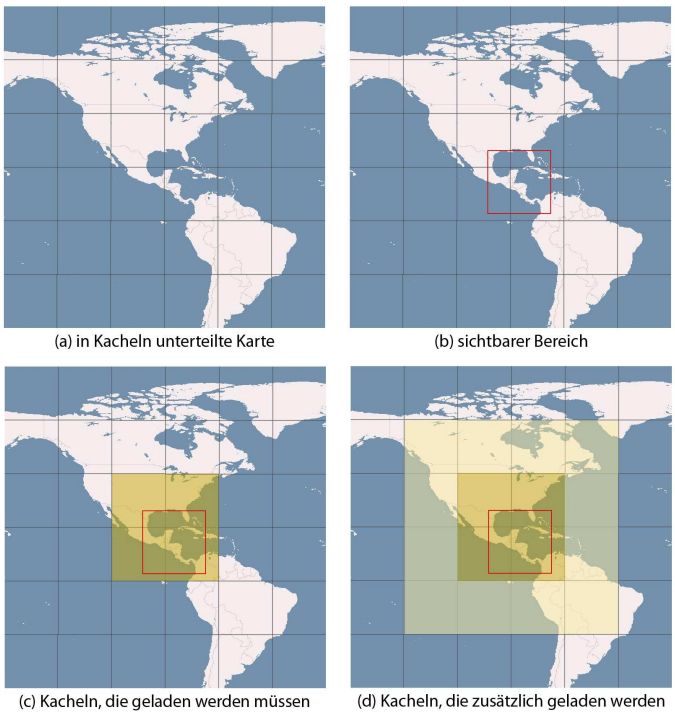


Abbildung 2.4: Bestimmen der zu ladenden Kacheln (vgl. [Langfeld, 2006; Kunze, 2006])

Tiling lässt sich in zwei Varianten realisieren: clientseitig und serverseitig. Beim clientseitigen Tiling definiert der WebMapping-Client die Kachelgrenzen und stellt für jede einzelne Kachel eine Anfrage an den Mapserver. Beim serverseitigen Tiling unterteilt ein Server die komplette Karte in mehrere kleinere Kacheln und leitet sie an den Webbrowser zur Darstellung weiter.

Alle geladenen Kacheln werden in der Regel von der WebMapping-Anwendung clientseitig verwaltet und in dem DOM-Baum, der die aktuelle HTML-Webseite repräsentiert, vorgehalten. Sobald sich der Kartenausschnitt ändert, wird die Kachelordnung im Programm neu berechnet, wonach überflüssige Kacheln aus den DOM-Baum entfernt und neue Kacheln in den DOM-Baum hinzugefügt werden.

Durch diesen dynamischen »Tile-Preloading-Prozess« entsteht zwar ein erhöhter Datentransfer, die Anwendung – und damit speziell das Navigieren in der Karte (das *Map Browsing*) – wird dadurch aber maßgeblich beschleunigt.

Für jede Zoomstufe existiert ein eigenes »Kachelgitter«. Die definierte Kachelgröße (in der Regel 256 Pixel) ist in allen Zoomstufen identisch. Zoomt der Anwender um eine Zoomstufe in die Karte hinein, verdoppeln sich die Breite und Höhe jeder Kachel. D. h. jede Kachel wird in vier neue Kacheln zerlegt, die dann detailliertere Informationen enthalten. Beim Herauszoomen ist es analog: aus vier Kacheln wird eine.

2.4 Open Geospatial Consortium

Das 1994 aus der *Open GRASS Foundation (OGF)* hervorgegangene *Open Geospatial Consortium (OGC)*[8] – bis zum Jahr 2004 unter dem Namen *OpenGIS Consortium* bekannt – ist ein internationales, gemeinnütziges Industriekonsortium mit aktuell mehr als 340 Mitgliedern (Stand: Mai 2007) aus Firmen, Regierungsorganisationen und Universitäten [OGC; Joos, 2003]. Ziel des OGC ist es, offene Standards in Form von Schnittstellen und Protokollen für einheitliche Zugriffsmethoden auf raumbezogene Informationen zu definieren und zum Zweck der Interoperabilität diese Spezifikationen frei zugänglich zu machen [Pichler u. Klopfer, 2005]. Alle Spezifikationen werden im Konsensprinzip erarbeitetet [Joos, 2003]. *OpenGIS®* ist registriertes Markenzeichen des Open Geospatial Consortiums [OGC].

OGC konforme Produkte und Dienste erlauben es dem Nutzer raumbezogene Informationen zwischen verteilten Systemen problemlos auszutauschen und zu nutzen. Das Interoperieren – auch konkurrierender Anwendungen – gibt dem Anwender außerdem die Freiheit, die für ihn und seine Anwendungsumgebung am Besten geeignete Anwendung auszuwählen [Pichler u. Klopfer, 2005].

[8] http://www.opengeospatial.org [Abruf: 15.05.2007]

2.5 Web Map Service

Ein *Web Service* ist ein über das Internet verfügbarer Dienst, der ein standardisiertes XML-basiertes Nachrichtenverfahren nutzt. Web Services sind integrale Bestandteile der Interoperatilität zwischen Software-Anwendungen, d. h. Web Services unterstützen durch eine standardisierte Schnittstelle den Austausch von mehreren, voneinander unabhängigen (Web-)Applikationen. Web Services sind unabhänig von Programmiersprachen und Betriebssystemen [Mitchell, 2005].

Web Services erfreuen sich auch in WebMapping-Anwendungen großer Beliebtheit und werden bereits als die »Zukunft des WebMapping« bezeichnet [Mitchell, 2005]. Die sogenannten *Web Map Services (WMS)* ermöglichen eine problemlose Integration von verteilt vorliegenden (gerasterten oder vektoriellen) Karten in WebMapping-Anwendungen über das Internet. Die Kartenerstellung wird dabei von einem *Web Map Server* (WMS-Server) übernommen [Erstling u. Simonis, 2005].

2.5.1 Web Map Client

Als *Web Map Client* bezeichnet man das System, welches den WMS integriert. Dabei spielt es keine Rolle, ob der Web Map Client den WMS serverseitig (z. B. in einem Mapserver) oder clientseitig (z. B. in einer WebMapping-Applikation) einbindet.

Bei einer **serverseitigen** WMS-Integration lassen sich einzelne Web Map Services kaskadieren, d. h. dass mehrere Web Map Services in einem einzelnen Service integriert werden können. Die dadurch entstehende Verkettung ist für den Endbenutzer nicht transparent und verhält sich wie ein einzelner WMS. Der Nachteil dieser WMS-Kaskadierung ist, dass der langsamste WMS die Reaktionszeit bestimmt, nach der die Karte im Browser dargestellt wird. Ein weiterer Nachteil: Sobald ein WMS-Dienst in dieser Kette ausfällt, bekommt der Web Map Client keine Antwort mehr und die Karte in der Anwendung bleibt leer [Erstling u. Simonis, 2005]. Der Mapserver übernimmt hierbei die Rolle des Web Map Clients.

Bei einer **clientseitigen** WMS-Integration fordert die verwendete WebMapping-Anwendung die Karten direkt von einem Web Map Server an und stellt diese im Browser dar. Bei Nutzung mehrerer Web Map Services kann jeder WMS in Form einer Ebene in der Anwendung präsentiert werden. Der Browser bzw. die WebMapping-Anwendung übernimmt hierbei die Rolle des Web Map Clients.

Werden nur externe Web Map Server genutzt, benötigt die WebMapping-Anwendung streng genommen keinen »eigenen« (im Sinne von »selbst aufgesetzten«) Mapserver, da die aufwändige Kartengenerierung auf den WMS-Server verlagert wird. Tatsächlich bauen aber etliche (Freie) WebMapping-Applikationen auf einen eigenen Mapserver auf (Näheres dazu im Kapitel 3.1).

2.5.2 OGC konformer WMS

Erfüllt ein Web Map Service die WMS-Spezifikationen[9] des OGC, so handelt es sich um einen OGC-konformen WMS. Dabei müssen die Anfrageoperationen an den WMS (requests) sowie die Antworten von dem WMS (responses) einen bestimmten Standard entsprechen. Die folgenden drei Anfrageoperationen sind derzeit von dem OGC spezifiziert, wobei die dritte optional ist [Erstling u. Simonis, 2005; Simonis u. Merten, 2004]:

1. *GetCapabilities*
   fragt die Fähigkeiten des WMS ab.
2. *GetMap*
   fordert eine konkrete Karte an.
3. *GetFeatureInfo*
   fragt Informationen zu einzelnen Kartenobjekten ab.

Abbildung 2.5 veranschaulicht in konzeptueller Weise, wie einige der OGC Web Services in Beziehung stehen und zeigt die wichtigsten, von ihnen definierten Operatoren.

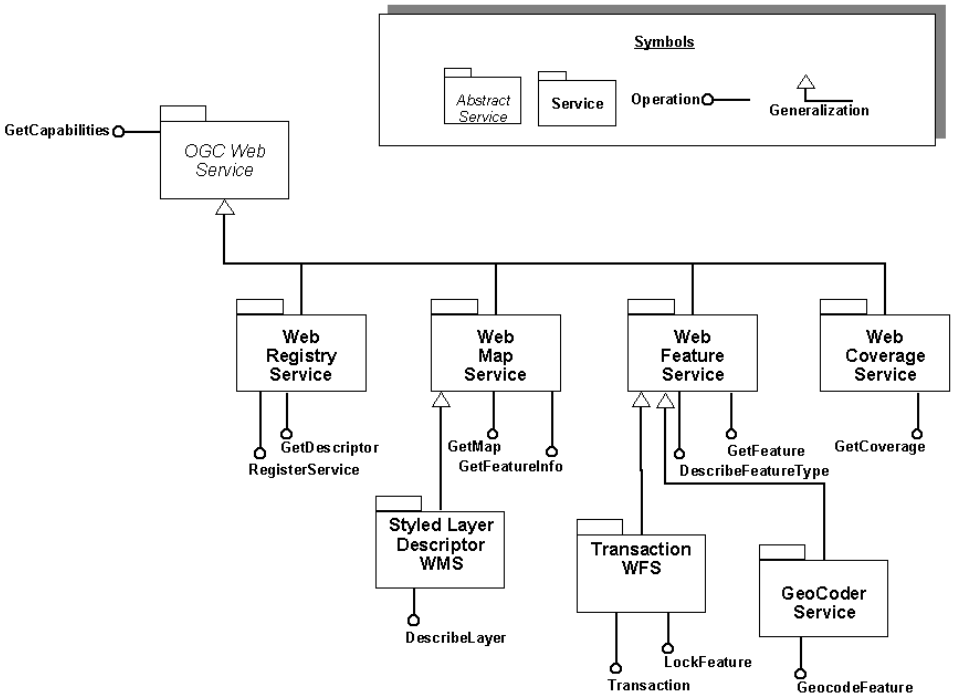


Abbildung 2.5: OGC Web Services Architektur (nach [OGC, 2002])

[9] http://www.opengeospatial.org/standards/wms [Abruf: 15.05.2007]

3 Analyse

In diesem Kapitel wird zunächst anhand einer Bestandsanalyse untersucht, welche Freien WebMapping-Anwendungen welche gebrauchstauglichen Features beherrschen. Ausgehend von den Untersuchungsergebnissen wird anschließend der Begriff *Smart Map Browsing* definiert und dessen Eigenschaften und Potenziale abgeleitet. Ein ausgewähltes Feature wird in Form einer Anforderungsanalyse, die Grundlage für die spätere Implementierung liefern. Die für diese Realisierung ausgewählte WebMapping-Anwendung wird am Ende des Kapitels im Detail analysiert.

3.1 Bestandsanalyse

3.1.1 Ziel

Das Ziel dieser Bestandsanalyse ist es, einen allgemeinen Überblick über die wichtigsten Freien WebMapping-Anwendungen zu geben und deren spezifischen, gebrauchstauglichen Navigationsmöglichkeiten herauszufiltern. Anhand der Ergebnisse sollen die Eigenschaften (und deren Potenziale) von *Smart Map Browsing* abgeleitet werden können.

3.1.2 Methode

Die Kriterien

Zu Beginn werden einheitliche Kriterien definiert, nach denen die Analyse durchgeführt wird. Die Kriterien gliedern sich in folgende acht Kategorien:

1. *Allgemein*
   Name, URLs (Home, Dokumentation, Download, Live-Demo), aktuelle Version, letztes Update, Lizenz, Unternehmen/Organisation (die an der Entwicklung maßgeblich beteiligt waren/sind), Kurzbeschreibung
2. *System*
   Architektur (clientseitig oder client-serverseitig), Programmiersprache, Voraussetzungen, unterstützte Browser, ggf. Integration anderer Freier Software

3. *Community*
   URL der Mailinglisten (ML), Entwickler- und Anwender-ML (Angaben zu Mails je Monat und Gesamtzahl der aktiven Entwickler/Anwender – bezogen auf einen festen Zeitraum), Versionsverwaltung, kommerzieller Support
4. *Dokumentation*
   zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URLs)
5. *Usability*
   Gesamteindruck, Hauptkarte, Übersichtskarte, Ebenenübersicht, Legende, Maßstab(-sbalken), Werkzeugleiste, Zoomnavigationsleiste, Pannavigationssteuerkreuz, Zooming allgemein, Zooming per Doppelklick/Mausrad/Zoombox, Panning allgemein, Zooming & Panning per Tastatur, Tiling
6. *Weitere Features*
   Analyse-, Such-, Hilfe-, Druckfunktionen
7. *Bemerkungen*
8. *Screenshot*
   von der untersuchten Demo

Die Kategorien 1 und 2 geben allgemeine Informationen über die WebMapping-Anwendung. Kategorie 3 beleuchtet die jeweilige Community und untersucht deren Aktivität. Der Punkt 4 gibt Hinweise, kurze Bewertungen und weiterführende Informationen zu verfügbaren Dokumentationen. Der Hauptbestandteil der Untersuchung ist die Usabilityanalyse der Anwendung (Kategorie 5). Hierbei werden die einzelnen GUI-Komponenten auf Interaktionsmöglichkeiten und Gebrauchstauglichkeit geprüft und bewertet. Dies geschieht auf Basis einer vorher ausgewählten Demo. In Kategorie 6 werden zusätzliche interaktive (GIS-)Funktionen untersucht, die für die Usability der Anwendung einer eher untergeordnete Rolle spielen. Bemerkungen und Besonderheiten werden in der Kategorie 7 gelistet. Zur Veranschaulichung der getesteten Demo dient der unter (8) erfasste Screenshot.

Die Untersuchung soll *keine* vollständige Feature-Auflistung jeder Anwendung darstellen. Spezielle technische Details (beispielsweise die Unterstützung von WMS oder WFS) spielen bei dieser Analyse keine Rolle und werden daher vernachlässigt. Im Mittelpunkt der Untersuchung steht die Usability der Anwendung.

Die Mailinglisten-Analyse

Die Aktivität einer FS-Community spiegelt sich maßgeblich in den Mailinglisten (MLn) wider. In der Regel haben Freie WebMapping-Anwendungen eine Entwickler- und eine Anwender-ML. Um die Aktivität der Listen vergleichen zu können, werden zwei Zahlen für jede Liste bestimmt:

- Die durchschnittliche Anzahl der Mails pro Monat.
- Die Gesamtanzahl aller Entwickler bzw. Anwender, die sich auf der Liste in einem bestimmten Zeitabschnitt zu Wort gemeldet haben. Anmerkung: Unter Entwickler werden in diesem Fall auch die Personen verstanden, die an der Entwicklung interessiert sind, selber aber nicht aktiv dazu beitragen.

Als Zeitraum seien sechs Monate definiert (Oktober 2006 bis März 2007). Grundlage für diese Bestimmung sind die in Monaten strukturierten ML-Archive.

Für eine automatische Bestimmung der gesuchten Zahlen wird ein Skript in Python[1] geschrieben, das die Zählung übernimmt (vgl. Listing 3.1). Voraussetzung dafür ist, dass in einem separaten Verzeichnis die monatlichen ML-Archive (entpackt) vorliegen. Alle Mails eines Monats befinden sich in *einer* Datei (unformatiertes Textformat). Andernfalls kann keine korrekte automatische Analyse erfolgen.
Für die Untersuchung werden zwei beliebte Typen von ML-Archiven (mit ihrer unterschiedlichen Syntax bei der Angabe des Absenders) unterstützt: das sehr weit verbreitete *GNU Mailman*[2]-Archiv (gepackte txt-Dateien; Syntax »`From [name] at [domain]`«) und das ML-Archiv von *SourceForge*[3] (HTML-Format; Syntax »`From: [real name] <name@do...>`«).

Das Skript wird auf das Verzeichnisses mit den einzelnen Archivdateien angewendet. Hier die wichtigsten Schritte, die dabei durchlaufen werden:

1. Datei einlesen (Zeile 11)
2. Zeile mit String »`From` « oder »`From:` « finden (13;23)
3. Mail-Adresse des Absenders auslesen (14-17; 24-27)
4. Mail-Zählvariable hochsetzen (18; 28)
5. User-Zählvariable hochsetzen, sofern Adresse noch nicht vorhanden (19-20; 29-30)
6. Schritte 2 - 5 für alle restlichen Zeilen der Datei wiederholen
7. Analog alle übrigen Dateien nach Schritten 1 - 6 auslesen
8. Abschließende Ausgabe der berechneten Werte (34-38)

[1] http://www.python.org [Abruf: 15.05.2007]
[2] http://www.gnu.org/software/mailman [Abruf: 15.05.2007]
[3] http://sourceforge.net [Abruf: 15.05.2007]

```
import os
def walker(arg,dir,path):
    months=0
    mails=0
    users=[]

    for file in path:
        i=file.find(".",0,1)
        if i==-1:
            months=months+1
            text=open(file).readlines()
            for line in text:
                if line.find("From " )>-1:
                    fields=line.split(" ")
                    try:
                        if fields[2]=="at":
                            user=fields[1]+"@"+fields[3]
                            mails=mails+1
                            if users.count(user)==0:
                                users.append(user)
                    except IndexError:
                        pass
                if line.find("From: " )>-1:
                    fields=line.split(" ")
                    try:
                         if line.find("...>" )>-1:
                            user=fields[1]
                            mails=mails+1
                            if users.count(user)==0:
                                users.append(user)
                    except IndexError:
                        pass
            print file+": %s "%mails +"mails"
    print "---"
    print "months      : "+str(months)
    print "mails total: "+str(mails)
    print "mails/month: "+str(mails/months)
    print "users total: "+str(len(users))
os.path.walk(".",walker,None)
```

Listing 3.1: Pythonscript zum Analysieren von ML-Archiven; *MLanalyse.py*

Beispielausgabe von der Analyse der Entwickler-ML von *OpenLayers*:

```
2006-December.txt: 36 mails
2006-November.txt: 131 mails
2006-October.txt: 196 mails
2007-January.txt: 262 mails
2007-February.txt: 342 mails
2007-March.txt: 510 mails
---
months      : 6
mails total: 510
mails/month: 85
users total: 72
```

Die Live-Demo

Anhand einer Beispiel-Demo werden die aufgestellten Usabilitykriterien (Kategorie 5) und die weiteren definierten Features (Kategorie 6) untersucht. Die Demo soll ohne Installation der Anwendung und vorinstallierten PlugIns im Firefox-Browser nutzbar sein. Eine JavaScript-Aktivierung wird vorausgesetzt. Eine solche Live-Demo ist in der Regel auf der Homepage der jeweiligen WebMapping-Anwendung zu finden.

Zu berücksichtigen ist, dass die Funktionalität der Demo nicht zwingend den vollen Funktionsumfang der Anwendung widerspiegelt. Auch die Benutzeroberfläche der Demo stellt nur eine *mögliche* GUI-Gestaltung der Anwendung dar. Aus diesem Grund wird deutlich darauf hingewiesen, dass die Analyse sich ausschließlich auf die verwendete Demo beschränkt.

Ein weiterer erwähnenswerter Punkt ist die Objektivität der Usabilityuntersuchung. Die Beurteilung von Gebrauchstauglichkeit und Gestaltung der Anwendungen lässt sich nicht immer vollständig von subjektiven Empfindungen trennen (vgl. Abschnitt 2.2.2).
Viele Interaktionsmöglichkeiten in (Web-)Anwendungen haben sich bereits zu Standards entwickelt und decken sich größtenteils mit der Erwartungshaltung des Nutzers. Bezogen auf WebMapping-Anwendungen gehören dazu beispielsweise das Drag&Drop-Verhalten beim Verschieben der Karte oder die selbsterklärenden Icons zum Zoomen oder Verschieben der Karte. Solche Usabilitykriterien lassen sich nahezu objektiv untersuchen.
Die Gestaltung der Anwendungen – z. B. die Verwendung von Farben oder die Anordnung von Komponenten – ermöglichen jedoch einen gewissen Spielraum bei der Usabilitybeurteilung. Bei der nachfolgenden Analyse soll dieser subjektive Anteil so gering wie möglich gehalten werden.

3.1.3 Auswahl

Für die Auswahl der zu untersuchenden Freien WebMapping-Anwendungen wird das Internet-Portal *FreeGIS*[4] genutzt. Unter dem Schlagwort »Web Mapping« in der Rubrik »Software« sind mehr als 50 Software-Tools gelistet (Stand: 10.4.2007). Alle zu analysieren wäre für diese Arbeit eindeutig zu umfangreich. Eine Einschränkung ist demnach nötig.

Zunächst konnten nach der WebMapping-Begriffsdefinition (vgl. Kapitel 2.3.1) einige Einträge gestrichen werden: Beispielsweise zählen *MapServer*, *Geoserver* und *deegree* nicht zu den gesuchten WebMapping-Clients.

Nach Sichtung und Abwägung der restlichen Einträge wurden schließlich elf Freie Webmapping-Client-Applikationen ausgewählt, die am vielversprechendsten und in ihrer Erwäh-

[4] http://freegis.org [Abruf: 15.05.2007]

nung auf anderen einschlägigen GIS-Portalen (wie *OSGeo*[5] oder *MapTools*[6]) am bedeutendsten erschienen.

Demnach erhebt diese Bestandsanalyse keinen Anspruch auf Vollständigkeit. Es soll vielmehr ein Überblick über die Vielfältigkeit der Freien WebMapping-Anwendungen und ihren gebrauchstauglichen Funktionen gegeben werden.

Aufgrund der großen Bedeutung und Beliebtheit von *Google Maps* im WebMapping-Bereich [Ramsey, 2006] wird zusätzlich diese proprietäre Lösung als Vergleich zu den elf Freien Anwendungen hinzugezogen.

Abschließend folgen alle zwölf ausgewählten WebMapping-Anwendungen (in alphabetischer Reihenfolge):

- CartoWeb
- Chameleon
- Google Maps
- iGeoPortal
- ka-Map
- Mapbender
- Mapbuilder
- MapGuide Open Source
- MappingWidgets
- OpenLayers
- p.mapper
- WMS Mapper

[5] http://www.osgeo.org [Abruf: 15.05.2007]
[6] http://www.maptools.org [Abruf: 15.05.2007]

3.1.4 Ergebnis

Die vollständigen Analyseergebnisse von allen zwölf untersuchten WebMapping-Anwendungen befinden sich im Anhang A (S. 89ff.) – jedes Tool umfasst zwei Seiten.

Die Anwendungen lassen sich in folgende drei Typen klassifizieren:

Typ 1: Frei und 100% clientseitig

Typ 2: Frei und client-serverseitig

Typ 3: Proprietär und 100% clientseitig

2 von 12 Anwendungen sind Freie Software und *reine* JavaScript Anwendungen, die komplett auf dem Client laufen (Typ 1). D. h. sie kontaktieren einen externen Mapserver lediglich zur Kartenanforderung; die Funktionalitäten der Anwendung sind zu 100% auf dem Client implementiert. Ein Mapserver (z. B. ein WMS-Server) wird zwar benötigt, gehört aber in diesen Fällen nicht zur eigentlichen WebMapping-Anwendung. In dieser Arbeit wird von *clientseitig* gesprochen, wenn diese Merkmale erfüllt sind.
9 der 12 untersuchten Anwendungen sind Freie Software und bilden eine client-serverseitige Anwendung (Typ 2). D. h. die Anwendung impliziert in der Regel einen eigenen Mapserver bzw. setzt eine solchen voraus. Client-Server-Mix-Anwendungen erfordern (im Gegensatz zu den clientseitigen) eine Installation. Für diese Kategorie von WebMapping-Anwendungen wird in dieser Arbeit der Begriff *client-serverseitig* verwendet.
Google Maps ist die einzige proprietäre Applikation im Test und ist, wie Typ 1, eine reine JavaScript-WebMapping-Anwendung. Wie im Abschnitt 3.1.3 erwähnt, konzentriert sich die Analyse auf *Freie* WebMapping-Lösungen. *Google Maps* wurde jedoch aufgrund der Bedeutsamkeit als Vergleich in die Untersuchung mit aufgenommen. Proprietäre Client-Server-Lösungen werden in dieser Analyse nicht berücksichtigt.

Die Tabelle 3.1.4 stellt die markantesten Analyseergebnisse gegenüber und bietet die Möglichkeit, die untersuchten Applikationen zu vergleichen.

Tabelle 3.1: Aufstellung der Analyseergebnisse der WebMapping-Anwendungen

	Frei & 100% clientseitig		Frei & client-serverseitig									proprietär& 100%clients.
	OpenLayers S.90f	**WMS Mapper** S.92f	**CartoWeb** S.95f	**Chameleon** S.97f	**iGeoPortal** S.99f	**ka-Map** S.101f	**Mapbender** S.103f	**Mapbuilder** S.105f	**MapGuide Open Source** 107f	**Mapping Widgets** 109f	**p.mapper** S.111f	**Google Maps** S.114f
Version	2.3	0.03	3.3.0	2.4.1	1.2.1	1.0	2.4.1	1.0.1	1.1.0	0.3.1	3.0.1	2.79
letztes Update	21.02.07	k.A.	31.08.06	06.09.06	15.09.05	05.02.07	23.03.07	19.07.06	09.12.06	17.03.06	30.12.06	18.04.07
FS-Lizenz	BSD	AFL	GNU GPL	X11	GNU GPL	MIT	GNU GPL	GNU LGPL	GNU LGPL	GNU GPL	GNU GPL	-
Revisionsverwaltung	SVN	-	CVS	CVS	SVN	CVS	SVN	SVN	SVN	SVN	SVN	-
Entwickler-ML	✓	-	✓	-	✓[3]	✓	✓	✓	✓	-	-	-
Mails je Monat[1]	85		16		49	11	55	120	168			-
aktive Entwickler[2]	72		12		101	7	46	47	42			-
Anwender-ML	✓	-	✓	✓	✓[3]	✓	✓	✓	✓	-	✓	✓
Mails je Monat[1]	197		80	51	92	96	107	65	527		82	-[4]
aktive Anwender[2]	150		86	63	106	121	98	78	297		61	-[4]
Übersichtskarte	✓	-	✓	✓	✓	✓	✓	-	-	-	✓	✓
Ebenenübersicht	✓	-	✓	✓	-	✓	✓	✓	✓	-	✓	✓
Legende	-	-	✓[5]	✓	✓	✓[5]	✓	-	✓[5]	✓	✓	-
Maßstab/-sbalken	✓/ -	- / -	✓/ ✓	- / ✓	- / -	✓/ ✓	✓/ ✓	✓/ -	✓/ -	-	✓/ ✓	- / ✓
Zoomnavigationsleiste	✓	-	-	-	-	-	-	-	✓	-	✓	✓
Pannavigation												
am Kartenrand	-	-	✓	✓	✓	-	✓	-	-	-	-	-
in PanZoomBar	✓	-	-	-	-	-	-	-	✓	-	-	✓
Zooming per												
Doppelklick	✓	-	-	-	-	-	-	-	-	-	-	✓
Mausrad	✓	-	-	-	-	✓	-	-	-	-	✓	✓
Zoombox (Shiftkey)	✓(✓)	-	✓(✓)	✓(-)	✓(-)	✓(-)	✓(-)	✓(✓)	✓(✓)	✓(-)	✓(✓)	-
Panning per												
Übersichtskarte	✓	-	✓	✓	✓	✓	✓	-	-	-	✓	✓
Zooming & panning												
per Tastatur	✓	-	-	-	-	-	-	-	-	-	✓	✓
Tiling	✓	✓	-	-	-	✓	-	-	✓	-	-	✓

[1] durchschnittliche monatliche Mailanzahl im Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[3] nutzt allgemeine User- bzw. Developer-ML von *deegree*
[4] Automatische Analyse in *Google Groups* nicht möglich.
[5] Legende ist in Ebenenübersicht integriert

Stand: 15.05.2007

3.1.5 Auswertung

Die Analyseergebnisse aus Abschnitt 3.1.4 werden nun verglichen und unter dem Gesichtspunkt der Usability ausgewertet. Die Auswertung bietet einen ersten genauen Überblick über beispielhafte *Smart Map Browsing* Fähigkeiten von WebMapping-Anwendungen und stellt die Grundlage für die *Smart Map Browsing* Begriffsdefinition im Abschnitt 3.2.

Die Auswertung wird in folgende drei Abschnitte gegliedert: GUI-Komponenten, Pan-Zoom-Verhalten und Community.

A) GUI-Komponenten

Übersichtskarte 8 von 12 Anwendungen haben eine Übersichtskarte, die (per Klick oder Doppelklick) zum Verschieben der Hauptkarte benutzt werden kann. *Mapbender* bietet die Möglichkeit einen Bereich aufzuziehen, wodurch eine Kombination von Zooming und Panning möglich ist. *Google Maps* bieten ein animiertes Panning von Hauptkarte und Übersichtskarte nach Doppelklick oder Drag&Drop des Ausschnitts. *Ka-Map* zeigt bereits ähnliche Animationsansätze.
Zoomt der Anwender in die Karte, passt sich der markierte Ausschnitt der Übersichtskarte an. *Google Maps* und *OpenLayers* zeigen dabei eine dynamische, mitzoomende Übersichtskarte (bei den anderen Tools bleibt sie statisch). Eine Zentrierung des Ausschnittes nach manuellem Verschieben gelingt aber nur *Google Maps*.
Google Maps zeigt die Referenzkarte stets im gleichen Stil wie die Hauptkarte; bei *OpenLayers* hingegen wird der Stil nur von der aktiven Basisebene definiert; Overlays werden nicht dargestellt. Ebenfalls nur von *Google Maps* und *OpenLayers* wird eine minimierbare Übersichtskarte angeboten.

Fazit: Nahezu perfekte Umsetzung in *Google Maps*; gute Ansätze und ähnliches Potenzial in *OpenLayers*. Statische Übersichtskarten der anderen Tools sind bei hoher Zoomtiefe nicht sehr hilfreich.

Ebenenübersicht 9 von 12 Anwendungen nutzen eine Ebenenübersicht. Davon nutzen 3 sie in Kombination mit einer Legende. *OpenLayers* und *Mapbender* können die Übersicht komplett minimieren. Andere bieten zusammenklappbare (Unter-)Ebenen. Hierbei besteht jedoch die Gefahr der Verschachtelung (siehe *CartoWeb*, *ka-Map*). Andernfalls, bei zu langen Ebenenlisten, könnte der Benutzer leicht die Übersicht verlieren (vgl. *Chameleon*). Eine Ebenenauswahl, die der Nutzer erst mit einem zusätzlichen Klick auf einen Aktualisierungs-Button bestätigen muss, wirkt sich schlecht auf die Usability aus (vgl. *CartoWeb*, *Chameleon*, *iGeoPortal*). *Ka-Map* und *iGeoPortal* ermöglichen es einzelne Ebenen zu selektieren und in ihrer Reihenfolge zu ändern. *Google Maps* stellt die Ebenen lediglich durch 3 Buttons (Karte, Satellit, Hybrid) dar.

Fazit: Gute Kombination mit der Legende möglich; teilweise Verschachtelungsgefahr; manuelle Aktualisierung nachteilig; überzeugende Schlichtheit von *Google Maps* und *OpenLayers*.

Maßstab Die Darstellung des Maßstabs ist sehr unterschiedlich gelöst: Teilweise als Maßstabsbalken (u. a. *ka-Map*), als einfache Maßstabszahlanzeige (*OpenLayers*), als vordefinierte Auswahlbox (*CartoWeb*, *Mapbender*) oder als editierbares Maßstabstextfeld (*Mapbuilder*, *p.mapper*).

Fazit: Auswahlbox und Textfeld wirken komplizierter als Balken oder schlichte Maßstabsanzeige. Ein halbtransparenter Balken fügt sich gut in die Karte ein.

Werkzeugleiste Alle analysierten Anwendungen haben eine Werkzeugleiste; von sehr minimalistisch (*OpenLayers*, *WMS Mapper*) bis sehr umfangreich (*ka-Map*, *Mapbender*). Vom Funktionsumfang unterscheiden sie sich nur geringfügig. Auffallend sind die Zoom-History-Buttons (u. a. bei *Mapbender* und *MapGuide Open Source*). In den meisten Anwendungen fehlt eine Anpassung des Mauscursors an das gewählte Werkzeug.

Fazit: In der Regel haben die Werkzeugleisten »übliche« Funktionen ohne große Unterschiede. Die 100%-clientseitigen Anwendungen beschränken sich auf Basisfunktionalitäten.

Zoomnavigationsleiste 4 von 12 Anwendungen bieten eine Zoombar (*OpenLayers*, *MapGuide Open Source*, *p.mapper* und *Google Maps*). Bei *MapGuide Open Source* ist sie frei auf der Karte platzierbar. *P.mapper* bietet ein bemerkenswertes *continuous zooming* Feature: Beim Bewegen des Sliders wird die Karte zeitgleich skaliert; nach dem Loslassen wird sie neu gezeichnet. Irritierend wirkt, dass die + und – »Buttons« an den Zoombar-Enden nicht funktionieren.

Fazit: Zoomnavigationsleisten ermöglichen dem Anwender eine gute Zoomtiefenorientierung. Alle 4 Anwendungen bieten einen auffallend hohen Grad an Gebrauchstauglichkeit.

Pannavigationssteuerkreuz In 4 Anwendungen sind im Kartenrand Buttons zur Pan-Navigation integriert. 3 Anwendungen bieten hingegen ein Steuerkreuz innerhalb einer zusammenhängenden Pan-Zoom-Bar-Navigation (siehe *OpenLayers*, *MapGuide Open Source* und *Google Maps*). In den restlichen 5 Anwendungen kann der Benutzer lediglich per Drag&Drop die Karte verschieben.

Fazit: Eine Pannavigation ist nützlich. Durch eine Kartenrandnavigation ist oftmals keine optisch-nahtlose Integration der Karte in eine Website möglich. Eine PanZoom-Bar bietet kürzere »Mauswege« beim Navigieren.

B) Pan- und Zoom-Verhalten

Zooming per Doppelklick Lediglich *OpenLayers* und *Google Maps* bieten dem Nutzer die Möglichkeit per Doppelklick in die Karte hineinzuzoomen. Anwendungen, bei denen erst auf das *ZoomIn*-Werkzeug umgeschaltet werden muss bevor per Klick hineingezoomt werden kann, finden hier keine Berücksichtigung.

Fazit: Dieses gebrauchstaugliche Feature zeigt, dass ein problemloser Wechsel zwischen panning und zooming möglich ist *ohne* ein neues Werkzeug auswählen zu müssen. Ein flüssiges *Map Browsing* ist möglich.

Zooming per Mausrad 3 Anwendungen erlauben es per Mausrad die Zoomstufe zu ändern (*OpenLayers*, *Google Maps*, *p.mapper*). Nur *Google Maps* behält dabei die geografische Kartenposition unter dem Mauszeiger nach dem Mausradbetätigen an der gleichen Stelle wie vor dem Zoomen. Im Vergleich dazu: *OpenLayers* zentriert die Karte auf die Stelle der Mausposition; *p.mapper* zoomt ohne Berücksichtigung des Mauscursors.

Fazit: Ebenso wie beim Doppelklick-Feature ist ein schneller Wechsel zwischen Verschieben und Zoomen der Karte möglich. Nur *Google Maps* bietet dabei ein erwartungskonformes Verhalten im Umgang mit der Mausposition.

Zooming per Zoombox 10 von 12 Anwendungen bieten die Möglichkeit über eine mit der Maus aufgezogene Box in die Karte hineinzuzoomen. Nur *WMS Mapper* und *Google Maps* bieten diese Funktion nicht. Dabei fällt eine farbige Unterlegung des Zoombox-Bereichs positiv auf (u. a. *OpenLayers*, *ka-Map*). Bei 5 Anwendungen ist die Zoombox-Benutzung zusätzlich über die gedrückte Shift-Taste möglich (u. a. *OpenLayers*, *mapbuilder*).

Fazit: Das Feature gehört zur »Grundausstattung« (überraschend, dass *Google Maps* dies nicht anbietet) und ist sehr nützlich für direktes Zoomen auf einen bestimmten Bereich; Shifttasten-Gebrauch verbreitet.

Zooming/Panning per Tastatur Bei *OpenLayers*, *p.mapper* und *Google Maps* ist eine Zoom-Pan-Steuerung über die Tastatur möglich. *Google Maps* Panning-Tastenbelegungen sind sehr intuitiv und umfangreich.

Fazit: Tastatursteuerung wird nur von wenigen Anwendungen unterstützt. Erweiterte Panning-Tastenbelegungen sind sehr komfortabel.

Zoomstufen Ein wichtiger Usabilityaspekt ist die Orientierung des Nutzers, in welcher Zoomstufe er sich gerade befindet.
Bei 4 Anwendungen gibt es endlose Min-/Max-Zoomstufen; d. h. keine Beschränkung des Zoomlevelintervalls (siehe *Chameleon*, *iGeoPortal*, *Mapbuilder* und *Mapping-Widgets*). 2 Anwendungen (*WMS Mapper*, *CartoWeb*) definieren zwar die Min-/Max-Zoomstufe, ermöglichen aber dennoch ein Zoomen »auf der Stelle«. Die restlichen 6 Anwendungen der Analyse fangen das Min-/Max-Zoomverhalten erfolgreich ab. Insbesondere die Anwendungen mit einer Zoomnavigationsleiste bieten eine ideale Zoomtiefenorientierung.

Fazit: Eine gute Zoomtiefenorientierung wirkt sich positiv auf die Usability aus; endlose Zoomstufen hingegen nachteilig.

Zoom-Reset Außer *WMS Mapper* bieten alle untersuchten Anwendungen eine Option, die Karte auf ihre maximale Ausdehnung zurückzusetzen. In der Regel als Button in der Werkzeugleiste realisiert. *Google Maps* integriert die Funktion gut in die Pan-Zoom-Navigationsleiste. *OpenLayers* bietet standardmäßig keine Möglichkeit die Zoom-Reset-Funktion mit der erweiterten Zoomnavigationsleiste zu kombinieren.

Fazit: Eine Standardfunktion, die vom Nutzer erwartet wird und in keiner Anwendung fehlen sollte.

Tiling Eine Kachelung der Karte wird von 5 Anwendungen unterstützt (*OpenLayers*, *WMS Mapper*, *ka-Map*, *MapGuide Open Source* und *Google Maps*). Ein spürbar verzögerungsfreies Panning bieten nur *OpenLayers* und *Google Maps*. Die anderen sind in ihrem Panning-Verhalten etwas schwerfällig.

Fazit: Spürbar gutes Pan-Zoom-Verhalten bei den kachelbasierten Anwendungen. Überragendes Panning bei *OpenLayers* und *Google Maps*.

Ladezeit Die Ladezeit beim Verschieben und Zoomen der Karte ist ein entscheidener Faktor bei der Bewertung der Usability.
Beim Panning haben die kachelbasierten Anwendungen einen klaren Geschwindigkeitsvorteil im Vergleich zu herkömmlichen Kartenanwendungen.
Beim Zooming lassen sich 3 unterschiedliche Formen des Kartenaufbaus erkennen:

- Karte wird durch nacheinander folgende Kacheln aufgebaut (*OpenLayers*, *ka-Map*, *MapGuide Open Source*, *Google Maps*)
- neue Karte erscheint komplett (u. a. *Mapbender*, *Mapbuilder*)
- neue Karte erscheint komplett; Ladezeit wird mit »loading«-Meldung überbrückt (*iGeoPortal*, *p.mapper*, *CartoWeb*)

Fazit: Die Ladezeiten fordern teilweise viel Geduld; besonders beim Panning sehr störend. Beim Zoomwechsel wird oftmals ein leerer Kartenhintergrund sichtbar. Ein allmählicher Kachelaufbau sorgt für Dynamik und sichtbaren Fortschritt während des Ladens.

C) Comunity

Revisionsverwaltung 7 Anwendungen bieten eine Quellcodeverwaltung mit SVN[7], 3 eine mit CVS[8]. *WMS Mapper* und *Google Maps* stellen keine Revisionsverwaltung bereit.

Fazit: Eine Revisionsverwaltung ist Standard in jedem FS-Projekt. Die Verteilung zu Gunsten des moderneren SVN ist nicht überraschend.

Entwickler-Mailingliste 7 von 12 Anwendungen stellen eine Entwickler-ML zur Verfügung. Wertet man die Zahlen der Mails pro Monat aus, fallen *MapGuide Open Source* (168), *Mapbuilder* (120) und *OpenLayers* (85) auf. Die restlichen bewegen sich zwischen 10 und 50 Mails im Monat.
Bei der Anzahl von aktiven Entwickler auf der ML im untersuchten 6-Monatszeitraum führt *iGeoPortal* mit 101 (anzumerken ist, dass *iGeoPortal* die *deegree*-ML nutzt). *OpenLayers* liegt mit 72 unterschiedlichen Entwicklern auf Platz 2. Die anderen Anwendungen bewegen sich etwa zwischen 10 und 50 Entwicklern.

Fazit: *MapGuide Open Source* überrascht mit hohem monatlichen Mailaufkommen bei verhältnismäßig wenigen Entwicklern. *OpenLayers* zeigt ein überdurchschnittliches Mailaufkommen und die höchste Anzahl an aktiven Entwicklern eines Einzelprojektes.

Anwender-Mailingliste 10 von 12 Applikationen nutzen eine Anwender-ML. *MapGuide Open Source* sticht mit 527 Mails pro Monat starkt heraus. Diese Zahl lässt vermuten, dass die Software durch den früheren proprietären Hintergrund von Autodesk stark verbreitet wurde. *OpenLayers* steht auch auf der Anwender-ML mit 197 Mails im Monat weit vorn. Der Großteil bewegt sich im Bereich 50 bis 100 Mails.
Auch bei den aktiven Anwendern liegt *MapGuide Open Source* klar vorn (297); gefolgt von *OpenLayers* mit 150. Die restlichen Mailverteiler liegen zwischen 60 und 120 aktiven Anwendern in den 6 Monaten.

Fazit: Auffallend hohe monatliche Mailanzahl bei *MapGuide Open Source*; Spitzenposition auch bei den aktiven Anwendern. Bemerkenswert: *OpenLayers* Anzahl von Mails und Anwendern ist (wie auch auf der Entwickler-ML) überdurchschnittlich hoch.

[7] http://subversion.tigris.org [Abruf: 15.05.2007]
[8] http://www.nongnu.org/cvs [Abruf: 15.05.2007]

3.2 *Smart Map Browsing*

Auf Grundlage der vorangegangenen Analyseauswertung wird in diesem Abschnitt eine Definition des Begriffs *Smart Map Browsing* vorgenommen, dessen aktuelle Eigenschaften abgeleitet und die Potenziale herausgearbeitet. Der Ausdruck *Smart Map Browsing* wurde bisher in keiner Literatur gefunden und wird in der vorliegenden Arbeit geprägt.

3.2.1 Begriff

Der Begriff *Smart Map Browsing* wird auf Basis der Usability-ISO-Norm (vgl. Abschnitt 2.2.1) definiert:

> *Smart Map Browsing beschreibt eine für den Benutzer effektive, effiziente und zufriedenstellende Gebrauchstauglichkeit von WebMapping-Anwendungen.*
>
> *Die Eigenschaften von Smart Map Browsing definieren eine Vielzahl von selbsterklärenden, interaktiven Elementen und Funktionen, die erwartungskonforme Interaktionsverhalten aufweisen und eine zum aktuellen Stand der Technik ideale und gebrauchstaugliche WebMapping-Anwendung kennzeichnen.*

3.2.2 Eigenschaften

Der aktuelle Stand der Technik von *Smart Map Browsing* lässt sich anhand von zwei Kategorien – *GUI-Komponenten* und *Pan-Zoom-Verhalten* – im Detail beschreiben. Es sei angemerkt, dass es sich bei den Eigenschaften um den *derzeitigen* Stand handelt (gewissermaßen *Smart Map Browsing 1.0*) und mittelfristig Aktualisierungen aufgrund technologischer Neuerungen erforderlich werden.

A) GUI-Komponenten

Hauptkarte Bei reinen Kartenanwendungen, wo die Karte im Mittelpunkt der Aufmerksamkeit steht und kein nützliches »Beiwerk« ist, sollte sie den Großteil der Anwendungsfläche ausfüllen. Eine stufenlos anpassbare Kartengröße (z. B. durch Verschieben der Kartenränder) ermöglicht dem Benutzer interaktiv seine individuelle Einstellung zu finden. Zusätzlich wirkt sich eine dynamische Anpassung der Karte an die aktuelle Browsergröße gut auf das *Smart Map Browsing* aus.
Ein nicht zu verachtender Usabilityaspekt bei WebMapping-Anwendungen ist die Gestaltung der Karte. Ein überzeugendes Kartenlayout; klar lesbare Signaturen, Symbole und Schriften; selbsterklärende Farben in ansprechender Kombination sowie eine gut gewählte grafische Bilddichte beeinflussen die Attraktivität einer (Web-)Karte positiv [Räber u. Jenny, 2003].

Übersichtskarte Die Navigationsmöglichkeiten der Übersichtskarte sind ein wichtiges *Smart Map Browsing* Kriterium. Per Klick/Doppelklick sowie Drag&Drop verschiebt sich der markierte Ausschnitt in der Referenzkarte. Nach jeder Interaktion mit der Übersichtskarte zentriert sich der Ausschnitt. Dies ist wichtig für die Orientierung und die erneute Interaktion des Nutzers.

Beim Verschieben der Hauptkarte passt sich die Übersichtskarte unmittelbar an. Hierbei ist eine zeitgleiche Verschiebungsanimation (*animated panning*) von Haupt- und Referenzkarte ideal. Der Nutzer wird zur neuen Kartenposition »geführt«; er behält so die Orientierung.

Die Übersichtskarte passt darüber hinaus auch ihre Zoomstufe an. Der Maßstab ist dabei immer um einen angemessenen Faktor größer als die Hauptkarte, so dass die Referenzkarte dem Nutzer auch tatsächlich als »Übersicht« dient.

Die Übersichtskarte zu minimieren ist ein nützliches Feature. Eine Platzierung innerhalb der Hauptkarte bietet einen größeren Hauptkartenbereich und stellt die Übersichtskarte mehr ins Zentrum der Aufmerksamkeit.

Zur weiteren Verbesserung der Usability wäre es hilfreich, alle aktivierten Ebenen aus der Hauptkarte auch in der Übersichtskarte anzuzeigen.

Ebenenübersicht Bei De-/Aktivierung einer Ebene aktualisiert sich die Karte automatisch (ohne manuelle Bestätigung). Dies ist ein wesentliches Merkmal von *Smart Map Browsing*. Eine Minimierung der Übersicht ermöglicht dem Benutzer seine Aufmerksamkeit auf die Karte zu konzentrieren. Des Weiteren sind eine änderbare Ebenenreihenfolge und die Integration einer Legende in die Ebenenübersicht je nach Anwendung sinnvoll.

Legende Legenden sind so knapp wie möglich zu fassen und müssen einen Bezug zum Karteninhalt haben. Die Elemente der Legende sollten nach ihrer Wichtigkeit geordnet und mit ähnlichen Elementen gruppiert dargestellt werden [Räber u. Jenny, 2003]. Eine Minimierung der Legende dient der besseren Übersicht.

Maßstab Maßstabsangaben und -balken machen nur Sinn, wenn sie für jede Zoomstufe neu berechnet werden [Räber u. Jenny, 2003]. Ein Maßstabsbalken sollte dem Benutzer beim *Map Browsing* zur Verfügung stehen, um Entfernungen leicht auf einen Blick abschätzen zu können. Eine Balkengrafik ist (speziell für Kartenanwendungen mit »ungeübten« Kartennutzern) empfehlenswert und einer Maßstabszahlanzeige vorzuziehen. Idealerweise befindet sich die Grafik halbtransparent innerhalb der Karte.

Werkzeugleiste Zu umfangreiche Werkzeugleisten können sich nachteilig auf die Usability der Anwendung auswirken. Für ein »smartes« *Map Browsing* wäre eine Reduzierung auf die erweiterten Abfrage- und Analysefunktionen ratsam; einfache Zooming- und Panning-Funktionen lassen sich besser durch eine Pan-Zoom-Navigationsleiste oder andere intuitive Pan-Zoom-Interaktionen (z. B. Mausrad, Doppelklick) realisieren. Wichtig für ein flüssiges *Map Browsing* ist, dass ein Wechsel zwischen Panning und Zooming *ohne* Umschalten eines Werkzeuges möglich ist.

Der Mauscursor sollte sich (soweit dies sinnvoll erscheint) an das gewählte Werkzeug anpassen, um den Nutzer eine visuelle Rückmeldung seiner Auswahl zu geben.
Das Zoom-History-Feature sei hier als hilfreiches *Smart Map Browsing* Feature erwähnt. Dem Nutzer ist es dadurch möglich seine letzen Pan- und Zoomschritte rückgängig zu machen oder wiederherzustellen.

Zoomnavigationsleiste Eines der derzeit bedeutendsten *Smart Map Browsing* Features ist die Zoomnavigationsleiste. Sie bietet dem Benutzer einen hohen Grad an Interaktivität und eine gute Visualisierung der aktuellen Zoomstufe in Bezug auf das verfügbare Zoomstufenintervall.
Unter dem Begriff *continuous zooming* wird die Möglichkeit verstanden beim Bewegen des Zoomsliders die Karte on-the-fly durch Verkleinern oder Vergrößern zu skalieren. Erst nach dem Loslassen des Sliders wird die Karte in der entsprechenden Zoomstufe neu geladen. Dieses Feature verbessert die Nutzerorientierung beim Zoomvorgang und ist ein gutes Beispiel für *Smart Map Browsing*.

Pannavigationssteuerkreuz Ein Steuerkreuz bietet eine intuitive und präzise Panning-Möglichkeit. Die Integration in eine Pan-Zoom-Navigationsleiste ist einer Kartenrandnavigation vorzuziehen. Die Navigationselemente sind so klarer stukturiert und verkürzen die »Mauswege« beim Navigieren.

Nach [Räber u. Jenny, 2003] müssen Steuerelemente in WebMapping-Anwendungen selbsterklärend und effizient sein. Sie sollten logisch gruppiert angeordnet werden, um den Anwender die Kartennutzung zu erleichtern. Der Benutzer soll gewünschte Funktionen schnell finden und darf nicht durch zu viele Möglichkeiten überfordert werden. Die Steuerung einer Karte mit einer Vielzahl von Elementen ist nicht trivial und erfordert einen Lernprozess vom Kartenanwender. Eine intuitive Kartennavigation sollte im Mittelpunkt stehen; komplizierte Bedienungen von erweiterten Funktionen sollten so weit wie möglich vereinfacht bzw. ggf. auf ein Minimum reduziert werden [Räber u. Jenny, 2003].

Steuerkomponenten in WebMapping-Anwendungen müssen nicht zwingend von der Karte getrennt platziert werden, sondern können sich auch gut innerhalb der Hauptkarte befinden. Dies bietet den Vorteil, dass der Benutzer Interaktionsmöglichkeiten früher wahrnimmt, da die Karte bereits im Zentrum seiner Aufmerksamkeit steht. Eine ansprechende grafische Gestaltung der Elemente wirkt dabei unterstützend [Räber u. Jenny, 2003].

B) Pan- und Zoom-Verhalten

Zooming per Doppelklick Zoomen durch einen Doppelklick auf die Karte ermöglicht einen problemlosen Wechsel zwischen Pan- und Zoomvorgängen ohne den Modus explizit umschalten zu müssen. Die Benutzbarkeit der Anwendung wird durch diese Navigationsmöglichkeit intuitiver und ist hiermit als Eigenschaft von *Smart Map Browsing* definiert.

Zooming per Mausrad Ebenso wie das Doppelklickverhalten wird auch durch die Mausradbenutzung ein komfortabler und einfacher Wechsel in den Zoommodus möglich.
Ein *Smart Map Browsing* Feature ist es dann, wenn folgendes korrektes Zoomverhalten vorliegt: Die geografische Position unterhalb des Mauszeigers zum Zeitpunkt des Mausradbetätigens befindet sich nach dem Zoomvorgang an der gleichen Pixelposition des dargestellten Kartenausschnitts wie vor dem Zoom. Ein Zooming mit Zentrierung auf den Punkt der Mausposition oder ein Zoomen ohne Berücksichtigung der Mausposition ist nicht erwartunskonform und somit nicht im Sinne der *Smart Map Browsing* Definition.

Zooming per Zoombox Das Zoomen in den Bereich einer aufgezogenen Zoombox ist mittlerweile Standard fast jeder WebMapping-Anwendung und wird von Benutzern erwartet. Alternativ zum Zoombox-Button in der Werkzeugleiste sollte auch mit gedrückter Shift-Taste das Zeichnen einer Zoombox möglich sein. Eine einfarbige, halbtransparente Unterlegung des aufgezogenen Bereichs dient dem Benutzer als visuelle Rückmeldung seiner Interaktion.
Es handelt sich hierbei um ein nützliches Feature, das zwingend erforderlich für ein gutes *Smart Map Browsing* ist.

Zooming/Panning per Tastatur Eine einfache Navigationssteuerung sollte über die Plus- und Minus- bzw. Pfeiltasten möglich sein. Eine erweiterte, intuitive Tastenbelegung zum Verschieben der Karte ist zu Gunsten der Usability wünschenswert (naheliegende Tasten für großstufigere Kartenverschiebungsschritte sind Pos1, Ende, Bild↑, Bild↓).

Zoomstufen Das Zoomlevelintervall sollte klar eingeschränkt sein. Bei Erreichen der minimalen bzw. maximalen Zoomstufe ist die ZoomOut- bzw. ZoomIn-Funktion deaktiviert; die entsprechenden Buttons symbolisieren auch visuell eine Inaktivität. Die mögliche Zoomtiefe lässt sich zur besseren Orientierung des Benutzers auf einer Zoombar abbilden.

Zoom-Reset Eine Zoom-Reset-Funktion ist eine vom Benutzer erwartete Funktion zum Zurücksetzen der Karte auf ihre maximale Ausdehnung. Folgt man der o. g. Usabilityrichtline, wonach ähnliche Elemente logisch gruppiert werden sollen, wäre eine Integration des Zoom-Reset-Buttons in die Pan-Zoom-Bar empfehlenswert. Sämtliche Zoom- und Verschiebungsfunktionen könnten so aus der Werkzeugleiste ausgelagert werden (vgl. auch Punkt *(A) GUI-Komponenten*).

Animation Der Einsatz von Animationen in (Web-)Karten kann verschiedene Ziele haben: Die Aufmerksamkeit des Betrachters erhöhen, seine Aufmerksamkeit auf spezielle Objekte lenken, seine Orientierung während der Navigation verbessern oder ihn durch ein Thema führen. Es unterscheiden sich 2 Typen von Animationen: die temporale und die nontemporale Animation [Dickmann, 2004; Räber u. Jenny, 2003].

In der temporalen Animation werden räumliche Veränderungen in einem bestimmten Zeitintervall dargestellt [Dickmann, 2004; Räber u. Jenny, 2003]. Ein Beispiel wäre der Bevölkerungswachstum einer Stadt im letzen Jahrhundert. Der Benutzer sollte durch selbsterklärende Steuerelemente in die Animation eingreifen können. Komplexe Sachverhalte in Karten können durch zeitliche Abläufe verständlicher dargestellt werden. Solche Animationen sollten im Bereich der Hauptkarte ablaufen und das interaktive *Map Browsing* einer WebMapping-Anwendung nicht ersetzen.

Die nontemporale Animation präsentiert räumliche Daten *eines* konkreten Zeitpunkts in verschiedenen Darstellungen [Dickmann, 2004; Räber u. Jenny, 2003]. Ein Beispiel wäre eine automatische »Zoomfahrt« auf ein bestimmtes Ziel hin. Der Benutzer wird durch nontemporale Animationen zu einem bestimmten Ziel geführt. Kombiniert mit der Kartennavigation lassen sich so dynamische Pan- und Zoomvorgänge realisieren, die bei [OpenLayers, b] mit den Begriffen *animated panning* und *animated zooming* beschrieben werden. Der Benutzer behält während solcher Animationsprozesse eine gute Orientierung und bekommt ein Gefühl für die Entfernung bzw. den Maßstab.

Kartographische Animationen erhöhen die Aufmerksamkeit des Benutzers und bieten der Karte einen bemerkenswerten »Mehrwert«. Animationen (im angemessenen Umfang) sind eine wichtige Eigenschaft von *Smart Map Browsing*.

Tiling Eines der grundlegendsten Eigenschaften von *Smart Map Browsing* ist die Kachelung von Karten (Tiling). Dabei ist ein clientseitiges Tiling dem serverseitigen Verfahren vorzuziehen, um aufwändige Serverrechenzeiten zu reduzieren (vgl. Abschnitt 2.3.5). Belegbare Messungen sind hierfür nicht erfolgt. Im Idealfall ist mit Tiling ein absolut verzögerungsfreies Verschieben der Karte möglich. Beim Zoomvorgang bauen sich die Kacheln nacheinander spiral- oder sternförmig auf, beginnend in der Kartenmitte. Der Benutzer erhält durch diesen dynamischen Ladeprozess eine sichtbare Rückmeldung über den Ladefortschritt der Karte.

Ladezeit Das wichtigste Kriterium einer Internetseite ist ihre schnelle Ladezeit [Nielsen, 2000]. Dabei muss zwischen objektiver und subjektiver Ladezeit unterschieden werden. Erstere ist die Zeit, die tatsächlich benötigt wird; letztere bezieht sich darauf, wie lange dem Nutzer die Zeit erscheint. Nach Jakob Nielsen sollte die objektive Ladezeit eine Website acht Sekunden nicht überschreiten. Alles darüber hinaus stört empfindlich die Gebrauchstauglichkeit einer Web-Anwendung.
Bezogen auf WebMapping-Anwendungen ist die wichtigste Frage, wann der Benutzer eine fertige Karte zu sehen bekommt. Weniger von Bedeutung ist die Ladezeit der gesamten Seite mitsamt allen Details und ggf. vorgeladenen Kacheln für den nicht sichtbaren Bereich [Nielsen, 2000].

Beim Verschieben einer Karte sollte im Idealfall keine Verzögerung entstehen (vgl. Punkt *Tiling*). Ein dynamischer Kartenaufbau durch nacheinander erscheinende Kacheln bewirkt eine subjektiv schnellere Ladezeit als beim Aufbau einer ungekachelten Karte, die komplett erscheint. Eine Technik, um beim Einsatz von Tiling die objektive Ladezeit zu beschleunigen, ist die Nutzung eines *Tile Caches*. Eine genauere Betrachtung folgt im nächsten Abschnitt.

Es ist im Sinne der *Smart Map Browsing* Definition, dass der Benutzer vielfältige Möglichkeiten hat, mit der WebMapping-Anwendung zu interagieren. Die Gebrauchstauglichkeit steht bei jeder Interaktion im Vordergrund. Das Verhalten der Komponente(n) sollte dabei stets erwartungskonform sein.

3.2.3 Potenziale

Die wichtigsten Eigenschaften von *Smart Map Browsing* und ihre Potenziale im WebMapping-Bereich werden nachfolgend herausgestellt:

Tiling (an dieser Stelle wird nur vom *clientseitigen* Tiling gesprochen) ist zweifelsohne eines der bedeutendsten Merkmale von *Smart Map Browsing* und stellt aktuell ein großes Potenzial für gebrauchstaugliche WebMapping-Anwendungen dar. Der Einsatz von Tiling in WebMapping-Anwendung (vgl. Kapitel 2.3.5) bedeutet jedoch nicht zwangsläufig ein gutes *Smart Map Browsing*. Auch die leistungsstärkste Anwendung wird bei langen Antwortzeiten des Mapservers keine gute Usability bieten können. Tiling sagt noch nichts über die objektiven Ladezeiten der Kacheln aus. Ziel – im Sinne der *Smart Map Browsing* Definition – sollte es sein, die objektiven Ladezeiten so weit wie möglich zu reduzieren.

Eine Lösung dafür ist das serverseitige ***Tile Caching*** (ein clientseitiges Caching wird hier nicht betrachtet). Die Kacheln werden dabei bereits in fertig gerenderten Grafiken auf dem Server vorgehalten. Der Vorteil ist eine geringere Antwortzeit des Mapservers: Bei einer Kartenanfrage des Clients muss der Server die Karte nicht erst aufwändig generieren, sondern kann auf seine (in der Regel im Dateisystem abgelegten) fertigen Kacheln zurückgreifen und somit die angefragte Karte wesentlich schneller dem Client zurückliefern. Die Karten müssen dazu in einem vordefinierten Kachelgitter (*grid*) mit einheitlicher Kachelgröße vorliegen.
WMS Tile Caching[9] oder *WMS-Cached* – kurz WMS-C – stellt eine erste allgemeine Empfehlung dar, wie ein solcher Standard auf Grundlage der OGC WMS Spezifikation aussehen könnte. Einen konkreten Vorschlag für eine einheitliche Lösung beschreibt die *WMS Tiling Client Recommendation*[10], die als Ergebnis einer Tiling-Diskussionsrunde auf der *FOSS4G* Konferenz 2006[11] in Lausanne entstand. Auf Grundlage dessen ist die Implementierung von *TileCache*[12] – ein WMS-C konformer Server, verfügbar unter der BSD Lizenz – von dem

[9] http://wiki.osgeo.org/index.php/WMS_Tile_Caching [Abruf: 15.05.2007]
[10] http://wiki.osgeo.org/index.php/WMS_Tiling_Client_Recommendation [Abruf: 15.05.2007]
[11] http://www.foss4g2006.org [Abruf: 15.05.2007]
[12] http://www.tilecache.org [Abruf: 15.05.2007]

US-Unternehmen *MetaCarta*[13] entwickelt worden. *TileCache* ist ein auf Python basierender WMS/TMS[14]-Server mit Mechanismen zum Rendern und Cachen von Kacheln. Im einfachsten Fall kann *TileCache* mit Schreibzugriff auf eine Festplatte, der Fähigkeit Python CGI Skripte auszuführen und der Angabe eines beliebigen WMS-Servers einen eigenen, lokalen Cache aller Kacheln auf der Festplatte anlegen. Ein WMS-C oder TMS unterstützender Client (wie *OpenLayers*) kann die gecachten Kacheln anschließend abfragen. *TileCache* beschleunige die WMS-Anfragen um Faktor 10 bis 100. Bei Nutzung von *TileCache* unter *mod_python*[15] seien dabei mehr als 300 Anfragen pro Sekunde möglich [MetaCarta].
Serverseitiges Tile Caching bietet ein bedeutendes Potenzial für die weitere Entwicklung von *Smart Map Browsing* in WebMapping-Anwendungen. Trotz der sehr jungen Geschichte und der noch ausstehenden OGC-Standardisierung von WMS-C sind schon heute beachtliche Geschwindigkeitsfortschritte in der Ladezeit von WMS-Karten erkennbar. Demnach verspricht Tile Caching ein elementares *Smart Map Browsing* Feature zu werden.

Konzentriert man den Fokus auf die Steuerelemente von WebMapping-Applikationen ist die *Smart Map Browsing* Fähigkeit der **Zoomnavigationsleiste** bemerkenswert. Die Analyse (vgl. Abschnitt 3.1) zeigt bei den vier Anwendungen, die eine solche Komponente einsetzen, eine spürbar verbesserte Benutzbarkeit. In Kombination mit einem Panning-Steuerkreuz und einem Zoom-Reset-Button lassen sich o. g. Usabilityrichtlinien umsetzen. Die definierten Aspekte in der Zoomtiefenorientierung (vgl. Abschnitt 3.2.2) bieten zusätzliche Möglichkeiten Einfluss auf die Usability auszuüben.
Diese Tatsachen lassen den Schluss zu, dass der Einsatz einer Zoombar ein großes Potenzial darstellt.

Eng damit verbunden sind die noch sehr jungen und innovativen Lösungen Kartennavigationsvorgänge zu animieren. ***Animated panning* und *animated zooming*** sind *Smart Map Browsing* Features, die bis heute nur in drei der untersuchten Anwendungen ansatzweise realisiert sind.
Animated panning bietet, im Zusammenhang mit Tiling, eine gute Möglichkeit den Benutzer animiert und verzögerungsfrei von A nach B zu führen. Dieser dynamische Prozess zeigt dem Anwender, ähnlich wie beim Drag&Drop-Panning, dass der eingeschränkte Kartenausschnitt nur ein Teil einer ganzen, nahtlos zusammenhängenden Karte ist – ein klarer Vorteil für die Orientierung des Benutzers.
Beim *animated zooming* Feature handelt es sich im Kern um einen automatischen Zoomprozess, um den Übergang von Zoomstufe A zu Zoomstufe B zu animieren. Die Karte wird während des Zoomvorgangs skaliert und nach Erreichen von Zoomstufe B neu gezeichnet. Diese Animationsart ist für eine oder mehrere Zoomstufen möglich und kann dabei sowohl in die Karte hinein- als auch herauszoomen. Auch eine Kombination aus *animated panning* und *zooming* ist denkbar.
Das *animated zooming* Feature impliziert noch eine weitere Funktion: Das Ziehen des Sliders einer Zoombar bewirkt ein Skalieren der aktuellen Karte durch Verkleinern oder Vergrößern.

[13] http://www.metacarta.com [Abruf: 15.05.2007]
[14] http://wiki.osgeo.org/index.php/Tile_Map_Service_Specification [Abruf: 15.05.2007]
[15] http://www.modpython.org [Abruf: 15.05.2007]

Diese Skalierung läuft stufenlos ab. Es handelt sich hierbei um eine *manuelle* nontemporale Animation.
Es gibt noch keine klare Nomenklatur für Zoomanimationen im WebMapping-Bereich. Aufgrund der Analogie zum *animated panning* Verhalten bietet sich die Verwendung des Begriffs *animated zooming* für animierte Zoomvorgänge an. *Google Maps* hingegen beschreibt seine Zoomanimation per Mausrad (funktioniert bisher nur unter *Windows*-Betriebssystemen) mit dem Begriff *continuous zooming*. Für die vorliegende Arbeit wird der Begriff *continuous zooming* ausschließlich für *manuelle* Zoomvorgänge beim Einsatz des Zoomsliders verwendet werden. *Automatisch* ablaufende Zoomprozesse (hervorgerufen z. B. per Klick, Doppelklick, Mausrad oder Zoombox) werden hier mit dem Begriff *animated zooming* beschrieben. Darüber hinaus stehe *animated zooming* als Sammelbegriff für alle animationsgestützten Zoomprozesse in WebMapping-Anwendungen, inklusive *continuous zooming*. Der Begriff *seamless zooming* ist hier mit *continuous zooming* gleichzusetzen.
Es liegt nahe, dass diese zwei jungen Features (*animated panning* und *zooming*) eine bedeutende Rolle in der Entwicklung von *Smart Map Browsing* spielen werden. Aktuell sind Performanceprobleme (durch beschränkte Bandbreite oder Client-Hardware) bei umfangreichen Animationen wahrscheinlich, was die Usability stören könnte. Die Potenziale dieser Erweiterungen sind aber eindeutig erkennbar und könnten sich mittelfristig zu einem nicht mehr wegzudenkenden Merkmal von WebMapping-Anwendungen etablieren.

3.3 Anforderungsanalyse

Beim Betrachten der Untersuchungsergebnisse aus Abschnitt 3.1.4 überrascht *p.mapper* mit seinem außergewöhnlichen *continuous zooming* Feature – ein Alleinstellungsmerkmal. *OpenLayers* überzeugt dagegen in fast allen untersuchten Bereichen und besitzt eine der aktivsten Communities unter den Freien WebMapping-Anwendungen. Darüber hinaus lässt sich nach der Begriffsprägung von *Smart Map Browsing* (vgl. Abschnitt 3.2) ein großes Entwicklungspotenzial für *OpenLayers* und animierte Zoomvorgänge ableiten. Ein ähnliches Zoom-Feature, wie *p.mapper* es anbietet, wäre für *OpenLayers* eine ideale Ergänzung.

Aus diesen Gründen wird für den praktischen Entwicklungsteil der vorliegenden Arbeit die Freie WebMapping-Anwendung *OpenLayers* ausgewählt und mit einem neuartigen *animated zooming* Feature erweitert. Dieser Abschnitt definiert die genauen Anforderungen an eine derartige Erweiterung.

3.3.1 Zielbestimmung

Ziel des praktischen Entwicklungsteils dieser Diplomarbeit ist es, am Beispiel der Freien WebMapping Anwendung *OpenLayers* das *Smart Map Browsing* Feature *animated zooming* zu implementieren.

Diese Erweiterung soll in erster Linie eine Navigationshilfe darstellen, die dem Benutzer eine verbesserte Orientierung beim Zoomvorgang gewährleistet.
Anmerkung: Eine Verbesserung ist in dieser Arbeit nicht nachweisbar, da keine empirischen Messungen durchgeführt werden. Die *Smart Map Browsing* Definition lässt aber den Schluss zu, dass sich eine derartige Erweiterung positiv auf die Usability auswirken könnte. Die Vermutung liegt nahe, dass im Vergleich zu nicht animierten Zoomvorgängen (bei denen die Karte sprunghaft die Zoomstufe wechselt), ein stufenloser Zoomprozess die Aufmerksamkeit und die Orientierung des Benutzers spürbar verbessert. Darüber hinaus lässt sich eine Verbesserung auch an der Akzeptanz des Features von der Community ableiten. Auf Mailinglisten, wo über *animated zooming* diskutiert wird, herrscht meist einheitliche Zustimmung, dass dies eine attraktive Erweiterung sei.

3.3.2 Musskriterien

Die nachfolgenden *Muss*-Anforderungen beschreiben die Funktionalität der Anwendung aus dem Blickwinkel des Benutzers:

1. Der Nutzer soll mit gedrückter linker Maustaste den Zoomslider frei auf der Zoombar bewegen können. Dabei sind folgende Anwendungsfälle zu realisieren:
   a) Der Kartenausschnitt der Hauptkarte wird beim Bewegen des Zoomslides on-the-fly durch Verkleinern oder Vergrößern der Karte skaliert.
   b) Beim **ZoomIn** wird die Karte durch Vergrößern der sichtbaren Kacheln skaliert. Abhängig von der Zoomleveldifferenz werden die Kachel-Bitmapgrafiken durch diese Skalierung gering bis auffallend stark pixelig.
   c) Beim **ZoomOut** wird zunächst die Hauptkarte durch Verkleinern der sichtbaren Kacheln skaliert. Der dadurch entstehende weiße Rand (um die kleiner werdende Karte) soll so ergänzt werden, dass der Nutzer immer eine vollständig gefüllte Kartenfläche sieht. Die nachgeladene Kartenfläche wird analog zu den Kacheln des ursprünglichen Kartenausschnitts skaliert.
   d) Sind **mehrere Overlays** aktiviert, werden diese alle gemeinsam mit der gewählten Basisebene skaliert. Dadurch werden z. B. auch Symbole oder Schriften vergrößert oder verkleinert.
   e) Die **Übersichtskarte** enthält eine Markierung, deren Größe sich beim Zoomvorgang (zeitgleich mit den Änderungen in der Hauptkarte) anpasst. Die Markierung zeigt stets den aktuellen Bereich der Hauptkarte an.
2. Sobald der Nutzer den Zoomslider durch Loslassen der gedrückten linken Maustaste an einer bestimmten Zoomstufe »absetzt«, werden die bis dahin nur skalierten Bitmap-Kacheln (mit allen aktivierten Overlays) neu gezeichnet.

3.3.3 Wunschkriterien

Es sind folgende *Kann*-Anforderungen für das *animated zooming* Feature definiert:

1. Vollständige *animated zooming* Unterstützung für alle von *OpenLayers* zur Verfügung gestellten Ebenentypen.
2. Zoomanimation per Zoombar(-Buttons)
   a) Klickt der Benutzer auf den + bzw. – Button am Zoombarende, wird die Karte um genau eine Zoomstufe hinein- bzw. herausgezoomt.
   b) Klickt der Benutzer auf einen Punkt auf der Zoombar, wird die Karte bis zu der entsprechenden Zoomstufe hinein- oder herausgezoomt.

3. Zoomanimation per Doppelklick
    a) Macht der Benutzer einen Doppelklick auf die Karte, wird um genau eine Zoomstufe in die Karte hineingezoomt. Die geografische Position des Doppelklicks ist als neuer Mittelpunkt des Kartenausschnitts definiert.
    b) Bei der Doppelklick-Animation werden Zoom- und Panvorgänge kombiniert. Es ist hier nur eine Animation des Zoomprozesses gefordert; die Verschiebung läuft vor dem Zoomen ohne Animation ab.
4. Zoomanimation per Mausrad
    a) Beim Bewegen des Mausrads um eine oder mehrere Stufen nach oben (vom Benutzer weg) bzw. nach unten (zum Benutzer hin), wird die Karte um genau eine Zoomstufe hinein- bzw. herausgezoomt.
    b) Das Mausrad-Zoomen verhält sich nach der unter Abschnitt 3.2.2 beschriebenen *Smart Map Browsing* Eigenschafts-Definition des Mausradzoomens: *»Die geografische Position unterhalb des Mauszeigers zum Zeitpunkt des Mausradbetätigens befindet sich nach dem Zoomvorgang an der gleichen Pixelposition des dargestellten Kartenausschnitts wie vor dem Zoom.«*
5. Zoomanimation per Zoombox
    a) Nachdem der Benutzer einen Bereich mit der Maus auf der Karte aufgezogen hat und die (linke) Maustaste loslässt, wird bis auf diesen Ausschnitt in die Karte hineingezoomt.
    b) Klickt der Benutzer mit dem Zoombox-Werkzeug ohne Aufziehen einer Zoombox in die Karte, wird um genau eine Zoomstufe in die Karte hineingezoomt. Die geografische Position des Klicks ist als neuer Mittelpunkt des Kartenausschnitts definiert.
    c) Bei der Zoombox-Animation werden Zoom- und Panvorgänge kombiniert. Es ist hier nur eine Animation des Zoomprozesses gefordert; die Verschiebung läuft vor dem Zoomen ohne Animation ab.
6. Zoomanimation per Tastatur
    a) Drückt der Benutzer die + bzw. – Taste, wird die Karte um genau eine Zoomstufe hinein- bzw. herausgezoomt.

Für jede Zoomanimation gilt:
Der animierte Zoomvorgang läuft automatisch in einer definierten Zeit ab. Der Zoomslider bewegt sich zeitgleich zur Animation bis zur entsprechenden Zielposition nach oben oder unten. Befindet sich die Karte in der minimalen bzw. maximalen Zoomstufe, ist keine ZoomOut- bzw. ZoomIn-Animation möglich. Die Markierung in der Übersichtskarte passt sich stets während der Zoomanimation an den Ausschnitt der Hauptkarte an. Solange eine Zoomanimation abläuft ist ein erneutes Auslösen eines Zoomvorgangs nicht möglich.

3.4 Technische Untersuchung von *OpenLayers*

An dieser Stelle wird die Funktionsweise von *OpenLayers* genauer betrachtet. Diese Analyse dient zum besseren Verständnis der Anwendung und ist Voraussetzung für die nachfolgende Konzeptplanung und Realisierung.

3.4.1 Allgemein

OpenLayers ist eine reine JavaScript-API zum Erstellen von WebMapping-Anwendungen und bietet Unterstützung zum Anzeigen zahlreicher (standardisierter) Formate (OGC WMS, OGC WFS, GeoRSS, ka-Map, WorldWind u. v. m.), die als Ebenen in *OpenLayers* eingebunden werden.

Darüber hinaus bietet *OpenLayers* ein Verschieben und Zoomen von Karten, clientseitiges Tiling, Markers, »Popup-Blasen«, verschiedene anpassbare Steuerelemente, Tastatursteuerungsbefehle, einen robusten Event-Handling-Mechanismus u. a. m. Jeder Teil von *OpenLayers* ist konfigurierbar [OpenLayers, d].

Ein Python/Shell-basiertes Skript ermöglicht das automatische Bauen einer *lite*-Version von *OpenLayers*, wodurch nur gewünschten Klassen in eine einzige JavaScript-Datei integriert werden können [OpenLayers, a]. Dies ist das Ergebnis einer Diskussion auf der FOSS4G-Konferenz im September 2006, wonach eine solche Lösung unter dem Namen *webmap.js* gefordert wurde [OSGeo, 2006a,b].

Die Zielgruppe von *OpenLayers* sind alle Anwender oder Entwickler, die eine Karte im Internet darstellen oder eine kartenbasierte Internetanwendung aufbauen möchten [OpenLayers, d].

Geschichte

Die Veröffentlichung von *Google Maps* im Februar 2005 bewirkte große Aufmerksamkeit. Aufgrund zahlreicher *reverse engineerings* des *Google Maps* Codes (u. a. von Phil Lindsay[16]) antwortete Google kurz darauf, im Juni 2005, mit einer ersten *Google Maps* API[17]. Es fehlte jedoch die Möglichkeit, diese kommerziell einzusetzen [Ramsey, 2006; OpenLayers, f].

Das US-Unternehmen *MetaCarta* nahm sich dieses Mangels an und entwickelte, unter der Mitwirkung von Phil Lindsay, einen ersten Prototypen des späteren *OpenLayers*, der auf der *Where 2.0* Konferenz Ende Juni 2005 in San Francisco präsentiert wurde[18]. Ein Jahr später, kurz nach der *Where 2.0* im Juni 2006 in San Jose, veröffentlichte das Team um die Hauptentwickler Schuyler Erle, Christopher Schmidt und Erik Uzureau am 5. Juli 2006 die erste offizielle Version (1.0) von *OpenLayers*. Im August 2006 erschien das Projekt bereits unter der Versionsnummer 2.0 [OpenLayers, d,f].

[16] http://stuff.rancidbacon.com/gmaps-standalone [Abruf: 15.05.2007]
[17] http://www.google.com/apis/maps [Abruf: 15.05.2007]
[18] http://conferences.oreillynet.com/cs/where2006/view/e_sess/9310 [Abruf: 15.05.2007]

Aktuell befindet sich *OpenLayers* im Inkubationsprozess der *Open Source Geospatial Foundation (OSGeo)*, mit dem Ziel, im Sommer 2007 als Softwareprojekt in diese unabhängige Organisation aufgenommen zu werden. Parallel laufen ernsthafte Bestrebungen, Kernfunktionalitäten von *OpenLayers* in andere Freie WebMapping-Anwendungen zu integrieren [OpenLayers, d] – aktuelle Diskussionen werden derzeit bei *ka-Map*[19,20], *Mapbuilder*[21] und *Mapbender*[22] geführt.

3.4.2 Klassenübersicht

Die *OpenLayers*-API gliedert sich durch die objektorientierte Programmierweise in eine Vielzahl von Klassen. Die wichtigsten werden nachfolgend kurz beschrieben. Auf die restlichen Klassen soll hier nicht weiter eingegangen werden, da sie für das grundlegende Verständnis der API nicht von Bedeutung sind. Für detailliertere Informationen wird auf die *JSDoc*-Dokumentation[23] verwiesen. Ein vollständiges Klassendiagramm von *OpenLayers* (Version 2.4 RC5; Stand: 25.5.2007) befindet sich im Anhang B auf Seite 117.

OpenLayers.Map ist die zentrale Klasse der *OpenLayers*-API. Sie erstellt die Hauptkarte der Anwendung und stellt zahlreiche Methoden zum Verwalten der Kartenanzeige zur Verfügung. Dazu zählen u. a. das Darstellen von Ebenen und Bedienelementen, das Verändern der Zoomstufen und das Verschieben der Karte. Zusätzlich ermöglicht die Klasse, über zahlreiche get-Methoden, den aktuellen Kartenstatus abzufragen.

OpenLayers.Layer: Die Ebenen repräsentieren den wichtigsten Bestandteil von *OpenLayers*. Alle Kartendarstellungen basieren auf der *Layer*-Klasse. *Layer.js* erzeugt einzelne Ebenen, setzt deren Sichtbarkeit und Auflösung und stellt grundlegende get-Methoden zur Verfügung.
OpenLayers.Layer dient als Basisklasse für eine Vielzahl von speziellen Unterklassen. In der *OpenLayers*-Version 2.4 RC5 werden folgende Ebenentypen unterstützt: Freie kachelbasierte Ebenen (*WMS, MapServer, KaMap, TMS, WorldWind*), Freie ungekachelte Ebenen (*WMSUntiled, MapServerUntiled*), Ebenen von proprietären Drittanbietern (*Google, VirtualEarth, Yahoo, MultiMap*) sowie die Ebenen *Image* (zum Darstellen einer Rastergrafik) und *Canvas* (zum Zeichnen von farbigen Linien, nur Overlay-Darstellung möglich).

Nachfolgend werden vier Ebenenklassen genauer beschrieben:

OpenLayers.Layer.Grid
Abgeleitet von der *HTTPRequest*-Klasse steht *Grid* als eine wichtige Basisklasse für die o. g. Freien kachelbasierten Ebenen zur Verfügung. *Grid.js* unterteilt die Ebenen in Kacheln, verwaltet diese in einem Array und lädt sie in spiralförmiger Reihenfolge, beginnend in der Mitte.

[19] http://lists.maptools.org/pipermail/ka-map-users/2006-June/001729.html [Abruf: 15.05.2007]
[20] http://ka-map.ominiverdi.org/wiki/index.php/OpenLayers_ka-Map_Merge [Abruf: 15.05.2007]
[21] http://geoservices.cgdi.ca/mapbuilder/demo/openlayers/index.html [Abruf: 15.05.2007]
[22] http://www.mapbender.org/index.php/Open_layers [Abruf: 15.05.2007]
[23] http://dev.openlayers.org/docs/overview-tree.html [Abruf: 15.05.2007]

OpenLayers.Layer.WMS
Als Unterklasse von *Grid* bekommt *WMS* die URL des WMS-Servers übermittelt und ist zuständig für die Generierung der einzelnen Kachel-WMS-URLs anhand der gewünschten Kachelbegrenzung (*bounds*).

OpenLayers.Layer.WMS.Untiled
Die *Untiled*-Klasse bildet den gesamten Kartenausschnitt in einer WMS-Kachel-URL ab. Zu beachten ist dabei die von vielen WMS-Servern in der Regel vordefinierte maximale Kachelgröße von 2048 Pixeln; fordert *OpenLayers* einen größeren Kartenausschnitt an, gibt der WMS-Server eine Fehlermeldung zurück und die Karte bleibt leer.

OpenLayers.Layer.Image
Abgeleitet von *Layer* erlaubt die *Image*-Klasse das Darstellen einer einzelnen Bitmap-Grafik als Kartenebene. Gefordert sind neben der URL der Grafik die Pixelgröße sowie die geografische Ausdehnung des zu verwendenden Bildes.

OpenLayers.Control: Bedienelemente in *OpenLayers* sind Elemente zur Kartennavigation sowie zur Darstellung von Karteninformationen (z. B. Maßstab). Die *Control*-Klasse dient als Basisklasse für alle Bedienelemente. Nachfolgend eine Auswahl:

OpenLayers.Control.PanZoom
erstellt eine Pan-Zoom-Navigation mit einem Pan-Steuerkreuz (4 Pfeil-Buttons) sowie jeweils einem ZoomIn-, ZoomOut- und ZoomReset-Button. Der *slideFactor*-Parameter definiert die Anzahl der Pixel, um die die Karte beim Benutzen der Pan-Steuerbuttons verschoben wird.

OpenLayers.Control.PanZoomBar
erstellt eine Pan-Zoom-Navigationsleiste mit einem Pan-Steuerkreuz sowie einer Zoombar mit einem Zoomslider und jeweils einem ZoomIn- und ZoomOut-Button an den Enden der Leiste (vgl. Abb. 2.3 auf Seite 17).
Die Zoombar ist für die Entwicklung des *animated zooming* Features wesentlich. Aus diesem Grund wird ihre Funktionsweise hier im Detail erläutert.
Der Nutzer hat drei Möglichkeiten die Zoombar zu benutzen:

1. Mausklick auf den + oder – Button am jeweiligen Ende der Zoombar.
   Damit vergrößert bzw. verkleinert sich der Kartenausschnitt sprunghaft um jeweils eine Zoomstufe.
2. Mausklick auf eine beliebige Stelle der Zoombar.
   Der Slider bewegt sich zur nächstliegenden vordefinierten Zoomstufe. Der Kartenausschnitt aktualisiert sich sprunghaft auf die entsprechende Zoomstufe.
3. Mittels Drag&Drop den Zoomslider stufenlos über die Zoombar bewegen.
   Sobald der Nutzer die gedrückte linke Maustaste loslässt, rastet der Slider auf der nächstliegenden vordefinierten Zoomstufe ein. Der Kartenausschnitt aktualisiert sich sprunghaft auf die entsprechende Zoomstufe.

OpenLayers.Control.OverviewMap
stellt eine kleine Übersichtskarte dar (vgl. Abb. 2.3; S. 17). Sie zeigt den aktuellen Kartenausschnitt der Hauptkarte und dient als Positions- und Navigationshilfe für den Anwender. Die Übersichtskarte liegt standardmäßig im Bereich der unteren rechten Ecke der Karte und lässt sich durch einen Button am Kartenrand minimieren. Die Klasse stellt u. a. Funktionen zum Abfragen und Setzen des Markierungsrahmens bereit, der mittels definierter Maus-Events verschoben werden kann.

OpenLayers.Control.KeyboardDefaults
definiert auf welche Tastaturbefehle wie reagiert werden soll.

OpenLayers.Control.MouseDefaults
definiert das Verhalten der Karte bei Maus-Events. Dazu gehören Klick-, Doppelklick-, Mausrad- und Mausbewegungs-Events.

OpenLayers.Tile: Wird eine Karte in Kacheln unterteilt, ist jede Kachel durch ein *Tile*-Objekt definiert. Zu deren Erzeugung muss die zugehörige Ebene, die Pixelposition, die geografische Kachelausdehnung, die URL und die Pixelgröße der Kachel angegeben werden. Die Standardkachelgröße beträgt 256 Pixel. Über eine Option beim Initialisieren der Karte lässt sich diese Größe ändern.

OpenLayers.Tile.Image
Abgeleitet von der *Tile*-Klasse hält das *Tile.Image*-Objekt die eigentliche Kachelgrafik vor und legt beim Zeichnen der Karte ein img-Div-HTML-Element für jede Kachel an.

OpenLayers.Events übernimmt das Event-Handling von *OpenLayers*.

OpenLayers.Pixel repräsentiert ein Bildschirmkoordinatenpaar in x- und y-Pixelwerten.

OpenLayers.Size repräsentiert ein Pixelgrößenwertepaar in Breite und Höhe.

OpenLayers.LonLat repräsentiert ein geografisches Koordinatenpaar in geografischer Länge (*longitude*) und Breite (*latitude*).

OpenLayers.Bounds repräsentiert einen rechteckigen Bereich (*bounding box*), dessen vier Grenzen (links, unten, rechts, oben) mit geografischen Koordinaten im *float*-Format angegeben werden. Die *Bounds*-Klasse stellt verschiedene get-Funktionen (z. B. Mittelpunkt und Pixelausdehung der *bounding box*) und Vergleichsfunktionen (z. B. ob sich ein Pixel innerhalb der definierten *bounding box* befindet) bereit.

OpenLayers.Util beinhaltet unterschiedliche Funktionen und Einstellungen, die keiner anderen Klasse von *OpenLayers* zuzuordnen sind.

3.4.3 Komponententests

OpenLayers benutzt das *Test.AnotherWay*[24] Framework für Komponententests (engl. *unit tests*). In der Version 2.4 RC5 (vom 25.5.2007) stellt *OpenLayers* knapp 1500 Unittests bereit, die sich über ein zentrales Webinterface[25] steuern lassen. Das Framework bietet die Möglichkeit, HTML- und JavaScript-Quellcode zu testen und die Ergebnisse anzuzeigen. Jede HTML-Testseite enthält eine oder mehrere JavaScript-Testfunktionen, die mit `test_` beginnen und ein Testobjekt `t` als Übergabeparameter definieren müssen [Test.AnotherWay; OpenLayers, g].

```
11 <html>
12 <head>
13     <script src = "../lib/OpenLayers.js"></script>
14     <script type = "text/javascript">
15         function test_Map_Zoom(t) {
16             t.plan(1);
17             var map = new OpenLayers.Map("map");
18             var layer = new OpenLayers.Layer.WMS("ABC",
19                                                  "http://example.com/123",
20                                                  {'layers':'test'});
21             map.addLayer(layer);
22             map.zoomTo(0);
23             t.eq(map.zoom, 0, "Map zoomed to level 0 correctly.");
24         }
25     </script>
26 </head>
27 <body>
28     <div id="map" style="width: 512px; height: 512px;"/>
29 </body>
30 </html>
```

Listing 3.2: Eine einfache Beispieltestseite mit einer Testfunktion

Listing 3.2 zeigt eine Beispieltestseite, wie sie für einen einfachen Test einer *OpenLayers*-Funktion eingesetzt werden kann. In Zeile 6 definiert die Testfunktion die Anzahl der geplanten Tests (hier: einer), von denen erwartet wird, dass sie erfolgreich durchlaufen werden. Stimmt die Anzahl nicht mit der tatsächlichen Anzahl von Tests überein, schlägt die ganze Testfunktion fehl. Die Zeilen 7 bis 12 erzeugen eine Karte mit einer WMS-Ebene und setzen die Zoomstufe auf 0. Zeile 13 testet die Aussage, dass sich die Karte tatsächlich in der Zoomstufe 0 befindet.

[24] http://straytree.com/TestAnotherWay/doc [Abruf: 15.05.2007]
[25] http://openlayers.org/dev/tests/run-tests.html [Abruf: 15.05.2007]

Das Test-Framework stellt fünf Testmethoden zur Verfügung, um selbstdefinierte Aussagen (engl. *assertions*) zu testen:

- `t.ok(boolean, ‘‘String for output’’)`
  Der Test ist erfolgreich, wenn die Aussage `boolean` wahr ist.
- `t.eq(value1, value2, ‘‘String for output’’)`
  Der Test ist erfolgreich, wenn `value1` dem erwarteten Wert `value2` entspricht.
- `t.like(string, regEx, ‘‘String for output’’)`
  Der Test ist erfolgreich, wenn `string` dem regulären Ausdruck `regEx` entspricht.
- `t.html_eq(HTML1, HTML2, ‘‘String for output’’)`
  Der Test ist erfolgreich, wenn `HTML1` dem erwarteten Wert `HTML2` entspricht; jeder HTML-Wert kann als DOM Element oder HTML-String angegeben werden.
- `t.fail(‘‘String for output’’)`
  Für explizites Fehlschlagenlassen eines Tests; erwartet nur Ausgabestring.

Beispiele und weiterführende Informationen zur Funktionsweise des Frameworks sind in der Dokumentation [Test.AnotherWay] zu finden.

4 Konzept

Nachdem die Ausgangssituation analysiert wurde und ein Überblick über die Funktionsweise von *OpenLayers* besteht, wird in diesem Kapitel ein Konzept für die Implementierung des *animated zooming* Features entworfen.

4.1 Zielsetzung

Ziel der praktischen Entwicklungsarbeit dieser Diplomarbeit ist es, eine neue *Smart Map Browsing* Erweiterung für die Freie JavaScript-WebMapping-Anwendung *OpenLayers* zu realisieren.

Das neue *animated zooming* Feature soll die Funktionalität von *OpenLayers* um eine gebrauchstaugliche Zoommöglichkeit erweitern: Bewegt der Nutzer den Zoomslider, passt sich der Maßstab der Hauptkarte stets zeitgleich und stufenlos an die aktuelle Sliderposition an. Die Karte wird dabei durch Verkleinern oder Vergrößern skaliert. Nach dem Loslassen des Sliders wird die Karte in der aktuellen Zoomstufe neu gezeichnet. Eine Verbesserung der Orientierung des Nutzers während des Zoomvorgangs steht bei der Realisierung dieser Erweiterung im Vordergrund.

Um einen bestmöglichen Nutzen aus dem zu implementierenden Feature zu erzielen, ist eine Zusammenarbeit mit den Entwicklern von *OpenLayers* angestrebt. Dabei sollen Anregungen der Entwickler Berücksichtigung finden. Ziel ist die Aufnahme des *animated zooming* Features in die aktuelle SVN-Entwicklungsversion, so dass die Erweiterung Bestandteil einer nächsten offiziellen Version von *Openlayers* wird.

4.2 Funktionsweise

An dieser Stelle wird das konzipierte *animated zooming* Skalierungsmodell im Detail vorgestellt. Dazu wird zunächst der gesamte Zoomprozess bei Nutzung des Zoomsliders erläutert und anschließend die Besonderheiten des ZoomOut-Vorgangs beschrieben.

Es seien Z_n die momentan aktuelle Zoomstufe der Hauptkarte, $Z_{\max}$ die maximale Zoom(In)stufe und $Z_{\min} = 0$ die minimale Zoom(Out)stufe für die gelten:
$Z \in \mathbb{N}$ und $Z_{\min} \leq Z_n \leq Z_{\max}$

Die folgenden Schritte beschreiben das entwickelte Skalierungsmodell und das genaue Vorgehen, solange der Benutzer den Zoomslider bedient. Ein Zustandsdiagramm (Abb. 4.1; S. 55) und ein Aktivitätsdiagramm (Abb. 4.2; S. 56) veranschaulichen dieses Modell. Dabei beziehen sich die Zahlen in den Aktionszuständen des Aktivitätsdiagramms auf diese einzelnen Schritte:

1. Der Nutzer drückt mit der linken Maustaste auf den Zoomslider und hält diese Taste bis auf weiteres gedrückt. Sobald der Nutzer die Taste wieder loslässt, wird der Prozess an dieser Stelle gestoppt und mit Schritt 8 abgeschlossen.
2. Die aktuelle Zoomstufe Z_n wird bestimmt.
3. Alle Kacheln k innerhalb der Zoomstufe Z_n, die ganz oder teilweise im sichtbaren Bereich liegen, und alle bereits vorgeladenen Kacheln, die an diesen Bereich angrenzen, werden bestimmt.
4. Der Nutzer entscheidet sich für eine Zoomrichtung (ZoomIn oder ZoomOut).
5. Nun wird geprüft, ob die Entscheidung von (4) mit der aktuellen Zoomstufe Z_i (beim 1. Durchgang ist $Z_i = Z_n$) vereinbar ist; d. h. wenn $Z_i = Z_{\min}$ bzw. $Z_i = Z_{\max}$ gilt, ist kein ZoomOut bzw. ZoomIn möglich. In diesem Fall muss der Nutzer sich erneut für eine Zoomrichtung entscheiden (Schritt 4). Im anderen Fall geht es mit (6) weiter.
6. Parallel zur Bewegung des Zoomsliders in die unter (4) ausgewählte Richtung wird nun der entsprechende Zoom-Skaliervorgang durchgeführt:
   a) Die neue Zoomsliderposition P_i wird bestimmt.
   b) Daraus wird die neue Zoomstufe $Z_i = Z(P_i)$ berechnet.
   c) Die neue (skalierte) Kachelgröße $K(Z_i)$ der aktuellen Zoomstufe Z_i wird auf Basis von Z_n berechnet. Als Regeln gelten:

$$K(Z_i) = 2^{Z_i - Z_n} * K(Z_n) \quad \text{für } Z_n < Z_i \tag{4.1}$$

$$K(Z_i) = K(Z_n) \quad \text{für } Z_n = Z_i \tag{4.2}$$

$$K(Z_i) = \frac{1}{2^{Z_n - Z_i}} * K(Z_n) \quad \text{für } Z_n > Z_i \tag{4.3}$$

   Ausgehend von der Zoomstufe Z_n wird mit jeder neuen Zoomstufe Z_i die gegebene Kachelgröße $K(Z_n)$ verdoppelt (für $Z_n < Z_i$; vgl. Gleichung 4.1) bzw. halbiert (für $Z_n > Z_i$; vgl. Gleichung 4.3), um die neue Kachelgröße $K(Z_i)$ der aktuellen Zoomstufe Z_i zu erhalten. Bei Zoomstufengleichheit ($Z_n = Z_i$) sind die Kachelgrößen gleich groß (vgl. Gleichung 4.2).

   Anmerkung: Die einzelnen Zoomstufen dienen lediglich als definierte Zwischenwerte. Der Skaliervorgang selbst verläuft komplett stufenlos.

d) Alle momentan geladenen (sichtbaren und nicht sichtbaren) Kacheln werden beim ZoomIn- bzw. ZoomOut-Vorgang auf die errechnete Kachelgröße $K(Z_i)$ skaliert (genauer: vergrößert bzw. verkleinert).

e) **Nur für ZoomOut:** Sobald alle unter (3) bestimmten Kacheln k soweit verkleinert werden, dass sie alle in den sichtbaren Bereich treten, müssen weitere Kacheln im Hintergrund nachgeladen werden, um einen weißen Rahmen beim ZoomOut zu vermeiden. Eine detaillierte Erläuterung, wie mit diesem Tilepreloading beim ZoomOut-Vorgang umgegangen wird, gibt der nachfolgende Abschnitt »Besonderheiten beim ZoomOut«.

7. Solange die linke Maustaste gedrückt wird (siehe Schritt 1), kann der Nutzer sich jederzeit für eine andere Zoomrichtung entscheiden und springt damit wieder zu (4). Lässt der Nutzer die Maustaste los, wird der Zoomvorgang mit (8) beendet.

8. Der sichtbare Bereich wird in der aktuellen Zoomstufe Z_i mit allen dafür benötigten Kacheln in der Standardkachelgröße $K(Z_n)$ neu gezeichnet. Zusätzlich werden (nach Prinzip des Tiling-Verfahrens; vgl. Abschnitt 2.3.5) alle angrenzenden, nicht sichtbaren Kacheln in der gleichen Größe vorgeladen. Der Zoomprozess ist damit beendet.

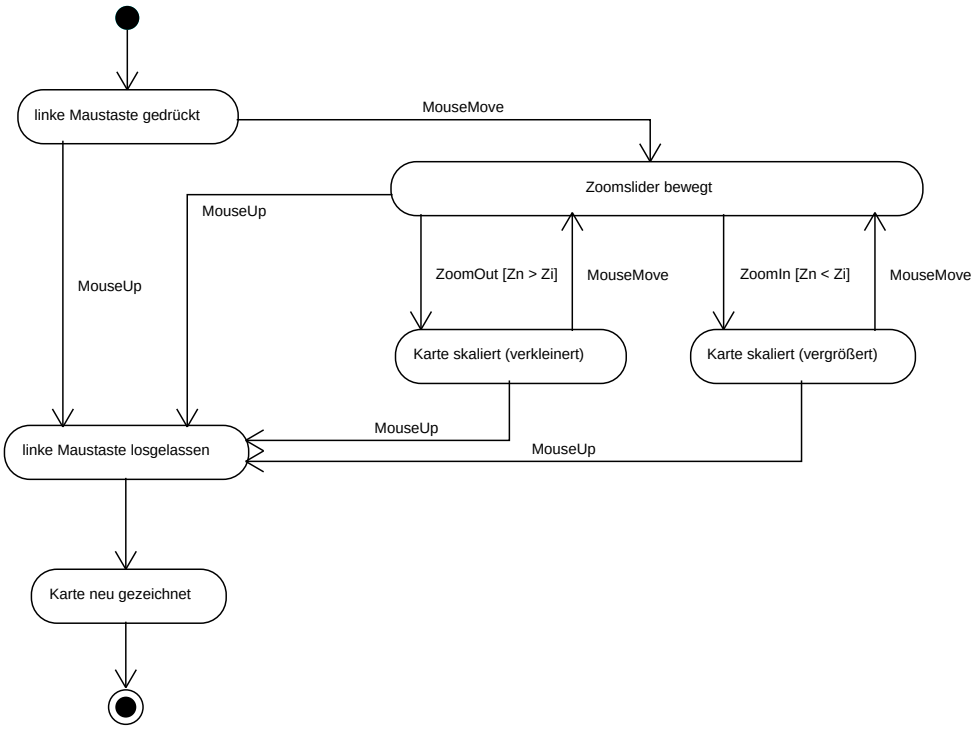


Abbildung 4.1: Zustandsdiagramm des Zoomprozesses bei Nutzung des Sliders

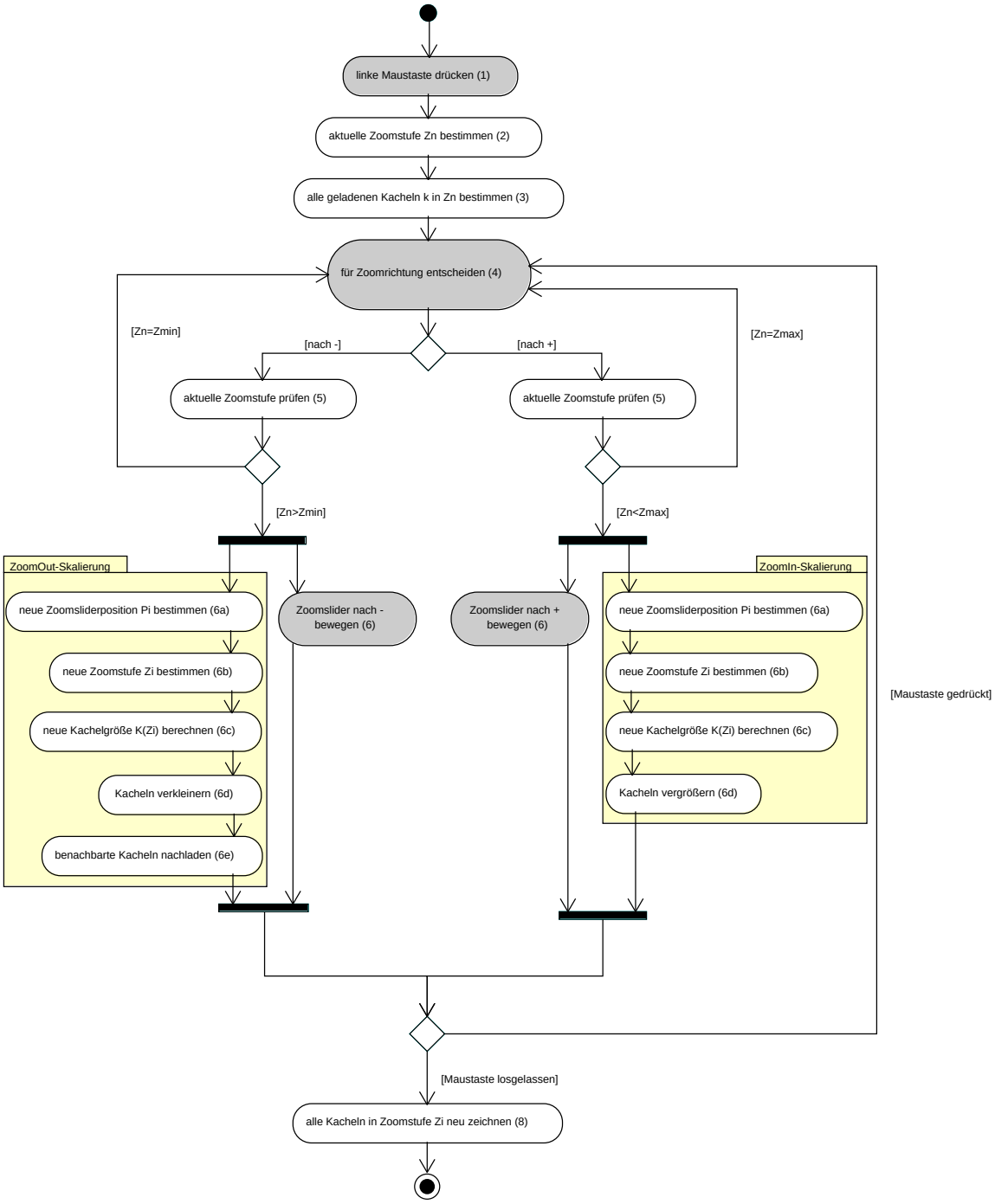


Abbildung 4.2: Aktivitätsdiagramm des Zoomprozesses bei Nutzung des Sliders

Besonderheiten beim ZoomOut

Nachfolgend werden zwei Lösungen vorgestellt, wie im Schritt 6e des ZoomOut-Prozesses mit dem Vorladen der Kacheln umgegangen werden kann. Wichtigstes Ziel dabei ist, dem Benutzer ein möglichst schnelles, verzögerungsfreies Nachladen der Kacheln zu gewährleisten und damit einen weißen Rahmen um die kleiner werdende Karte zu vermeiden. Die zwei unterschiedlichen Lösungsansätze versuchen, dieses Ziel umzusetzen:

A) Sobald eine der vorher nicht sichtbaren Kacheln von k in den sichtbaren Bereich eintritt, werden alle nun nicht sichtbaren Kacheln neu bestimmt. Dabei werden die noch unbekannten Kacheln im Hintergrund in der Zoomstufe Z_n vorgeladen. Sie werden als neue, nicht sichtbare Kacheln in dem DOM-Baum vorgehalten. Darüber hinaus passen sich alle nachgeladenen Kacheln sofort dem momentanen Skalierungsstand an, d. h. sie werden ebenfalls synchron zu den restlichen Kacheln auf die berechnete Kachelgröße $K(Z_i)$ skaliert.

   Nachteil dieser Lösung ist, dass mit stetigem Herauszoomen immer mehr Kacheln nachgeladen und zusammen mit den bisherigen Kacheln skaliert werden müssen. Es ist sehr wahrscheinlich, dass die Performance bei zunehmender Kachelanzahl dadurch stark beeinträchtigt wird.

B) Die Besonderheit der zweiten Lösungsvariante ist eine neu definierte »ZoomOut-Kachel«. Zusätzlich zu den unter Schritt 3 bestimmten Kacheln k wird einmalig beim Initialisieren der Anwendung *eine* weitere Kachel k_* vorgeladen, um den ZoomOut-Prozess durchzuführen. k_* bildet die geografische Ausdehnung der Gesamtkarte auf einer Kachel ab. Die vordefinierte Kachelpixelgröße K_* von k_* sei um den Faktor x größer als die vordefinierte Kachelgröße K_k aller Kacheln k: $K_* = x * K_k$, wobei $x \geq 1$ und $x \in \mathbb{R}$ ist. Ein geeigneter Wert für x ist abhängig von der Gesamtkartengröße sowie der Anzahl der möglichen Zoomstufen und wird in der Realisierung (vgl. Abschnitt 5.2.3) ermittelt. Das Zentrum der Gesamtkarte sei die geometrische Mitte von k_*.

   Ziel dieses Lösungsansatzes ist es, das aufwändige Nachladen der Kacheln aus Lösung (A) beim On-the-fly-Herauszoomen zu reduzieren. Statt dessen wird k_* als eine Art Basiskachel genommen, mit der sich der Großteil der Skalierungsvorgänge durchführen lässt. Dadurch müssen nicht, wie in Lösung A, zahlreiche zum »Abdecken« des weißen Rands benötigte Kacheln nachgeladen und skaliert werden. Zumal die hohe Detailstufe dieser Kacheln für eine ZoomOut-Skalierung (Verkleinern) gar nicht benötigt wird.

   Nachteilig wird sich die starke »Verpixelung« von k_* bei hinreichend weiter Vergrößerung auswirken. Abhängig vom Zoomstufenintervall kann dies dazu führen, dass Kartenausschnitte im hohen Skalierungsprozess nicht mehr identifiziert werden können.

4.3 Technische Rahmenbedingungen

Für die Realisierung des *animated zooming* Features soll die *OpenLayers*-Beispieldemo *controls.html*[1] als Grundlage dienen. Anpassungen seien möglich. Folgende Ebenen (mit ihren Kartenquellen) sollen verwendet werden:

- **WMS-BaseLayer** (default): http://labs.metacarta.com/wms/vmap0
- **WMS Untiled-BaseLayer**: http://labs.metacarta.com/wms/vmap0
- **Image-BaseLayer**: http://earthtrends.wri.org/images/maps/4_m_citylights_lg.gif
- **WMS-Overlay**: http://www2.dmsolutions.ca/cgi-bin/mswms_gmap

Erweiterbarkeit

Die Umsetzung berücksichtigt die spätere Erweiterbarkeit der Anwendung. Insbesondere die unter Abschnitt 3.3.3 genannten Wunschkriterien sollen im konzipierten *animated zooming* Feature mit möglichst geringem Aufwand integriert werden können. Das von *OpenLayers*-Entwicklern realisierte *animated panning* Feature[2] (aktuell noch im Review-Prozess) dient als Bestandteil für die Umsetzung der automatischen Zoomanimation. Der Algorithmus für die zeitabhängige Berechnung von Kraft-Bewegungskurven (beschleunigter Start, gedämpftes Ende) soll so erweitert werden, dass er für die Zoomanimation benutzt werden kann. Auch eine Kombination von Pan- und Zoomanimationsprozessen soll möglich sein.

Aufgrund der verschiedenen von *OpenLayers* unterstützten Ebenentypen (vgl. Abschnitt 3.4.2) ist eine Einzelbehandlung des jeweiligen, unterschiedlichen Skalierungsverhaltens nötig. Gleicher Quellcode kann in die Basisklasse *Layer.js* ausgelagert werden; ebenenspezifisches Zoomverhalten kann durch individuelle (überschriebene) Methoden behandelt werden. Beim Integrieren neuer Ebenentypen muss so nur die neue Ebenenklasse angepasst werden. Eine Änderung der Kernfunktionalitäten ist nicht erforderlich. Damit ist eine leichte Erweiterbarkeit des *animated zooming* Features gewährleistet.

[1] http://www.openlayers.org/dev/examples/controls.html [Abruf: 15.05.2007]
[2] http://trac.openlayers.org/ticket/110 [Abruf: 15.05.2007]

4.4 Entwicklungsrichtlinien von *OpenLayers*

Das *OpenLayers*-Projekt definiert klare Enwicklungsrichtlinien:

Jede Person hat die Möglichkeit, einen gefundenen Fehler (Bug) oder einen neuen Erweiterungswunsch in das Ticket-System[3] von *OpenLayers* einzutragen. Dabei sollen spezielle Regeln[4] befolgt werden. Eine fertige Patchdatei[5], die den Bug behebt oder das Feature implementiert, wird an das erstellte Ticket angefügt; der Ticketstatus wird auf *review* gesetzt. Über die Entwickler-Mailingliste sollte anschließend eine Zusammenfassung der implementierten Änderungen verschickt werden, die bei Bedarf diskutiert und ggf. noch einmal modifiziert werden können. Sind keine Änderungswünsche vorhanden, wird der Patch von einem oder mehreren autorisierten *project committers* geprüft, ggf. geändert und in die aktuelle SVN-Entwicklungsversion, den *Trunk*, integriert [OpenLayers, c].

Registrierte Entwickler haben die Möglichkeit, sich ihre eigene SVN Sandbox für Test- und Demonstrationszwecke einzurichten. Jede Sandbox ist unabhängig vom Trunk. Entwickler haben innerhalb ihrer eigenen Sandbox alle Freiheiten und müssen keine Richtlinien befolgen.

Das *Project Steering Committee (PSC)* von *OpenLayers* besteht aktuell aus sieben Mitgliedern[6]. Sie überwachen den Projektablauf und versuchen, im Sinne der Community Entscheidungsprozesse voranzubringen. Bei wichtigen Entscheidungen – wenn z. B. substantielle Quellcodeänderungen vollzogen, die Rückwärtskompatibilität verändert, neue Veröffentlichungstermine entschieden oder Verfahrensfragen geklärt werden sollen – stimmen die PSC-Mitglieder darüber ab. Jede Person aus der *OpenLayers*-Community kann (über die Entwickler-Mailingliste) einen Antrag stellen oder diesen kommentieren; aber nur die Komitee-Mitglieder sind berechtigt zu votieren. Dabei gelten klare Abstimmungsregeln, die wie folgt zählen [OpenLayers, e]:

`+1` volle Unterstützung des Antrags; Bereitschaft zur aktiven Mitarbeit

`+0` Unterstützung des Antrags; jedoch keine Bereitschaft zur Mitarbeit

`0` keine Meinung

`-0` geringe Nichtübereinstimmung mit dem Antrag; jedoch keine Blockadehaltung

`-1` Veto; Blockierung des Antrags

Ein Antrag ist angenommen, wenn die Gesamtsumme der Abstimmungsergebnisse mindestens +2 beträgt und keiner mit `-1` stimmt. Bei einem Veto muss der Votierende eine klare Begründung geben und eine Alternative zum Lösen des Problems innerhalb von zwei Tagen vorschlagen. Bei Streitfragen entscheidet der Komitee-Vorsitzende [OpenLayers, e].

[3] http://trac.openlayers.org/query [Abruf: 15.05.2007]
[4] http://trac.openlayers.org/wiki/FilingTickets [Abruf: 15.05.2007]
[5] http://trac.openlayers.org/wiki/CreatingPatches [Abruf: 15.05.2007]
[6] http://trac.openlayers.org/wiki/SteeringCommitteeMembers [Abruf: 15.05.2007]

Die Realisierung des *animated zooming* Features wird auf Grundlage der o. g. Entwicklungsrichtlinien durchgeführt. Regelmäßiges Kommunizieren bereitgestellter aktueller Zwischenversionen über die Developer-Mailingliste soll die *OpenLayers*-Community in den Entwicklungsprozess mit einbeziehen. Dadurch können frühzeitig vorgeschlagene Änderungen oder Ideen der Entwickler berücksichtigt werden. Diese Transparenz ermöglicht eine bestmögliche Akzeptanz des Features. An dieser Stelle sei betont, dass das *animated zooming* Feature die bestehende API erweitern wird und keine fundamentalen Änderungen an den Kernkomponenten von *OpenLayers* nötig sind. Ferner stellt die Begutachtung des Quellcodes durch andere Entwickler eine Qualitätssicherung dar.

4.5 Testmethodik

4.5.1 Komponententests

Komponententests unterstützen den Entwicklungsprozess der geplanten Erweiterung. Dabei wird das von *OpenLayers* genutzte *Test.AnotherWay*-Framework eingesetzt (vgl. Abschnitt 3.4.3). Die Unittests sollen die Korrektheit aller Funktionen und Klassen verifizieren und als Bestandteil der Qualitätssicherung dienen. Es ist eine testgetriebene Entwicklung (engl. *test first development*) angestrebt, wobei die Komponententests parallel zum eigentlichen Quelltext enstehen sollen.

Bei der Erstellung von Unittests für das *animated zooming* Feature ist die Besonderheit der automatischen Zoomanimationen zu berücksichtigen: Für eine automatisch ablaufende Animation ist im API-Quellcode eine zeitabhängige Komponente nötig, die diesen Prozess steuert. Die JavaScript-Funktion *window.setInterval* sorgt dabei für eine immer wiederkehrende, zeitlich unbegrenzte zyklische Ausführung einer bestimmten Funktion mit zwischengeschalteten Wartepausen. Mittels *window.clearInterval* lässt sich die Funktion abbrechen. Um solche asynchronen Funktionsaufrufe im Quellcode mit Komponententests zu überprüfen, bietet *Test.AnotherWay* die Methode *delay_call* an, die zwei Argumente unterstützt: die Wartezeit zwischen den Funktionsaufrufen in Sekunden und die auszuführende Funktion. Wird keine Zeit angegeben, wird eine Verzögerungszeit von 0,2 Sekunden genutzt. Die Beispieltestfunktion von Listing 3.2 muss wie folgt für die Zoomanimation abgeändert werden, damit ein erfolgreicher Testdurchlauf möglich ist:

```
11      ...
12              map.zoomTo(0);
13              t.delay_call(1, function() {
14                  t.eq( map.getZoom(), 5, "map.zoom is correct after calling zoomTo" );
15              });
16      ...
```

Listing 4.1: Beispieltestfunktion berücksichtigt asynchrone Funktionsaufrufe

4.5.2 Integrationstests

Nach den Komponententests überprüfen Integrationstests das Zusammenspiel mehrerer (getesteter) Einzelkomponenten. Dies dient der Qualitätssicherung. Ein Testplan beschreibt das Vorgehen ausgewählter Testfälle, die nach Abschluss der Implementierung durchgeführt werden. Der Plan gliedert sich in unterschiedliche Testsuiten, um eine logische Trennung der zu testenden Funktionalitäten vorzunehmen. Jede Testsuite kann separat ausgeführt werden. Sie gliedert sich in verschiedene Arbeitsschritte (Testfälle) und in die dazu erwarteten Resultate, die beim Durchführen der Testfälle geprüft werden. Weicht ein Resultat vom erwarteten Verhalten ab, schlägt die gesamte Testsuite fehl.

Auf Grundlage der Anforderungsanaylse (vgl. Abschnitt 3.3) wird der Testplan für die *animated zooming* Erweiterung erstellt. Eine vollständige Auflistung von allen *möglichen* Anwendungsfällen, die beim Benutzen des Features durchlaufen werden (können), ist für die vorliegende Arbeit zu umfangreich. Der Testplan konzentriert sich daher auf die typischen und extremen Fälle beim Zoomverhalten. Funktionalitäten der Anwendung, die unabhängig von dem implementierten Feature sind, werden im Testplan nicht berücksichtigt. Ihre Korrektheit wird in den Komponententests verifiziert und sollte im Idealfall von der Implementierung nicht beeinträchtigt werden. Um eine Vergleichbarkeit der Tests zu gewährleisten, werden alle Tests mit der *OpenLayers*-Demo *controls.html* durchgeführt. Zu jedem Test werden Datum, Betriebssystem, Browser, Name des Testers, Testdauer und ggf. Kommentare von der Testperson dokumentiert.

Im Anhang C (S. 118 ff.) befindet sich der komplette Testplan für die *animated zooming* und *panning* Erweiterungen. Das *animated panning* Feature wurde in den Testplan mit aufgenommen, weil es (zum Zeitpunkt der Konzeptplanung und Realisierung) den Review-Prozess durch die *OpenLayers*-Entwickler noch nicht abgeschlossen hat. Es wird daher eine eigene Qualitätssicherung durchgeführt, um die Korrektheit der Erweiterung zu verifizieren.

Anmerkung: Im Anschluss an die Realisierung wurde der Plan an die tatsächlich umgesetzten Kriterien aktualisiert, so dass eine erweiterte Version des Testplans entstand. Eine genauere Betrachtung der Implementierung und der damit verbundenen Aktualisierung des Testplans folgt im nächsten Kapitel.

5 Realisierung

Dieses Kapitel widmet sich der Umsetzung des im vorherigen Kapitel beschriebenen Konzepts. Der Realisierungsprozess gliedert sich in Vorbereitung, Implementierung, Tests und Schwierigkeiten.

5.1 Vorbereitung

Zu Beginn wird ein Benutzeraccount und eine eigene SVN Sandbox bei *OpenLayers* eingerichtet[1]. Die Sandbox ist unter http://svn.openlayers.org/sandbox/emanuel/ frei zugänglich. Eine aktuelle Version des Trunks wird in die Sandbox kopiert und dient als Entwicklungsgrundlage. Die Sandbox ermöglicht eine eigene Quellcodeverwaltung und führt nach jedem commit einen automatischen checkout in das öffentlich zugängliche Verzeichnis http://dev.openlayers.org/sandbox/emanuel/ durch. Dadurch lässt sich der bereitgestellte Quellcode für andere Entwickler unmittelbar im Webbrowser testen ohne zuvor einen svn checkout durchzuführen.

Die eigentliche Implementierung erfolgt in einer ausgecheckten Sandbox-Version auf einem Debian GNU/Linux System. Es ist keine Installation nötig; *OpenLayers* läuft ohne Webserver in einem beliebigen lokalen Verzeichnis. Eine Internetverbindung ist für die Entwicklung notwendig, um mit den unter Abschnitt 4.3 genannten Kartenebenen arbeiten zu können. Getestet wird während der Entwicklung mit dem Firefox-Webbrowser. Zusätzlich wird das Firefox-PlugIn *Firebug*[2] genutzt, um JavaScript-Code zu debuggen, einzelne HTML-Div-Elemente zu visualisieren und den DOM-Baum zu überprüfen. Die *OpenLayers*-Beispieldemo wird an die definierten Ebenen angepasst (vgl. Abschnitt 4.3).

5.2 Implementierung

Die nachfolgende Beschreibung der Implementierung teilt sich in aufeinander aufbauende Umsetzungsschritte, die in einzelnen Abschnitten behandelt werden. Die Beschreibung stützt sich auf die Verwendung einer gekachelten WMS-Ebene.

[1] http://trac.openlayers.org/wiki/FrequentlyAskedQuestions [Abruf: 15.05.2007]
[2] http://www.getfirebug.com [Abruf: 15.05.2007]

Die entwickelte *animated zooming* und *panning* Erweiterung wurde vor Abgabe dieser Arbeit auf die *OpenLayers*-Version 2.4 RC5 (vom 25.5.2007) aktualisiert und ist über zwei Medien verfügbar:

- über die o. g. SVN Sandbox (Revision 3197; vom 28.5.2007);
  entweder per checkout: svn co -r3197 http://svn.openlayers.org/sandbox/emanuel/animatedZooming/
  oder online per TracBrowser: http://trac.openlayers.org/browser/sandbox/emanuel/animatedZooming
- und über die beiliegende CD-ROM im Anhang F. Es handelt sich hierbei um eine Kopie der Sandbox Revision 3197.

Alle im folgenden genannten Quellcodeverweise beziehen sich auf die o. g. Revisionsnummer 3197. Die im Root-Verzeichnis liegende Beispieldemo *demo.html* wurde nach Abschnitt 4.3 angepasst und ist direkt von der CD bzw. über die aktuelle Sandbox-Version (http://dev.openlayers.org/sandbox/emanuel/animatedZooming/demo.html) ausführbar.

Das Klassendiagramm im Anhang B markiert alle, für die Umsetzung der *animated zooming* und *panning* Erweiterung, veränderten Klassen farblich. Zusätzlich sind alle editierten Methoden aufgelistet. Das Diagramm basiert ebenfalls auf der *OpenLayers*-Version 2.4 RC5 (Stand: 25.5.2007).

5.2.1 Zoomslider-Events

Das Benutzen des Zoomsliders läuft in der *PanZoomBar.js* in drei (Event-)Schritten ab: *MouseDown – MouseMove – MouseUp*.

Beim *MouseDown*-Event wird die aktuelle Zoomstufe vor Beginn des Zoomprozesses in der *Map*-Objekt-Eigenschaft zoomlevel_startScale gespeichert. Dieser Wert gilt als Berechnungsgrundlage für die Skalierung.

Tritt das *MouseMove*-Event ein (durch Bewegen des Sliders), werden mit jeder Bewegungsänderung folgende drei Werte neu berechnet:

1. **Sliderposition**:
   Die aktuelle Mausposition bildet ein Pixel-Wertepaar, dessen y-Wert die Sliderposition darstellt. Werte, die außerhalb der Zoombar liegen, werden abgefangen (vgl. *zoomBarDrag()*; *PanZoomBar.js*).
2. **Zoomstufe**:
   Die Differenz zwischen Sliderstartposition und aktueller Sliderposition geteilt durch den definierten Sliderraster ergibt die Zoomstufenabweichung. Addiert mit dem Zoomlevelstartwert folgt die neue Zoomstufe zoomlevel_scale (vgl. *calculateNewZoomlevel()*; *Map.js*):

```
var deltaY_zoomlevel = this.zoomStart.y − sliderPosition.y;
this.zoomlevel_scale = this.zoomlevel_startScale + deltaY_zoomlevel/zoomStopHeight;
```

3. **Kachelgröße**:
   Die Kachelgröße verdoppelt bzw. halbiert sich beim Skalieren mit jeder Zoomstufe (vgl. Abschnitt 4.2 und *calculateNewTileSize()*; *Map.js*). Für den ZoomIn-Prozess gilt exemplarisch für die Kachelbreite:

   ```
   this.tileSize_scale.w =
     Math.pow(2, (this.zoomlevel_scale − this.zoomlevel_startScale)) ∗ this.tileSize_startScale.w;
   ```

   Analog gilt für den ZoomOut-Vorgang:

   ```
   this.tileSize_scale.w =
     1 / (Math.pow(2, (this.zoomlevel_startScale − this.zoomlevel_scale))) ∗ this.tileSize_startScale.w;
   ```

Nach Berechnung dieser drei Werte wird jede Kachel auf die neu bestimmte Kachelgröße skaliert (genaue Beschreibung folgt im nächsten Abschnitt).

Nach dem Loslassen der Maustaste (*MouseUp*-Event) rastert der Slider in die nächstliegende ganze Zoomstufenzahl ein. Alle sichtbaren Kacheln werden neu gezeichnet.

5.2.2 Skalierung

Die Skalierung wird nur auf die aktive Basisebene angewandt; andere aktive Ebenen werden während des Skalierungsvorgangs ausgeblendet (vgl. *prepareZoomAnimation()*; *Map.js*). Diese Maßnahme ist notwendig geworden, da bereits bei zwei aktiven Ebenen eine spürbare Performanceverschlechterung zu verzeichnen war. Genaue Messungen sind nicht erfolgt. Es wurde entschieden, zu Gunsten eines schnelleren (und damit gebrauchstauglicheren) Zoom-Skalier-Prozesses auf die Anforderung, mehrere Overlays zu skalieren, zu verzichten.

Zu Beginn des Zoomprozesses muss die Kachel bestimmt werden, die den Mittelpunkt des sichtbaren Bereichs enthält. Liegt das Zentrum auf der Grenze mehrerer Kacheln, wird die erste gefundene Kachel als centerTile definiert (vgl. *getCenterTile()*; *Grid.js*).

Beim Bewegen des Zoomsliders werden die unter Abschnitt 5.2.1 bestimmten Werte genutzt, um zunächst die Zentrumskachel zu skalieren und zu positionieren (vgl. *scaleTileTo()*; *Grid.js*). Die neue Kachelgröße wird als *width*- und *height*-Styleparameter des HTML-imgDiv-Elements der Kachel gesetzt. Der Positionierungsalgorithmus basiert auf dem Strahlensatz. Durch elementargeometrische Streckenverhältnisse zwischen dem Kartenzentrum sowie den Eckpunkten der originalen und der skalierten Kachel lassen sich die neuen top- und left-Pixelpositionen der Kachelgrafik bestimmen.

Auf Grundlage der skalierten Zentrumskachel werden die restlichen (sichtbaren) Kacheln skaliert und neu positioniert.

Zeitgleich mit der Skalierung aktualisiert sich die Übersichtskarte (vgl. *updateOverview()*; *OverviewMap.js*). Der rote Markierungsrahmen wird mit dem Zoomvorgang vergrößert bzw. verkleinert und stellt stets den exakten Ausschnitt der Hauptkarte dar. Die Zoomstufe der

Übersichtskarte passt sich beim Bewegen des Sliders an die aktuelle Hauptkarte an. Dazu werden neu Kacheln für die Referenzkarte geladen. Der Zoomprozess wird dadurch nicht beeinträchtigt.

5.2.3 ZoomOut-Kachel

Beim Initialisieren der aktuellen Basisebene wird eine (um den Faktor x größere) Basiskachel geladen, um einen weißen Rahmen um die kleiner werdende Karte beim ZoomOut-Prozess zu verhinden (vgl. Abschnitt 4.2). Es wird eine Kachel bestimmt, deren Grenzen die Gesamtausdehnung der Karte (*maxExtent*) beinhalten; d. h. die Karte muss auf einer einzigen Kachel abbildbar sein. Beginnend mit der Auflösung von Zoomstufe 0 wird die Auflösung solange verdoppelt (und die Karte dadurch verkleinert), bis die Kachelgrenzen diese erfüllen (vgl. *setZoomOutTile_share*; *Layer.js*).

Als optimaler Kachelgrößenfaktor x hat sich nach einigen Tests der Wert 4 herausgestellt. Die Standardkachelgröße bei *OpenLayers* beträgt 256 Pixel (px). Damit würde die ZoomOut-Kachel mit 1024 px geladen werden. Hintergrund: 2048 px ist in der Regel die von vielen WMS-Servern (u. a. vom MapServer) vordefinierte maximale Kartengröße, die nicht überschritten werden sollte. Je größer die Kachelgröße, desto höher ist die Qualität beim Skalieren. Unter dem Gesichtspunkt der Ladezeit sollte eine möglichst geringe Kachelgröße (und damit ein geringer Faktor x) gewählt werden. Faktor 4 ist bei einer definierten Standardkachelgröße von 512 px noch in dem erlaubten Größenintervall (2048 px) und hat im Normalfall mit 1024 px eine vertretbare Ladezeit. Die Qualität ist mit 1024 px bei geringen Zoomstufenänderungen akzeptabel. Bei großen Zoomänderungen treten jedoch deutliche »Verpixelungen« auf, die auch bei einem höheren Faktor nicht ganz vermeidbar sind.

Um die ZoomOut-Kachel in der definierten Kachelgröße vom WMS-Server anzufordern, wurde die *getURL*-Methode in *WMS.js* um einen Parameter *tileSize* erweitert.

5.2.4 Automatische Zoomanimation

Alle unter Abschnitt 3.3.2 beschriebenen Anforderungen sind zu diesem Zeitpunkt berücksichtigt und (bis auf die Overlay-Skalierung) vollständig umgesetzt. Das *animated zooming* Feature über die Zoomnavigationsleiste wird nachfolgend mit den genannten Wunschkriterien (vgl. Abschnitt 3.3.3) erweitert.

Nach Absprache mit den PSC-Mitgliedern von *OpenLayers* wird der aktuell im *Review*-Status stehende *animated panning* Patch als Grundlage für die zeitabhängige Berechnung der Zoomanimationschritte genutzt. Die *easeInOutZoom*-Methode in der *Util.js* ist das Ergebnis aus der Anpassung der analog aufgebauten *animated panning* Funktion (vgl. Listing 5.1). Dabei ist `delta` die Differenz zwischen den Zoomstufen vor und nach der Animation, `totalsteps` die vordefinierte Anzahl der Animationsschritte, `step` der aktuelle »Schrittzähler« und `power` der vordefinierte Kraftfaktor für die Beeinflussung der Beschleunigungskurve.

```
1 OpenLayers.Util.easeInOutZoom = function(delta, totalSteps, step, power) {
2     var stepVal = Math.pow(((1 / totalSteps) * step), power) * delta;
3     return stepVal;
4 };
```

Listing 5.1: Kraft-Bewegungskurven-Algorithmus für Zoomanimationen; *Util.js*

Nach ausführlichem Testen mit unterschiedlichen Werten für die o. g. vordefinierten Variablen wurden totalsteps = 4 und power = 0.7 als ideale Werte für den Zoomprozess empfunden. Es wurde auf eine schnell ablaufende Zoomanimation geachtet, deren Dauer zwei Sekunden nicht überschreiten soll. Daher wurde eine geringe Schrittanzahl gewählt. Für den Kraftfaktor wurde ein Wert gesucht, der die Zoombewegung zu Beginn in kurzer Zeit beschleunigt und zum Ende allmählich langsamer macht. Die Wartezeit zwischen Benutzerinteraktion und erster *sichtbarer* Zoombewegung ist dadurch sehr gering.

Die unter Abschnitt 4.5.1 beschriebene *window.setInterval*-Methode kommt als »Taktgeber« für die schrittweise ablaufende Animation zum Einsatz und wird in der *zoomSlide*-Methode in der *Map.js* aufgerufen. Durch Klicken auf die Zoombar(-Buttons), Doppelklicken auf die Karte, Bewegen des Mausrades, Aufziehen einer Zoombox oder Drücken der +/− Tasten wird die automatische Zoomanimation von der *zoomTo*-Methode aus gestartet. Ein optionaler Funktionsparameter kann die Animation deaktivieren und einen »sprunghaften« Zoomvorgang ausführen.

Während des Ablaufens einer Zoomanimation ist ein wiederholtes Auslösen einer Animation durch o. g. Interaktionen nicht möglich. Beispielsweise bei mehreren Mausradbewegungen wird nur um eine Zoomstufe gezoomt. Eine Funktionserweiterung ist hier denkbar.

Bei den Zoommöglichkeiten per Doppelklick, Mausrad und Zoombox wird gleichzeitig zur Zoomanimation die Karte animiert zum neuen Mittelpunkt verschoben. Durch die Kombination von *animated zooming* und *panning* laufen parallel zwei unabhängige *window.setInterval*-Methoden ab – eine für das Zooming und eine für das Panning.

5.2.5 Kachelaufbau durch *smooth tile update*

Standardmäßig wird bei *OpenLayers* bei jedem Zoomvorgang die alte Karte komplett gelöscht (ein weißer Hintergrund erscheint) und anschließend, durch allmähliches Nachladen der Kacheln in der neuen Zoomstufe, wieder aufgebaut. Die beim *animated zooming* prognostizierte Verbesserung der Nutzer-Orientierung wäre durch diesen Effekt empfindlich gestört. Es wird daher eine Lösung umgesetzt, welche die skalierte Ebene im Hintergrund solange sichtbar hält, bis (in der Ebene darüber) alle neuen Kacheln im sichbaren Bereich vollständig geladen sind. Dieser Effekt wird in dieser Arbeit mit *smooth tile update* bezeichnet und lässt sich nach der Definition von *Smart Map Browsing* ebenfalls zu dessen Eigenschaften zählen.

Die konkrete Realisierung (vgl. *cloneBaseLayerDiv_ share()*; *Layer.js*):
Zunächst wird die aktive Basisebene mit allen ihren Kacheln geklont (baseLayerDivClone) und zu dem Ebenencontainer hinzugefügt. Der z-Index von baseLayerDivClone wird um eins erniedrigt, damit die Ebene direkt unter der Originalebene liegt. Anschließend muss die ZoomOut-Kachel der Originalebene unsichtbar geschaltet werden, um die darunter liegende geklonte Ebene beim Kachelneuaufbau nicht zu überdecken.

Ein *loadend*-Event in der *Layer.js* wird dann ausgelöst, wenn alle Kacheln der Originalebene im sichtbaren Bereich geladen wurden (vgl. *spiralTileLoad()*; *Grid.js*). Danach wird die geklonte Ebene wieder gelöscht und die Möglichkeit zur erneuten Zoomanimation freigegeben (vgl. *setLoadendVisibility()*; *Layer.js*).

5.2.6 Unterstützte Ebenen

Das *animated zooming* Feature unterstützt mit Abschluss der Implementierung die Ebenen *Image*, *WMS Untiled* sowie alle von *Grid* abgeleiteten Ebenenklassen (*WMS*, *MapServer*, *KaMap*, *TMS* und *WorldWind*; vgl. auch Klassendiagramm im Anhang B). *TMS* und *WorldWind* bieten jedoch keine ZoomOut-Kachel-Unterstützung. Hierfür ist eine ebenenspezifische Anpassung der *getURL*-Methode notwendig (analog zu *WMS.js*).

Aktuell noch nicht *animated zooming* fähig sind *MapServer Untiled* sowie die Ebenen der Drittanbieter (*Google*, *VirtualEarth*, *Yahoo* und *MultiMap*). *Canvas* dient nur als Overlay und wird damit auch nicht unterstützt.

Die unter Abschnitt 4.3 beschriebene Erweiterbarkeit neuer Ebenentypen wird durch die *Layer*-Basisklasse gewährleistet. Wie konzipiert, unterstützen neue Ebenentypen standardmäßig zunächst keine Zoomanimation. Durch das Implementieren der ebenenspezifischen Methoden (*getTileSize()*, *getCenterTile()* und ggf. *cloneBaseLayerDiv()*), wird das Default-Verhalten von *Layer.js* überschrieben.

5.2.7 Animated Panning

Das *animated panning* Feature wird für die Kombination von animierten Zoom- und Panvorgängen genutzt (vgl. Abschnitt 5.2.4). Abgesehen von dem Beheben eines Rundungsfehlers in der *getLonLatFromViewPortPx*-Methode (*Layer.js*) und dem Ergänzen von Unittests wurde an dem vorliegenden *animated panning* Patch nichts verändert.

Die Panning-Tastatursteuerung über die Pfeiltasten wurde nach den *Smart Map Browsing* Eigenschaften (vgl. Abschnitt 3.2.2; Punkt *Zooming/Panning per Tastatur*) erweitert. Danach ist es möglich, die Karte mit den Tasten Pos1, Ende, Bild↑, Bild↓ um jeweils 75% der aktuellen Kartenbreite bzw. -höhe zu verschieben.

5.2.8 Quellcode-Struktur

Die *Map*-Klasse übernimmt als zentrales Objekt in *OpenLayers* die Steuerung der Zoomanimation. Der *animated zooming* Quellcode gliedert sich im Kern in die Funktionen *prepareZoomAnimation()*, *runZoomAnimation()* und *finishZoomAnimation()*. Alle Zoomänderungen laufen über die *zoomTo*-Methode, von wo aus die drei o. g. Animationsfunktionen aufgerufen werden.

Zum Abschluss der Realisierung wurde der gesamte geschriebene bzw. geänderte Quellcode in logisch gruppierte Patches unterteilt. Tabelle 5.1 zeigt die acht erstellten Patchdateien mit der jeweiligen Anzahl der Gesamtzeilen sowie der hinzugefügten und gelöschten Quellcodezeilen. Listing 5.2 veranschaulicht exemplarisch an einem Auszug des Patchfiles *final_animatedZooming_PanZoomBar.patch* das Prinzip der hinzugefügten und gelöschten Zeilen. Die zugehörigen Unitteständerungen (Zeile 1 - 53) werden aus Platzgründen hier nicht dargestellt.

		Zeilen		
Nr.	**Patchdatei**	**gesamt**	**hinzugefügt**	**gelöscht**
1	final_animatedZooming_Core.patch	1883	1293	105
2	final_animatedZooming_LayerImage.patch	154	115	4
3	final_animatedZooming_LayerWMSUntiled.patch	339	1293	105
4	final_animatedZooming_Mouse.patch	325	220	34
5	final_animatedZooming_OverviewMap.patch	329	136	54
6	final_animatedZooming_PanZoomBar.patch	140	77	8
7	animatedPanning#2.patch	384	207	45
8	keyboardDefaults.patch	183	139	10
	Gesamt	**3737**	**2480**	**269**

Tabelle 5.1: Erstellte Patchdateien für *animated zooming* und *panning*

Die acht Patchdateien befinden sich auf der beiliegenden CD und sind zusätzlich unter den folgenden drei Tickets bei *OpenLayers* online verfügbar:

- »animated zooming« (Ticket 442)[3]
- »animated panning« (Ticket 110)[4]
- »new keys for better panning« (Ticket 580)[5]

[3] http://trac.openlayers.org/ticket/442 [Abruf: 15.05.2007]
[4] http://trac.openlayers.org/ticket/110 [Abruf: 15.05.2007]
[5] http://trac.openlayers.org/ticket/580 [Abruf: 15.05.2007]

```
Index: lib/OpenLayers/Control/PanZoomBar.js
===================================================================
--- lib/OpenLayers/Control/PanZoomBar.js    (Revision 2907)
+++ lib/OpenLayers/Control/PanZoomBar.js    (Arbeitskopie)
@@ -152,7 +152,10 @@
             var y = evt.xy.y;
             var top = OpenLayers.Util.pagePosition(evt.object)[1];
             var levels = Math.floor((y - top)/this.zoomStopHeight);
-            this.map.zoomTo((this.map.getNumZoomLevels() -1) - levels);
+
+            if (!this.map.zoomanimationActive)
+                this.map.zoomTo((this.map.getNumZoomLevels() -1) - levels);
+
             OpenLayers.Event.stop(evt);
         },

@@ -165,8 +168,12 @@
             this.map.events.register("mousemove", this, this.passEventToSlider);
             this.map.events.register("mouseup", this, this.passEventToSlider);
             this.mouseDragStart = evt.xy.clone();
-            this.zoomStart = evt.xy.clone();
+            this.map.zoomStart = evt.xy.clone();
             this.div.style.cursor = "move";
+
+            // get and set some settings for zoom animation
+            this.map.prepareZoomAnimation();
+
             OpenLayers.Event.stop(evt);
         },

@@ -186,6 +193,13 @@
                     this.slider.style.top = newTop+"px";
                 }
                 this.mouseDragStart = evt.xy.clone();
+
+                // set current slider position
+                var sliderPosition = new OpenLayers.Pixel(evt.xy.x, evt.xy.y);
+
+                // run zoom animation -> scale tile(s)
+                this.map.runZoomAnimation(this.zoomStopHeight, sliderPosition);
+
                 OpenLayers.Event.stop(evt);
             }
         },
@@ -197,12 +211,18 @@
          */
         zoomBarUp:function(evt) {
             if (!OpenLayers.Event.isLeftClick(evt)) return;
-            if (this.zoomStart) {
+            if (this.map.zoomStart) {
                 this.div.style.cursor="default";
                 this.map.events.unregister("mouseup", this, this.passEventToSlider);
                 this.map.events.unregister("mousemove", this, this.passEventToSlider);
-                var deltaY = this.zoomStart.y - evt.xy.y
-                this.map.zoomTo(this.map.zoom + Math.round(deltaY/this.zoomStopHeight));
+                var deltaY = this.map.zoomStart.y - evt.xy.y;
+
+                // zoom map to new zoomlevel
+                var finalZoomlevel = this.map.zoom + Math.round(deltaY/this.zoomStopHeight);
```

```
113 +
114 +             // finish zoom animation
115 +             this.map.finishZoomAnimation(finalZoomlevel);
116 +
117               this.moveZoomBar();
118               this.mouseDragStart = null;
119               OpenLayers.Event.stop(evt);
120 @@ -211,10 +231,17 @@
121
122       /*
123       * Change the location of the slider to match the current zoom level.
124 +     *
125 +     * @param {float} zoomlevel
126       */
127 -     moveZoomBar:function() {
128 -         var newTop =
129 -             ((this.map.getNumZoomLevels()-1) - this.map.getZoom()) *
130 +     moveZoomBar:function(zoomlevel) {
131 +         // check if zoomlevel is not a number
132 +         if (isNaN(zoomlevel)) {
133 +             zoomlevel = this.map.getZoom();
134 +         }
135 +
136 +         // set new top position
137 +         var newTop = ((this.map.getNumZoomLevels()-1) - zoomlevel) *
138               this.zoomStopHeight + this.startTop + 1;
139           this.slider.style.top = newTop + "px";
140       },
```

Listing 5.2: Patchdatei *final_ animatedZooming_ PanZoomBar.patch* (Auszug)

5.3 Tests

5.3.1 Komponententests

Mit dem Beginn der Umsetzung wurden Komponententests erstellt. Die Besonderheit von asynchronen Funktionsaufrufen bei automatischen Zoomanimationen wurden berücksichtigt (vgl. Abschnitt 4.5.1). Bei der Verwendung der *delay_ call*-Methode sind dabei folgende zwei beachtenswerte Eigenschaften aufgefallen:

1. Innerhalb einer Testfunktion darf nach der *delay_ call*-Methode keine weitere Testaussage folgen, die auf Aussagen und Anweisungen der *delay_ call*-Methode aufbaut. Andernfalls schlägt die Testfunktion fehl.
   Hintergrund: Durch die definierte Verzögerungszeit werden Anweisungen, die nach der *delay_ call*-Methode stehen, vorgezogen und abgearbeitet bevor *delay_ call()* ausgeführt wird. In den erstellten Komponententests wird *delay_ call()* daher stets als letzte Anweisung in Testfunktionen verwendet (vgl. Listing 5.3; Zeile 33/34).
2. Bei Testfunktionen, in denen mehrere automatische Zoomanimationen ablaufen sollen, müssen darauf ebenso viele *delay_ call*-Aufrufe folgen. Nach (1) dürfen diese nicht

nacheinander aufgelistet werden. Eine Lösung wäre, sie ineinander zu verschachteln; d. h. innerhalb eines Aufrufs einen neuen Aufruf ausführen. Eleganter ist es jedoch, die einzelnen Funktionen und deren Verzögerungszeiten in der Parameterliste *einer delay_call*-Methode aufzulisten (vgl. Listing 5.3).

Für nahezu alle neu implementierten bzw. geänderten Funktionen von *OpenLayers* (vgl. Klassendiagramm im Anhang B) wurden passende Komponententests erstellt. Dabei wurden drei Testseiten neu angelegt: *test_WMS_Untiled.html*, *test_MouseDefaults.html* und *test_KeyboardDefaults.html*. Die anderen vorhandenen Testdateien wurden um neue Methoden erweitert. Die *animated zooming* Implementierung machte es erforderlich, weitere Testfunktionen, die Zoomverhalten testen, anzupassen und mit *delay_call*-Methoden zu erweitern.

Zum Abschluß der Implementierung wurden alle Tests erfolgreich durchlaufen. Die Komponententests trugen zur Qualitätssicherung bei und stellen für künftige Erweiterungen die Korrektheit der *animated zooming* und *panning* Features sicher.

```
1  function test_05_Map_center(t) {
2      t.plan(6);
3      map = new OpenLayers.Map('map');
4      var baseLayer = new OpenLayers.Layer.WMS("Test Layer", "http://octo.metacarta.com/cgi-bin/mapserv?",
5          {map: "/mapdata/vmap_wms.map", layers: "basic"} );
6      map.addLayer(baseLayer);
7      var ll = new OpenLayers.LonLat(2,1);
8      map.setCenter(ll, 0);
9      map.zoomIn();
10     t.delay_call(
11         1, function() {
12             t.eq( map.getZoom(), 1, "map.zoom is correct after calling setCenter,zoom in");
13             t.ok( map.getCenter().equals(ll), "map center is correct after calling setCenter, zoom in");
14             // set zoomanimation flag manually,
15             // reason: loadend event in layers.js will not achieved in unittests
16             map.zoomanimationActive = false;
17             map.zoomOut();
18         },
19         1, function() {
20             t.eq( map.getZoom(), 0, "map.zoom is correct after calling setCenter,zoom in, zoom out");
21             map.zoomTo(5);
22         },
23         1, function() {
24             t.eq( map.getZoom(), 5, "map.zoom is correct after calling zoomTo" );
25             map.zoomToMaxExtent();
26         },
27         1, function() {
28             t.eq( map.getZoom(), 2, "map.zoom is correct after calling zoomToMaxExtent" );
29             var lonlat = map.getCenter();
30             var zero = new OpenLayers.LonLat(0, 0);
31             t.ok( lonlat.equals(zero), "map center is correct after calling zoomToFullExtent" );
32         }
33     );
34 }
```

Listing 5.3: mehrere *delay_call*-Aufrufe in einer Testfunktion; *test_Map.html*

5.3.2 Integrationstests

Nach Abschluss der Implementierung wurde der erstellte Testplan an die tatsächlich umgesetzten Funktionalitäten angepasst. Der endgültige Testplan gliedert sich in sieben Testsuiten (vgl. Anhang C, S. 118 ff.).

Der komplette Test wurde in sieben unterschiedlichen Browsern durchgeführt. Dazu wurde jeweils ein Testprotokoll erstellt. Tabelle 5.2 stellt die gestesteten Browserversionen, gegliedert nach den genutzten Betriebssystemen, dar.

Debian GNU/Linux 3.1	**Mac OS 10.4**	**Windows XP**
Firefox 1.5.0.7	Safari 2.0.4	Internet Explorer 7.0
Bon Echo 2.0.0.1[1]	Firefox 2.0.0.2	Firefox 2.0.0.3
	Opera 9.10	

[1] selbstkompilierte Version von Firefox 2.0.0.1

Tabelle 5.2: Getestete Browserversionen

Darüber hinaus wurde überprüft, ob der *KDE Konqueror*[6] (Version 3.5.5 und 3.3.2) tatsächlich nicht von *OpenLayers* unterstützt wird (vgl. auch Seite 90). Die Karte blieb beim Aufruf der Demo leer. Der Grund dafür konnte auch nach einer Anfrage über die Entwickler-Mailingliste von *OpenLayers* nicht beantwortet werden.

Der Testdurchlauf mit dem *Internet Explorer (IE)* 7 deckte drei Probleme auf:

1. Folgende Testplanfrage wurde stets mit *Nein* angekreuzt:
   »Symbolisiert die Statusleiste des Browsers nach vollständigem Laden der gesamten Karte die Beendigung des Ladevorgangs (z. B. durch die Meldung done)?«
2. Das Zoomen über die Tastatur funktionierte im IE nicht.
3. Sobald die PNG-Kachelgrafiken beim Skalieren eine Größe von 32768 Pixeln überstiegen, waren sie in der Karte nicht mehr sichtbar. Zusätzlich ging die CPU-Last sprunghaft in die Höhe. Teilweise reagierte der IE kurzzeitig nicht mehr oder stürzte ganz ab.

Punkt (1) sei vernachlässigbar, da es die Funktionsweise der Anwendung nicht beeinträchtigt. Eine Lösung für dieses browserspezifische Problem wird nicht gesucht.
Problem (2) ist ebenso nur im IE aufgetreten. Durch zwei zusätzliche IE-spezifische keyCodes für die + und – Taste konnte das Problem behoben werden (vgl. *KeyboardDefaults.js*).
Das Kernproblem von Punkt (3), dass ab 32768 Pixel keine Kacheln mehr angezeigt werden, zeigte sich auch in den drei getesteten Browsern unter Mac OS X und dem Firefox unter Windows. Eine Erklärung liegt nahe, dass Grafiken in diesen Browsern nur in einer Größe von maximal 16 Bit dargestellt werden können; d. h. 1 Bit für das Vorzeichen und 15 für die Größenzahl, also $2^{15} = 32768$ px. Die nur im IE aufgetretenden o. g. Performanceprobleme

[6] http://www.konqueror.org [Abruf: 15.05.2007]

sind so gravierend, dass eine Lösung hierfür zwingend erforderlich ist, um eine stabile Anwendung zu gewährleisten. Da der IE Pixelgrößen über 2^{15} anscheinend nicht abfängt, muss dies in der Anwendung ausgeglichen werden. *OpenLayers* prüft nun vor jedem Skalieren der Kacheln in den Methoden *scaleTileTo()*, *scaleTilesOfGrid()* und *scaleZoomOutTile_share()* (in *Layer.js*), ob eine Überschreitung des o. g. Wertes vorliegt. Die Kacheln werden dann nicht mehr weiter skaliert, sondern behalten ihre letzte Kachelgröße bei. Für die Orientierung des Nutzers ist dieser Umstand relativ unbedeutend, da sich die Karte bei einer Größe von 32768 Pixeln bereits in einem sehr verpixelten Zustand befindet. Die Tatsache, dass sich die Karte beim Bewegen des Zoomsliders ab einer gewissen Zoomstufe nicht mehr »mitbewegt«, kann sich jedoch irritierend auf den Nutzer, und damit auf das *Smart Map Browsing*, auswirken.

Die durchgeführten Integrationstests stellen den Abschluß der Realisierungs- und Testphase dar und haben durch das Aufdecken neuer Probleme gezeigt, wie wichtig Testprotokolle für die Qualitätssicherung sind.

5.4 Schwierigkeiten

5.4.1 Performance

Das größte Problem bei der Umsetzung des *animated zooming* Features war und ist die Performance. Je mehr Kacheln skaliert werden, desto langsamer läuft die Skalierung. Aus diesem Grund wurde bereits im frühzeitigen Entwicklungsstatus entschieden auf die gleichzeitige Skalierung aller aktiven Overlays zu verzichten (vgl. Abschnitt 5.2.2). Selbst mit einer Basisebene kann bei einem großflächigen sichtbaren Bereich der Zoomprozess spürbar schwerfälliger ablaufen, als bei einem kleinen Kartenfenster von beispielsweise 512 x 512 Pixel.

Es wurde nach Lösungen gesucht und dabei ein Ansatz zur Optimierung der Leistungsfähigkeit entwickelt: Die Idee ist, die Positions- und Größenänderungen der Kacheln während des Skalierens nicht anhand des CSS-Style-Parameters an jeder Grafik zu ändern, sondern an zentralen CSS-Klassen, die spalten- und zeilenweise den Kacheln zugeordnet sind. Demnach würde jeweils eine CSS-Klasse pro Kachelspalte bzw. -zeile die *left*- bzw. *top*-Position der Kacheln bestimmen. Die Kachelbreite und -höhe wäre für alle Klassen gleich. Wird ein Kachelgitter von 4 x 4 skaliert, wäre eine Änderung an 8 CSS-Klassen nötig – im Vergleich zu 16 CSS-Style-Eigenschaften an jeder Kachel.
Dieses Konzept wurde testweise umgesetzt, anschließend aber wieder verworfen, da es nicht die erhoffte, spürbare Geschwindigkeitsoptimierung brachte. Eine genaue Messung wurde nicht durchgeführt.

Eine zweite Optimierungslösung war erfolgreicher und ist nun Bestandteil der entwickelten Erweiterung: Nur die Kacheln, die ganz oder teilweise im sichtbaren Bereich liegen, werden

skaliert. Alle außerhalb liegenden Kacheln werden für die Skalierung nicht berücksichtig (vgl. *setTilesOutsideInvisible*; *Grid.js*). Beispiel: Standardmäßig legt *Grid.js* bei einem Kartenfenster von 512 x 512 px und einer Kachelgröße von 256 px ein 7 x 6 großes Kachelgitter an. Demnach können maximal 9 Kacheln gleichzeitig im sichtbaren Bereich liegen. Die Kachelanzahl, die skaliert werden muss, reduziert sich mit dieser Lösung von 42 auf maximal 9.

Ein weiteres Performanceproblem wurde beim Durchlaufen des Testplans entdeckt. Der *Internet Explorer* zeigte bei Kachelgrößen über 32768 Pixeln spürbare Geschwindigkeitseinbußen; dies führte teilweise sogar bis zum Absturz des Browsers. Eine Lösung dieses Problems ist im Abschnitt 5.3.2 beschrieben.

5.4.2 Ebenentypen

Die in den Wunschkriterien angestrebte Implementierung des *animtated zooming* Features für alle von *OpenLayers* unterstützten Ebenen konnte nicht vollständig umgesetzt werden (vgl. Abschnitt 5.2.6). Die Schwierigkeit (und der damit verbundene Aufwand), die ebenenspezifischen Besonderheiten zu implementieren, wurde im Konzept unterschätzt. Es wurde sich daher auf die Ebenentypen *Image*, *WMS Untiled* und *Grid* (inkl. Unterebenen) beschränkt.

5.4.3 Komponententests

Schwierigkeiten traten bei der Erstellung von Unittestteilen auf, in denen asynchroner Code getestet werden sollte. Die unter Abschnitt 5.3.1 beschriebenen zwei Eigenschaften von *delay_ call()* sind nicht in [Test.AnotherWay] dokumentiert. Auch ausführliche Recherchen und eine Anfrage über die *OpenLayers*-Entwickler-Mailingliste brachte keine Antwort. Erst zum Ende der Realisierung fand sich die oben erläuterte Lösung. Diese Situation führt u. a. dazu, dass die Komponententests nicht, wie geplant, unmittelbar parallel zur Implementierung entstanden sind.

6 Schlussfolgerung

6.1 Zusammenfassung der Ergebnisse

Im Mittelpunkt der Arbeit stehen drei Schritte: eine Bestandsanalyse von Freien WebMapping-Anwendungen, eine daraus abgeleitete Begriffsdefinition von *Smart Map Browsing* sowie die Implementierung einer ausgewählten *Smart Map Browsing* Eigenschaft.

Die **Bestandsanalyse** untersuchte zwölf ausgewählte WebMapping-Anwendungen nach ihrer Gebrauchstauglichkeit. Wie erwartet, nahm dabei *Google Maps* (als einziger proprietärer Vertreter) eine Vorreiterrolle ein. Auffallend ist das gute Abschneiden von *OpenLayers* in der Analyse.
Allgemein betrachtet lässt sich zusammenfassen, dass die kachelbasierten Anwendungen durch ihre verzögerungsfreie Kartenverschiebung und ihren dynamischen Kachelaufbau in der Nutzbarkeit sehr überzeugen. Kombiniert mit einer Zoomnavigationsleiste bieten diese Applikationen bereits einen deutlichen Usabilityvorteil gegenüber herkömmlichen WebMapping-Anwendungen.
Eine innovative und unter den untersuchten Applikationen bislang einmalige Interaktionserweiterung stellt *p.mapper* mit seinem *continuous zooming* Feature bereit. Darin wird ein großes Potenzial für die Entwicklung der WebMapping-Anwendungen gesehen.
Bei einem Großteil der untersuchten Anwendungen wurden eindeutige Schwächen durch fehlende Zoommöglichkeiten per Mausrad, Doppelklick oder Tastatur festgestellt. Auch das statische Zoomverhalten von vielen Übersichtskarten hinterließ einen negativen Usabilityeindruck.
Vor Abgabe dieser Arbeit wurden alle Untersuchungsergebnisse noch einmal überprüft und teilweise aktualisiert.

Der Begriff ***Smart Map Browsing*** wurde in dieser Arbeit geprägt. Eine Definition wurde auf Grundlage der Usability-ISO-Norm erarbeitet. Die Eigenschaften von *Smart Map Browsing* sind aus den Ergebnissen der Bestandsanalyse abgeleitet worden und fassen den aktuellen Stand der Technik von *Smart Map Browsing* zusammen. Die Eigenschaften sind Vorgaben, wie WebMapping-Anwendungen idealerweise konzipiert und gestaltet werden sollten, um eine hochgradige Gebrauchstauglichkeit zu erreichen. Dazu gehören intuitive, selbsterklärende GUI-Komponenten sowie erwartungskonforme Pan-Zoom-Verhalten nach Interaktionen mit diesen Elementen. Die *Smart Map Browsing* Intensität beschreibt den Grad der Gebrauchstauglichkeit. Erfüllt eine WebMapping-Anwendung alle gelisteten Eigenschaften handelt es sich um eine hochgradige *Smart Map Browsing* Anwendung. Andernfalls, bei

Nutzung weniger Eigenschaften besitzt die Anwendung ein geringes *Smart Map Browsing*. Die bedeutendsten Merkmale von *Smart Map Browsing* wurden in dieser Arbeit herausgestellt und hinsichtlich ihre Potenziale betrachtet. Das Ergebniss: Die Leistungsfähigkeit von (clientseitigem) Tiling kann durch (serverseitiges) Tile Caching spürbar verbessert werden. Ein konkrete Realisierung dessen stellt *TileCache* dar. Wichtig für die Zoomtiefenorientierung ist eine Zoomnavigationsleiste. Neue Möglichkeiten ergeben sich durch die Verwendung von *animated zooming* und *panning*, die bei vielen Interaktionen ein neues »Navigationserlebnis« für den Anwender erkennen lassen. Die Eigenschaften und ihre Potenziale entwickeln sich durch technische Neuerungen stets weiter und prägen die Entwicklung von *Smart Map Browsing*.

Der dritte Schritt dieser Arbeit bestand in der Umsetzung des *Smart Map Browsing* Features ***animated zooming*** in *OpenLayers*. Zusammenfassend betrachtet wurden nahezu alle aufgestellten Muss- und Kann-Anforderungen implementiert; nicht realisiert wurde die Skalierung von aktiven Overlays sowie eine vollständige Unterstützung für alle Ebenentypen. Die Performance machte die Kernschwierigkeit der Realisierung aus. Auf die daraus resultierenden offenen Probleme wird im nächsten Abschnitt eingegangen.
Das entwickelte Feature erweitert die bisherige *OpenLayers*-API. Durch die *Build Profils* von *OpenLayers* ist es möglich ausgewählte Klassen der API in eine einzige JavaScript-Datei einzubinden. Diese Datei lässt sich dann in beliebigen WebMapping-Anwendungen integrieren. Damit ist eine, in der Zielsetzung dieser Arbeit angestrebte, allgemeingültige Lösung erreicht.
Das Ergebnis der Realisierung ist in 8 Patches zusammengefasst und befindet sich aktuell im Review-Prozess durch die *OpenLayers*-Entwickler. Der Patch zur Verbesserung der Panning-Tastatursteuerung ist bereits unverändert angenommen und in der *OpenLayers*-Version 2.4 RC2 im April 2007 veröffentlicht worden. Die Integration von *animated zooming* und *panning* ist für die Version 2.5 geplant, die voraussichtlich im Herbst 2007 erscheinen wird.

6.2 Offene Probleme

Ein grundsätzliches Problem beim Einsatz von Tiling in WebMapping-Anwendungen ist die Tatsache, dass WMS-Server automatisch auf jede angefragte Karte einen Maßstabsbalken, ein vordefiniertes Logo, ein Wasserzeichen oder einen Schriftzug setzen können. Abbildung 6.1 veranschaulicht an zwei exemplarischen WMS-Diensten, wie ein daraus resultierender unschöner Kacheleffekt in *OpenLayers* aussehen könnte. Karten, die mit fremden, sich wiederholenden Details »ergänzt« werden, lenken die Aufmerksamkeit des Benutzers vom Karteninhalt ab und haben damit empfindliche (negative) Auswirkungen auf das *Smart Map Browsing*.
Einen technischen Ansatz, wie mit solchen Darstellungsproblemen umgegangen werden kann, bietet *TileCache*: Durch die Option *Metatile* wird eine große Kachel vom WMS-Server geladen und anschließend in mehrere kleinere, unterteilte Kacheln der WebMapping-Anwendung zur Verfügung gestellt. Ein *Metatile* gliedert sich standardmäßig in 5x5 Kacheln. Unter

Berücksichtigung der empfohlenen maximalen Kachelgröße von 2048px bei WMS-Servern, wäre eine vollständige Abbildung großer Karten mit nur einer Metakachel in der Regel nicht möglich. Somit würden auch mit dieser Lösung noch vereinzelte Wiederholungen des ungewünschten Elements in der Karte auftreten.

Ein zweites generelles Tilingproblem ist die Positionierung von Labels, die über Kachelgrenzen hinausgehen. In Abbildung 6.1 (linkes Bild) werden solche Labels einfach abgeschnitten. Ein eindeutiger Nachteil für die Lesbarkeit der Karte. In der vorliegenden Arbeit wird auf diese Problematik lediglich hingewiesen. Eine Diskussion über mögliche Lösungsansätze würde an dieser Stelle zu weit führen.

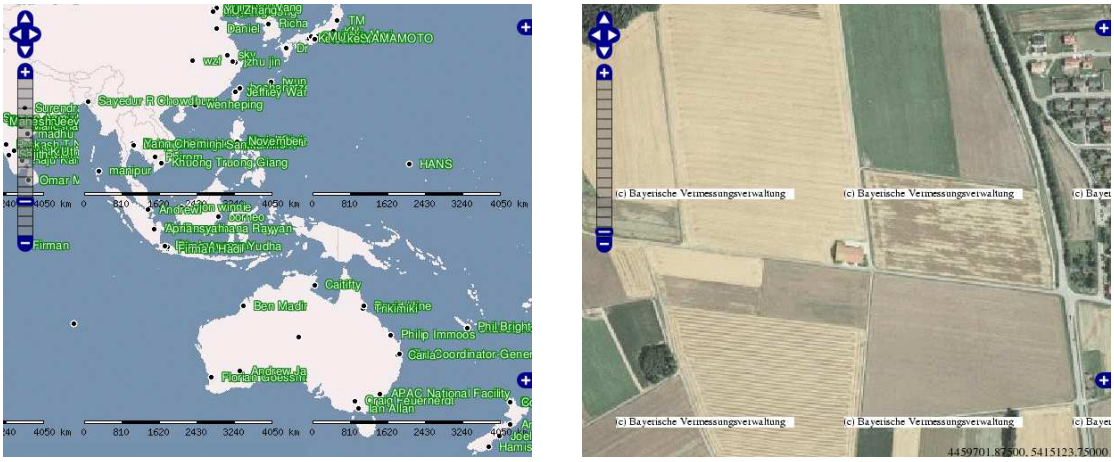


Abbildung 6.1: Tilingprobleme mit Maßstabsbalken, Logos und Labels bei den WMS-Diensten [GRASS user-map] und [GDI Bayern]

Ferner sind bei der entwickelten *OpenLayers*-Erweiterung *animated zooming* zwei offene Probleme festzustellen:

1. Beim Skalieren der Karte treten teilweise starke Qualitätsverluste auf. Die sich vergrößernden Kacheln weisen mit zunehmender Zoomstufe eine Vergröberung der Pixelstruktur auf, die dazu führen kann, dass eine Identifizierung des Karteninhalts kaum noch möglich ist. Insbesondere beim Herauszoomen aus einer hohen Zoomstufe liefert die ZoomOut-Kachel anfangs kaum brauchbare Informationen. Lediglich die sich bewegenden Pixel geben dem Anwender eine Rückmeldung auf seine Zoom-Interaktion. Eine mögliche Lösung wäre, mehrere ZoomOut-Kacheln (basierend auf verschiedenen Zoomstufen) einzusetzen. Sobald die Karte eine bestimmte Zoomstufe übersteigt, könnte so eine weitere ZoomOut-Kachel nachgeladen werden. Diese Variante erfordert jedoch eine erhöhte Lade- und Rechenzeit, was sich negativ auf die Performance auswirken könnte.

2. Aufgrund der bereits mehrmals erwähnten Performance-Schwierigkeiten wurde auf die Overlay-Skalierung verzichtet. Ein Aspekt, der sich nachteilig auf das *Smart Map Browsing* auswirkt. Darüber hinaus erscheint der Ablauf der automatischen Zoomanimation im Vergleich zum *animated panning* noch sehr zähflüssig. Eine stufenlose und in kurzer Zeit ablaufende Zoomanimation wäre wünschenswert und sehr im Sinne der Usability. Die genannten Performanceprobleme zeigen deutlich, dass die rechenaufwändigen Skalierungsprozesse im Browser aktuell noch engen technischen Grenzen ausgesetzt sind. Ein ideales *animated zooming* Verhalten ohne spürbare Verzögerungen scheint durch die derzeitige Client-Hardware nicht möglich zu sein. Durch die zukünftige technische Entwicklung wird sich dieses Problem voraussichtlich in den nächsten Jahren lösen.

6.3 Ausblick

Die vorliegende Arbeit stellt die These auf, dass *Smart Map Browsing* eine verbesserte Benutzerakzeptanz bewirkt. Es wurden Ansätze der heuristischen Evaluationsmethode genutzt, um diese Behauptung anhand einer Usabilityanalyse zu belegen. Ein empirischer Nachweis ist nicht erfolgt, da dies den Rahmen dieser Arbeit überstiegen hätte. Für weiterführende Forschungsaufgaben werden daher ausführliche Usabiltytests vorgeschlagen, die mit Hilfe wissenschaftlicher Methodik die Verbesserung der Gebrauchstauglichkeit von WebMapping-Anwendungen durch die hier definierten Eigenschaften von *Smart Map Browsing* belegen.

Darüber hinaus besteht in der Zukunft weiterer Forschungsbedarf bei der Aktualisierung der *Smart Map Browsing* Eigenschaften. Durch die Abhängigkeit vom aktuellen Stand der Technik sind die definierten Eigenschaften nur begrenzt gültig. Technische Neuerungen machen neue Analysen und Definitionen erforderlich.

Für das *animated zooming* Feature in *OpenLayers* sind für die Zukunft folgende Erweiterungen denkbar:

- Alle Ebenentypen sind vollständig *animated zooming* fähig.
- Ein »Aufaddieren« von mehreren Zoominteraktionen *während* des Ablaufens einer automatischen Zoomanimation bewirkt eine Neuberechnung der Zoomendstufe. Damit wird z. B. beim Mausradbewegen die Anzahl der Radpositionen berücksichtigt.
- Zur Optimierung der Kachelladezeit wird bereits während der automatischen Zoomanimation damit begonnen, die Kacheln für die Zoomendstufe im Hintergrund zu laden. Sollte sich die Zoomendstufe dabei ändern, werden die geladenen Kacheln wieder verworfen.
- Der ZoomReset-Globus aus der vereinfachten Zoombar wird in die Mitte des Pan-Steuerkreuzes integriert. Eine ZoomReset-Möglichkeit besteht nun auch in Kombination mit der erweiterten Zoomnavigationsleiste der PanZoomBar. Anmerkung: Die Zoomnavigationsleiste ist für das manuelle *animated zooming* elementar und sollte anstatt der Mini-Zoombar standardmäßig in *OpenLayers* eingebunden werden.

Betrachtet man die Diskussion um die Zukunft von Freien WebMapping-Anwendungen, so ist es sehr wahrscheinlich, dass *OpenLayers* einmal den funktionellen Kern für einige Map-Applikationen stellen wird – aktuell im Gespräch sind *ka-map*, *Mapbuilder* und *Mapbender*. *OpenLayers* bietet dafür schon heute eine anpassbare *lite*-Version, die definierte *OpenLayers*-Klassen in eine JavaScript-Datei ausgelagert (vgl. Abschnitt 3.4.1 und 6.1). Damit sei eine problemlose Integration in andere Anwendungen möglich.

Mit dieser *lite*-Version ist ein erster Schritt getan, um die Vielzahl der unterschiedlichen Freien WebMapping-Projekte miteinander zu vernetzen und deren spezifische Potenziale effektiv zu nutzen. Die in dieser Arbeit untersuchten *Smart Map Browsing* Eigenschaften von *OpenLayers* könnten so auch in anderen WebMapping-Anwendungen Einzug halten und dadurch deren Usability nachhaltig verbessern.

Literaturverzeichnis

[Asche u. Herrmann 2003] Asche, Hartmut ; Herrmann, Christian: *Web.Mapping 2. Telekartographie, Geovisualisierung und mobile Geodienste.* Heidelberg : Herbert Wichmann Verlag, 2003

[Bernhard u. a. 2005] Bernhard, Lars ; Fitzke, Jens ; Wagner, Roland M.: *Geodateninfrastruktur. Grundlagen und Anwendungen.* Heidelberg : Herbert Wichmann Verlag, 2005

[Bill u. a. 2002] Bill, Ralf ; Seuss, Robert ; Schilcher, Mattäus: *Kommunale Geo-Informationssysteme. Basiswissen, Praxisberichte und Trends.* Heidelberg : Herbert Wichmann Verlag, 2002

[Dickmann 2001] Dickmann, Frank: *Web-Mapping und Web-GIS.* Braunschweig : Westermann Schulbuchverlag GmbH, 2001

[Dickmann 2004] Dickmann, Frank: *Einsatzmöglichkeiten neuer Informationstechnologien für die Aufbereitung und Vermittlung geographischer Informationen – das Beispiel kartengestützter Online-Systeme.* Göttingen : Verlag Erich Goltze GmbH & Co. KG, 2004 http://webdoc.sub.gwdg.de/diss/habil/2004/dickmann/dickmann.pdf

[Ebinger u. Skupin 2007] Ebinger, Samara ; Skupin, Andre: Comparing Different Forms of Interactivity in the Visualization of Spatio-Temporal Data. In: *Kartographische Nachrichten* (2007), Nr. 2, S. 63–70

[Erstling u. Simonis 2005] Erstling, Reinhard ; Simonis, Ingo: Web Map Service. In: *G*eodateninfrastruktur. Grundlagen und Anwendungen **[Bernhard u. a., 2005]**, S. 108–125

[FIT 2005] FIT, Fraunhofer-Institut für Angewandte Informationstechnik: Willkommen im Usability Begriffszoo. (2005). http://www.fit-fuer-usability.de/1x1/basics/begriffszoo.html, Abruf: 15.05.2007

[Fitzke 1999] Fitzke, Jens: GIS im Internet – aus »Das GIS Tutorial«. (1999). http://www.giub.uni-bonn.de/gistutor/internet/inetgis/welcome.htm

[FSF Europe] FSF Europe: Was ist Freie Software? http://www.fsfeurope.org/documents/freesoftware.de.html, Abruf: 15.05.2007

[GDI Bayern] GDI Bayern: WMS GetCapabilities; Layer: DOP. http://deutschlandviewer.bayern.de/ogc/getogc.cgi?REQUEST=GetCapabilities, Abruf: 15.05.2007

[GNU-Projekt a] GNU-Projekt: Warum »Freie Software« besser ist als »Open Source«. http://www.gnu.org/philosophy/free-software-for-freedom.de.html, Abruf: 15.05.2007

[GNU-Projekt b] GNU-Projekt: Was ist das Copyleft? http://www.gnu.org/copyleft/copyleft.de.html, Abruf: 15.05.2007

[GRASS user-map] GRASS user-map: WMS GetCapabilities. http://mapserver.gdf-hannover.de/cgi-bin/grassuserwms?REQUEST=GetCapabilities, Abruf: 15.05.2007

[Grassmuck 2004] Grassmuck, Volker: *Freie Software – Zwischen Privat- und Gemeineigentum*. 2., korrigierte Auflage. Bonn : Bundeszentrale für politische Bildung, 2004

[heise open] heise open: EU-Studie: Open Source ist gut für die Wirtschaft. http://www.heise.de/open/artikel/83795, Abruf: 15.05.2007

[Joos 2003] Joos, Gerhard: Das Web-Mapping-Testbed des Open-GIS-Konsortiums. In: *W*eb.Mapping 2. Telekartographie, Geovisualisierung und mobile Geodienste **[Asche u. Herrmann, 2003]**, S. 181–189

[Klauer 2002] Klauer, R. H.: Potenziale und Techniken eines kommunalen Interneteinsatzes. In: *K*ommunale Geo-Informationssysteme. Basiswissen, Praxisberichte und Trends **[Bill u. a., 2002]**, S. 296–297

[Kunze 2006] Kunze, Ralf: SVGWeather – Entwicklung einer SVG Web Mapping Applikation zur Visualisierung von vierdimensionalen Daten am Beispiel von Wettervorhersagedaten. (2006). http://elib.ub.uni-osnabrueck.de/publications/diss/E-Diss603_thesis.pdf, Abruf: 15.05.2007

[Langfeld 2006] Langfeld, Dorothee: Entwicklung einer SVG Web Mapping Applikation zur Visualisierung von Geoinformationen. (2006). http://www.inf.uos.de/prakt/pers/dipl/dlangfel.pdf, Abruf: 15.05.2007

[MetaCarta] MetaCarta: TileCache. http://www.tilecache.org, Abruf: 15.05.2007

[Mitchell 2005] Mitchell, Tyler: *Web Mapping Illustrated*. Sebastopol (USA) : O'Reilly Media, 2005

[Nielsen 2000] Nielsen, Jakob: *Designing Web Usability: The Practice of Simplicity*. Indianapolis, USA : New Riders Publishing, 2000

[OGC] OGC: About OGC. http://www.opengeospatial.org/ogc, Abruf: 15.05.2007

[OGC 2002] OGC: Web Map Service Implementation Specification. (2002). http://portal.opengeospatial.org/files/?artifact_id=1081&version=1&format=pdf, Abruf: 15.05.2007

[OpenLayers a] OpenLayers: Build Profiles. http://trac.openlayers.org/wiki/Profiles, Abruf: 15.05.2007

[OpenLayers b] OpenLayers: Custom Query – Ticket-System. http://trac.openlayers.org/query?order=priority, Abruf: 15.05.2007

[OpenLayers c] OpenLayers: How to Contribute to OpenLayers? http://trac.openlayers.org/wiki/HowToContribute, Abruf: 15.05.2007

[OpenLayers d] OpenLayers: OSGeo Project Incubation Questionnaire. http://trac.openlayers.org/wiki/IncubationQuestionnaire, Abruf: 15.05.2007

[OpenLayers e] OpenLayers: Project Steering Committee. http://trac.openlayers.org/wiki/SteeringCommittee, Abruf: 15.05.2007

[OpenLayers f] OpenLayers: Why OpenLayers? http://trac.openlayers.org/wiki/BusinessCase, Abruf: 15.05.2007

[OpenLayers g] OpenLayers: Writing Unit Tests. http://trac.openlayers.org/wiki/WritingUnitTests, Abruf: 15.05.2007

[OSGeo 2006a] OSGeo: FOSS4G 2006 Webmap BOF. (2006). http://wiki.osgeo.org/index.php/FOSS4G_2006_Webmap_BOF, Abruf: 15.05.2007

[OSGeo 2006b] OSGeo: OpenLayers/ka-Map Merging BOF Results. (2006). http://lists.osgeo.org/pipermail/webmap-discuss/2006-September/000118.html, Abruf: 15.05.2007

[Peterson 2003] Peterson, Michael P.: Webmapping at the start of the new Millennium – state of the art. In: *W*eb.Mapping 2. Telekartographie, Geovisualisierung und mobile Geodienste **[Asche u. Herrmann, 2003]**, S. 3–17

[Pichler u. Klopfer 2005] Pichler, Günther ; Klopfer, Martin: Spezifikation und Standardisierung – OGC, OGC Europe und ISO. In: *G*eodateninfrastruktur. Grundlagen und Anwendungen **[Bernhard u. a., 2005]**, S. 9–17

[Plümer u. Asche 2004] Plümer, Lutz ; Asche, Hartmut: *Geoinformation – Neue Medien für eine neue Disziplin*. Heidelberg : Herbert Wichmann Verlag, 2004

[Rampl] Rampl, Hansjörg: Begriffsdefinition Usability. http://www.handbuch-usability.de/begriffsdefinition.html, Abruf: 15.05.2007

[Ramsey 2006] Ramsey, Paul: Mashing up the Enterprise. In: *Geospatial Solutions* (2006). http://www.refractions.net/white_papers/mashups, Abruf: 15.05.2007

[Reiter 2004] Reiter, Bernhard: Wandel der IT: Mehr als 20 Jahre Freie Software. In: *HMD – Praxis der Wirtschaftsinformatik* (2004), 83–91. http://www.intevation.de/~bernhard/publications/200408-hmd/200408-wandel_der_it_20j_fs.html, Abruf: 15.05.2007

[Räber u. Jenny 2003] Räber, Stefan ; Jenny, Bernhard: Karten im Netz – ein Plädoyer für mediengerechte Kartengrafik. In: *W*eb.Mapping 2. Telekartographie, Geovisualisierung und mobile Geodienste **[Asche u. Herrmann, 2003]**, S. 57–76

[Römeling 2003] Römeling, Nils: Usability im www. Redesign eines Webauftritts mit Hilfe von Usability Testing. Kapitel 1: Usability & Design. (2003). http://www.usability-diplomarbeit.de/usability/diplomarbeit/kapitel-usability/, Abruf: 15.05.2007

[Simonis u. Merten 2004] Simonis, Ingo ; Merten, Stephan: Die Geodateninfrastruktur des Webportals »geoinformation.net«. In: *G*eoinformation – Neue Medien für eine neue Disziplin **[Plümer u. Asche, 2004]**, S. 79–88

[Test.AnotherWay] Test.AnotherWay: Dokumentation. http://straytree.com/TestAnotherWay/doc/doc.html, Abruf: 15.05.2007

[van der Vlugt u. Stanley 2005] Vlugt, Maurits van d. ; Stanley, Ian: Trends in Web Mapping: It's all about usability. (2005). http://www.directionsmag.com/article.php?article_id=1988, Abruf: 15.05.2007

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BSD	Berkeley Software Distribution
CSS	Cascading Style Sheets
CVS	Concurrent Versions System
DOM	Document Object Model
FS	Freie Software
FSF	Free Software Foundation
GIF	Graphics Interchange Format
GIS	Geografisches Informationssystem
GPL	GNU General Public License
HTML	HyperText Markup Language
JPG	Joint Photographic Experts Group
LGPL	GNU Lesser General Public License
ML	Mailingliste
OGC	Open Geospatial Consortium
OSGeo	Open Source Geospatial Foundation
PNG	Portable Network Graphics
PSC	Project Steering Comittee
SVG	Scalable Vector Graphics
SVN	Subversion
TMS	Tile Map Service
WebGIS	Webbasiertes Geografisches Informationssystem
WFS	Web Feature Service
WMS	Web Map Service
WMS-C	Web Map Service-Cached

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang

Anhangsverzeichnis

A Bestandsanalyse

A.1 Frei & clientseitig

Freie Webmapping-Anwendungen mit 100% JavaScript

A.1.1 OpenLayers

Stand: 15.05.2007

1. Allgemein

Name	**OpenLayers**
URL	
Home	http://www.openlayers.org
Dokumentation	http://trac.openlayers.org/wiki/Documentation
Download	http://www.openlayers.org/download
Live-Demo	http://openlayers.org/gallery und http://www.openlayers.org/dev/examples
aktuelle Version	2.3 (stable)
letztes Update	21.2.2007
Lizenz	BSD
entwickelt von	MetaCarta, USA
Kurzbeschreibung	OpenLayers ist ein Freie (reine) JavaScript-API um dynamische Karten in beliebigen Webseiten zu integrieren.

2. System

Architektur	clientseitig
Programmiersprache	JavaScript
Voraussetzungen	-
unterstützte Browser	Mozilla 1.8; Firefox 1.0+; IE 6.0+; Safari 2.0+; Opera 9.0+; Netscape X.X
ggf. Integration anderer Software	benutzt Prototype und Komponenten von Rico

3. Community

Revisionsverwaltung	SVN (svn checkout http://svn.openlayers.org/trunk/openlayers/)
Mailing Listen (URL)	http://openlayers.org/mailman/listinfo
Entwickler-ML	
Mails je Monat[1]	85
aktive Entwickler insgesamt[2]	72
Anwender-ML	
Mails je Monat[1]	197
aktive Anwender insgesamt[2]	150
kommerzieller Support	MetaCarta, USA

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	keine Installation nötig (API); API-Referenz; Quick Tutorial (gute Basics, http://www.openlayers.org/QuickTutorial)

5. Usability

Alle Untersuchungen beziehen sich auf die Demo http://www.openlayers.org/dev/examples/controls.html

Usability – Gesamteindruck	grundsätzlich erwartungskonform; intuitiv; GUI schlicht & klar; minimierbare Ebenenübersicht und Übersichtskarte schaffen eine aufgeräumte Oberfläche; auffallend schnelles Nachladen der Kacheln – komfortables Navigieren

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	alle Elemente/Werkzeuge werden auf der Karte platziert
Übersichtskarte	Minimierbar; dynamische Zoomstufe; Ausschnitt verschiebbar per Drag&Drop bzw. Mausklick; zentrieren des Ausschnittes nach Verschieben fehlt!; passt sich nicht an Stil der Hauptkarte an (bildet nur Basisebene ab)
Ebenenübersicht	Minimierbare Ebenendarstellung; eingeteilt in Base Layers und Overlays; de-/aktivierbar durch Radiobuttons und Checkboxen
Legende	-
Maßstab/-sbalken	einfache Maßstabsanzeige möglich (in Demo nicht vorhanden)
Werkzeugleiste	keine klassische Werkzeugleiste; sehr minimalistisch; nur zoomBox- und pan-Button; fehlender zoomOut-Button anfangs irritierend; aktives Werkzeug nicht sofort erkennbar (schlechter Farbunterschied); Werkzeugleiste nicht wirklich notwendig
Zoomnavigationsleiste	halbtransparent in der Karte integriert
Pannavigationssteuerkreuz	oberhalb der Zoombar
Zooming allgemein	Wechsel zwischen Pan- und Zoommodus nicht notwendig; max/min-Zoomstufe nur an Zoomnavigationsleiste erkennbar; zurücksetzen der Karte auf Defaultzoomwert fehlt bei dieser Demo (bei anderen Beispielen über 'Weltkugel' in einer kleineren Zoombar)
Zooming per Doppelklick	ja
Zooming per Mausrad	ja
Zooming per Zoombox	ja (auch mit gedrückter Shifttaste möglich; farbige Unterlegung des aufgezogenen Bereichs)
Panning allgemein	freies, stufenloses panning per Drag&Drop; Nachladen der Karte im Hintergrund bewirkt absolut verzögerungsfreies(!) Bewegen der Karte
Zooming/panning über Tastatur	ja
Tiling	ja

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo http://www.openlayers.org/dev/examples/controls.html

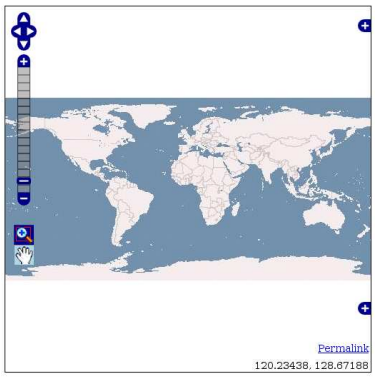
Analysefunktion	-
Suchfunktion	-
Hilfefunktion	-
Druckfunktion	-

7. Bemerkungen

- objektorientierte JavaScript Bibliothek (API)
- sehr leichte Integration in eigene Website
- *lite*-Version (bündelt gewünschte Klassen zu einer js-Datei, dadurch Integration in andere WebMapping-Anwendungen möglich)
- TileCache maßgeblich von OL vorangetrieben
- Einbinden von speziellen Ebenen möglich (GoogleMaps, ka-map, Yahoo, Virtual Earth uvm.); Map24-Support in Entwicklung

8. Screenshot

der untersuchten Demo



A.1.2 WMS Mapper

Stand: 15.05.2007

1. Allgemein

Name	**WMS Mapper**
URL	
Home	http://wms-map.sourceforge.net
Dokumentation	Siehe Home
Download	Siehe Home
Live-Demo	Siehe Home
aktuelle Version	0.03
letztes Update	k. A.
Lizenz	Academic Free License (AFL), Artistic License
entwickelt von	k. A.
Kurzbeschreibung	WMS Mapper ist eine schlanke JavaScript Bibliothek für Einbindung von WMS Diensten mit einfachen Zoom und Verschieben Funktionalitäten.

2. System

Architektur	clientseitig
Programmiersprache	JavaScript
Voraussetzungen	-
unterstützte Browser	k. A.
ggf. Integration anderer Software	Prototype

3. Community

Revisionsverwaltung	-
Mailing Listen (URL)	-
Entwickler-ML	-
Mails je Monat[1]	
aktive Entwickler insgesamt[2]	
Anwender-ML	-
Mails je Monat[1]	
aktive Anwender insgesamt[2]	
kommerzieller Support	k. A.

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	keine Installation nötig; Dokumentation nicht vorhanden - lediglich kurzes Einführungsbeispiel auf der Startseite

5. Usability

Alle Untersuchungen beziehen sich auf die Demo der Startseite

Usability – Gesamteindruck	schlicht, nur 2 Zoom-Buttons, intuitives Navigieren, Zoomtiefenorientierung und weitere Interaktionsmöglichkeiten fehlen

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	schlicht, gut in Website integrierbar
Übersichtskarte	-
Ebenenübersicht	-
Legende	-
Maßstab/-sbalken	-
Werkzeugleiste	keine klassische Werkzeugleiste; sehr minimalistisch; nur ZoomIn- und ZoomOut-Button; Klick auf Button bewirkt Zoomvorgang - somit keine Buttonaktivität nötig
Zoomnavigationsleiste	-
Pannavigationssteuerkreuz	-
Zooming allgemein	Wechsel zwischen Pan- und Zoommodus nicht notwendig; max/min-Zoomstufe nicht erkennbar; zurücksetzen der Karte auf Defaultzoomwert fehlt; zoomen nur mit Buttons; kein Zoomen auf bestimmten Punkt möglich
Zooming per Doppelklick	-
Zooming per Mausrad	-
Zooming per Zoombox	-
Panning allgemein	freies, stufenloses panning per Drag&Drop; Nachladen der Karte im Hintergrund bewirkt nahezu verzögerungsfreies Bewegen der Karte; bei großen Verschiebungsschritten reichen die vorgeladenen Kacheln nicht aus
Zooming/panning über Tastatur	-
Tiling	ja

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo der Startseite

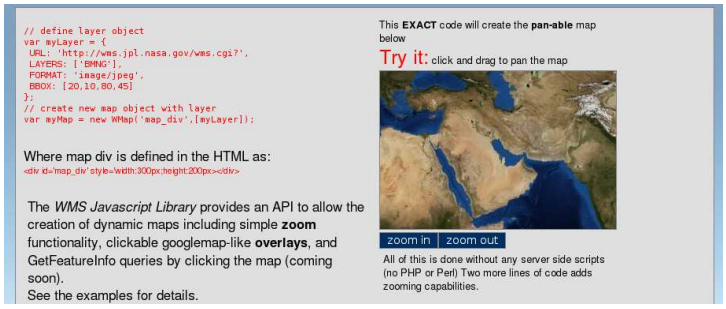
Analysefunktion	-
Suchfunktion	-
Hilfefunktion	-
Druckfunktion	-

7. Bemerkungen

- JavaScript API
- leicht in Websites integrierbar
- GetFeatureInfo folgt demnächst

8. Screenshot

der untersuchten Demo



A.2 Frei & client-serverseitig

Freie Webmapping-Anwendungen mit integrierter Serverkomponente

A.2.1 CartoWeb

Stand: 15.05.2007

1. Allgemein

Name	**CartoWeb**
URL	
Home	http://cartoweb.org
Dokumentation	http://cartoweb.org/documentation.html
Download	http://cartoweb.org/downloads.html
Live-Demo	http://cartoweb.org/demo.html
aktuelle Version	3.3.0
letztes Update	31.08.2006
Lizenz	GNU GPL
entwickelt von	Camptocamp SA, Schweiz
Kurzbeschreibung	CartoWeb ist ein umfangreiche Webmapping-Anwendung um aufwändigere oder spezielle Anwendungen zu erstellen. Eine Verwendung als SOAP Web Service ist ebenfalls möglich.

2. System

Architektur	client-serverseitig
Programmiersprache	PHP 5
Voraussetzungen	◦ Webserver (empfohlen: Apache) ◦ PHP 5.0.3+ ◦ UMN MapServer 4.4+ und PHP/MapScript ◦ PostgreSQL/PostGIS (optional)
unterstützte Browser	kompatibel: Mozilla 1.7+, Firefox 1.07+, IE 6+, Safari 1.3.2+ inkompatibel: Opera, IE 5 (Mac), Konqueror
ggf. Integration anderer Software	basiert auf UMN MapServer

3. Community

Revisionsverwaltung	CVS (cvs -d :pserver:anonymous@dev.camptocamp.com:/var/lib/cvs/ public co cartoweb3)
Mailing Listen (URL)	http://cartoweb.org/contact.html
Entwickler-ML	
Mails je Monat[1]	16
aktive Entwickler insgesamt[2]	12
Anwender-ML	
Mails je Monat[1]	80
aktive Anwender insgesamt[2]	86
kommerzieller Support	k. A.

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	sehr gut und umfangreich (250 S.), User manual und Developer manual (inkl. Installationsanleitung); Workshop: GettingStarted + Material

5. Usability

Alle Untersuchungen beziehen sich auf die Demo: http:// cartoweb.org/ demos/ demoCS.php

Usability – Gesamteindruck	wirkt recht statisch; auffallende Aktualisierung von Karte, Übersichtskarte und Legende nach jeder Änderung; 'Loading'-Meldung fordert viel Geduld

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	Kartengröße einstellbar per Combobox, aktuelle Koordinaten des Steuerkreuzes im Kartenrand
Übersichtskarte	konstante Zoomstufe, dadurch keine Zentrierung nötig; Ausschnitt nur durch Klick verschiebbar
Ebenenübersicht	kombiniert mit Legende; de-/aktivierbar durch Checkboxen; thematisch gegliedert und zusammenklappbar; viele Informationen: Verschachtelungsgefahr; nach Änderung muss Aktualisierungsbutton gedrückt werden
Legende	kombiniert mit der Ebenendarstellung; zusammenklappbar, Nachteil: wirkt sehr verschachtelt, mühsam alle Legendenteile aufzuklappen, Orientierung geht etwas verloren, teilweise nicht sofort ersichtlich, was ist Ebene und was nur Legende
Maßstab/-sbalken	Balken unter der Karte, zusätzlich Maßstabsauswahlfeld vorhanden
Werkzeugleiste	Übliche Funktionen, auffallend: Zeichenwerkzeuge
Zoomnavigationsleiste	-
Pannavigationssteuerkreuz	im Kartenrand
Zooming allgemein	max/min-Zoomstufe nur an Maßstabscombobox/-leiste erkennbar, Zoomlupe weiterhin aktiv
Zooming per Doppelklick	-
Zooming per Mausrad	-
Zooming per Zoombox	ja (auch mit gedrückter Shifttaste möglich; farbige Unterlegung des aufgezogenen Bereichs)
Panning allgemein	freies, stufenloses panning per Drag&Drop; zusätzlich schrittweise über Navigationspfeile
Zooming/panning über Tastatur	-
Tiling	-

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo: http://cartoweb.org/demos/demoCS.php

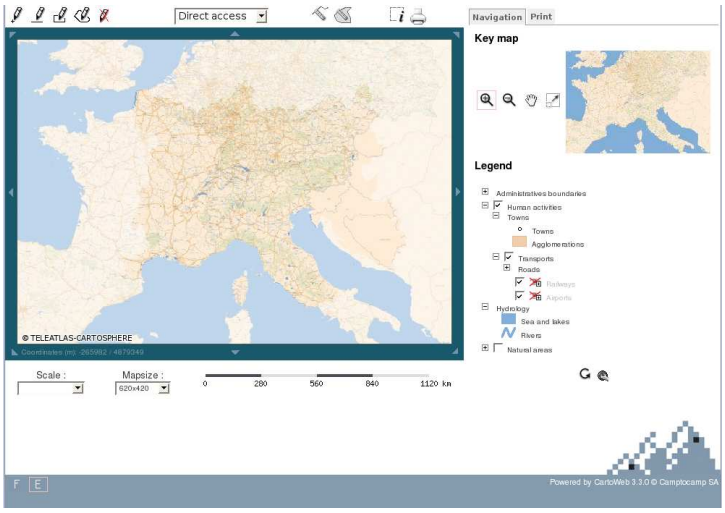
Analysefunktion	Entfernungs-, Flächenberechnungs- und Zeichenfunktion, Sachdatenabfrage
Suchfunktion	-
Hilfefunktion	-
Druckfunktion	sehr gut: realisiert als pdf-Export, mit Formateinstellungen, Titel, Beschreibung, Legendenposition und Referenzkarte

7. Bemerkungen

-

8. Screenshot

der untersuchten Demo



A.2.2 Chameleon

Stand: 15.05.2007

1. Allgemein

Name	**Chameleon**
URL	
Home	http://chameleon.maptools.org
Dokumentation	http://chameleon.maptools.org/index.phtml?page=docs.html
Download	http://chameleon.maptools.org/index.phtml?page=downloads.html
Live-Demo	http://www.mapsherpa.com/hawaii
aktuelle Version	2.4.1
letztes Update	06.09.2006
Lizenz	X11-Style
entwickelt von	DM Solutions Group, Kanada
Kurzbeschreibung	Chameleon ist eine verteilte, umfangreich konfigurierbare PHP-Umgebung für die Entwicklung von WebMapping-Anwendungen. Es basiert auf OGC-Standards für Web Mapping Services (WMS) und WMT Viewer Contexts. Chameleon erlaubt es, schnell eine neue Anwendung aus vorbereiteten Widgets zusammenzusetzen. Das System kann um eigene Widgets erweitert werden.

2. System

Architektur	client-serverseitig
Programmiersprache	PHP
Voraussetzungen	∘ PHP 4.3.x+ ∘ UMN MapServer 4.0+ und PHP/MapScript
unterstützte Browser	k. A.
ggf. Integration anderer Software	basiert auf UMN MapServer

3. Community

Revisionsverwaltung	CVS (http://chameleon.maptools.org/index.phtml?page=cvs.html)
Mailing Listen (URL)	nur eine ML für User+Dev: http://chameleon.maptools.org/index.phtml?page=mailinglist.html
Entwickler-ML	
Mails je Monat[1]	-
aktive Entwickler insgesamt[2]	-
Anwender-ML	
Mails je Monat[1]	51
aktive Anwender insgesamt[2]	63
kommerzieller Support	DM Solutions Group, Kanada

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	DeveloperGuide, InstallationsGuide, JavaScriptAPI, Widgets-Doku – alles sehr ausführlich, mit Beispielen

5. Usability

Alle Untersuchungen beziehen sich auf die Demo http://www.mapsherpa.com/hawaii

Usability – Gesamteindruck	Mauscursor passt sich nicht gewählten Funktionen an – irritierend; keine Rückmeldung über Ladezeitstatus während der Aktualisierungen; endlose max/min-Zoomtiefe irritierend; vordefinierte Kartenbereiche wählbar; übersichtliche Ebenendarstellung, nur manuelle Aktualisierung

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	Kartengröße per Auswahlfeld einstellbar, vordefinierte Kartenausschnitte
Übersichtskarte	konstante Zoomstufe, dadurch keine Zentrierung nötig; Ausschnitt nur durch Klick verschiebbar
Ebenenübersicht	de-/aktivierbar durch Checkboxen; thematisch gegliedert; nach Änderung muss Aktualisierungsbutton gedrückt werden; nicht zusammenklappbar: sehr lange Liste erfordert viel scrollen
Legende	öffnet im Popup, nur statische Kartenelemente; dynamische Kartenelemente werden über Ebenenliste angezeigt
Maßstab/-sbalken	Balken unter der Karte
Werkzeugleiste	klar, teilweise durch Trennstriche gegliedert, nebenstehende Erklärungen überflüssig; Buttons: zoomIn/Out, pan, reset, Entferungsfunktion, Update, Legende, QuickView; auffallend: Zoomfaktor
Zoomnavigationsleiste	-
Pannavigationssteuerkreuz	im Kartenrand
Zooming allgemein	endlose max/min-Zoomtiefe; fehlende Zoomtiefenorientierung; Zoomfaktor einstellbar
Zooming per Doppelklick	-
Zooming per Mausrad	-
Zooming per Zoombox	ja (mit zoom-In-Werkzeug)
Panning allgemein	freies, stufenloses panning per Drag&Drop; zusätzlich schrittweise über Navigationspfeile
Zooming/panning über Tastatur	-
Tiling	-

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo http://www.mapsherpa.com/hawaii

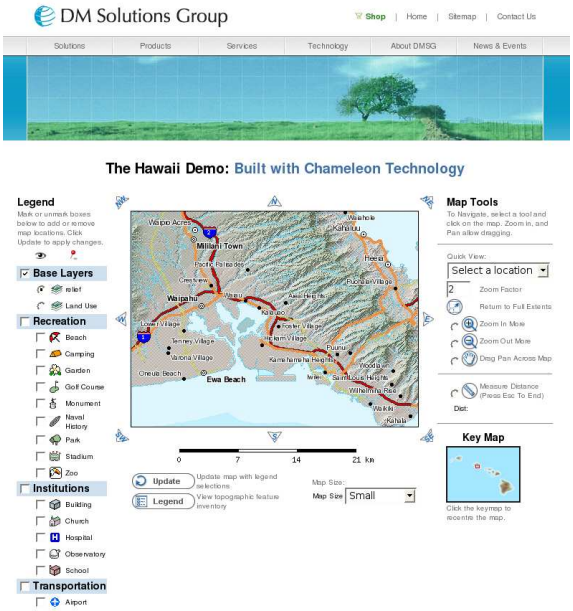
Analysefunktion	Entfernungsfunktion
Suchfunktion	-
Hilfefunktion	-
Druckfunktion	sehr gut: realisiert als pdf-Export, mit Formateinstellungen, Titel, Beschreibung, Legendenposition und Referenzkarte

7. Bemerkungen

	zahlreiche PlugIns (Widgets) verfügbar; Programmieren eigener Widgets leicht möglich

8. Screenshot

der untersuchten Demo



A.2.3 iGeoPortal

Stand: 15.05.2007

1. Allgemein

Name	**deegree iGeoPortal**
URL	
Home	http://deegree.org
Dokumentation	http://deegree.org/downloads/releases/igeoportal_std/deegree-igeoportal-std_v2pre1_doc.pdf
Download	http://deegree.org/deegree/portal/media-type/html/user/anon/page/default.psml/js_pane/download
Live-Demo	http://geoportal.wuppertal.de
aktuelle Version	1.2.1 (stable)
letztes Update	15.09.2005
Lizenz	GNU GPL
entwickelt von	lat/lon, Bonn
Kurzbeschreibung	iGeoPortal ist ein browserbasierender Client der auf die Dienste WMS, WFS und Proxy Service aufsetzt und vorwiegend zum Ansteuern des WMS dient. iGeoPortal ist die Klienten-/Portal-Komponente aus dem deegree-Projekt.

2. System

Architektur	client-serverseitig
Programmiersprache	PHP
Voraussetzungen	◦ Java 1.5.x ◦ Tomcat 5.5.x
unterstützte Browser	Mozilla/Firefox; IE
ggf. Integration anderer Software	-

3. Community

Revisionsverwaltung	SVN (http://wald.intevation.org/projects/deegree/)
Mailing Listen (URL)	iGeoPortal-Community nicht vorhanden - nur allgemeine deegree-User- und deegree-Dev-ML: https://lists.sourceforge.net/lists/listinfo/deegree-users https://lists.sourceforge.net/lists/listinfo/deegree-devel
Entwickler-ML	
Mails je Monat[1]	49
aktive Entwickler insgesamt[2]	202
Anwender-ML	
Mails je Monat[1]	92
aktive Anwender insgesamt[2]	106
kommerzieller Support	lat/lon, Bonn

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	eine neue (v2) und eine alte Doku als pdf vorhanden; nur kurze Installationsanleitung; umfangreicher Konfigurationsteil, URL s.o.

5. Usability

Alle Untersuchungen beziehen sich auf die Demo http://geoportal.wuppertal.de

Usability – Gesamteindruck	grundsätzlich verständlich und übersichtlich; relativ lange Ladezeiten für die Karte, Infomeldung 'Karte wird aktualisiert' fordert viel Geduld, besonders beim Panning inakzeptabel; unbegrenzte Zoomtiefe irritierend

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	viele Kartenthemen auswählbar
Übersichtskarte	konstante Zoomstufe, dadurch keine Zentrierung nötig; Ausschnitt nur durch Klick verschiebbar; passt sich nicht an Stil der Hauptkarte an
Ebenenübersicht	thematisch gegliedert (mithilfe von Tabs und Combobox); de-/aktivierbar durch Checkboxen; manuelle Aktualisierung der Karte nach Änderungen nötig; Ebenenreihenfolge änderbar, selektieren einer Ebene möglich
Legende	umfangreich; abhängig von ausgewählter Karte u. Themenliste; Legende, Ebenen u. Karten sollten zusammen dargestellt werden!
Maßstab/-sbalken	-
Werkzeugleiste	Klar, durch Trennstriche gegliedert; Aktualisierungsbutton umständlich; komfortables Hinzufügen von neuen WMS-Diensten, Download von Daten für registrierte User
Zoomnavigationsleiste	-
Pannavigationssteuerkreuz	im Kartenrand
Zooming allgemein	unbegrenztes Hereinzoomen, fehlende Zoomtiefenorientierung
Zooming per Doppelklick	-
Zooming per Mausrad	-
Zooming per Zoombox	ja (mit zoom-In-Werkzeug)
Panning allgemein	freies, stufenloses panning per Drag&Drop; zusätzlich schrittweise über Navigationspfeile
Zooming/panning über Tastatur	-
Tiling	-

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo http:// geoportal.wuppertal.de

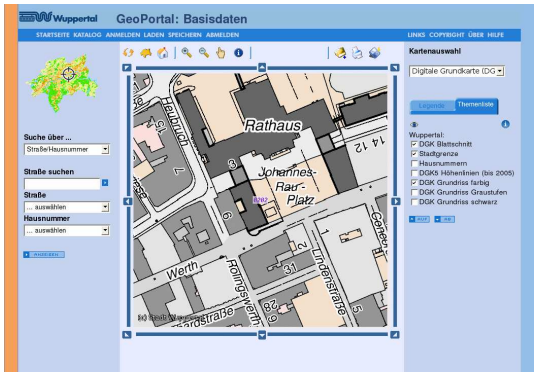
Analysefunktion	Sachdatenabfrage (Selektierung eines Eintrags in der Themenliste notwendig)
Suchfunktion	Stadtbezirk/Quartiersuche: 2 Comoboxen, Eingabe am Ende bestätigen; Straßensuche: mind. 3 Zeichen, Eingabe bestätigen, dann erscheinen alle passenden Strassen/HNr in Combobox, endgültige Auswahl muss noch mal mit extra Button bestätigt werden – sehr benutzerunfreundlich!
Hilfefunktion	ausführliche Benutzerhilfe in neuem Browserfenster aufrufbar
Druckfunktion	aktueller Kartenausschnitt mit zugehöriger Legende wird in neuem Browserfenster zum Druck geöffnet

7. Bemerkungen

iGeoPortal ist beliebig erweiterbar; bislang stehen folgende Module zur Verfügung:
map (unterstützt WMS, WMC und eigene Karten), download (Daten als GML oder Shapefile), gazetteer (unterstützt räumliche Recherche), security (Einbindung von deegree iGeoSecurity möglich), catalogue (Zugriff auf Metadaten über Catalogue Serivce)

8. Screenshot

der untersuchten Demo



A.2.4 ka-Map

Stand: 15.05.2007

1. Allgemein

Name	**ka-Map**
URL	
Home	http://ka-map.maptools.org
Dokumentation	http://ka-map.ominiverdi.org/wiki/index.php/Main_Page
Download	http://ka-map.maptools.org/index.phtml?page=downloads.html
Live-Demo	http://ka-map.ominiverdi.org/wiki/index.php/Links_to_some_ka-Map_applications
aktuelle Version	1.0
letztes Update	05.02.2007
Lizenz	MIT
entwickelt von	DM Solutions Group, Kanada
Kurzbeschreibung	Ka-Map zielt darauf ab, eine Javascipt API für die Entwicklung von hochinteraktiven WebMapping Schnittstellen bereitzustellen.

2. System

Architektur	client-serverseitig
Programmiersprache	JavaScript, PHP
Voraussetzungen	UMN MapServer und PHP/MapScript
unterstützte Browser	Mozilla/Firefox 1.0+; IE 6; Safari; Opera 7+; Netscape 7+; Epiphany
ggf. Integration anderer Software	basiert auf UMN MapServer

3. Community

Revisionsverwaltung	CVS (http://ka-map.maptools.org/index.phtml?page=cvs.html)
Mailing Listen (URL)	http://ka-map.maptools.org/index.phtml?page=mailinglist.html
Entwickler-ML	
Mails je Monat[1]	11
aktive Entwickler insgesamt[2]	7
Anwender-ML	
Mails je Monat[1]	96
aktive Anwender insgesamt[2]	121
kommerzieller Support	DM Solutions Group, Kanada

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	umfangreich (guter Einstieg mit Mapservereinstellungen etc., Overlay-API, Userguide(!) und Developer-Dokumentation); Installationsanleitung in der readme.txt im Installationsordner

5. Usability

Alle Untersuchungen beziehen sich auf die aktuelle CVS(!)-Demo: http://www.ominiverdi.org/ka-map/ka-map/htdocs/

Usability – Gesamteindruck	umfangreiche Werkzeugleiste; GUI übersichtlich; angenehmes Panning; Zoomtiefenorientierung fehlt

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	Kartengröße passt sich dynamisch an Browserfenstergröße an
Übersichtskarte	konstante Zoomstufe, dadurch keine Zentrierung nötig; Ausschnitt durch Drag&Drop bewegbar, Hauptkarte aktualisiert sich nach Loslassen
Ebenenübersicht	Kombiniert mit Legende; de-/aktivierbar durch Checkboxen; Ebenenreihenfolge und -transparenz änderbar; thematisch gegliedert und zusammenklappbar; viele Informationen: Verschachtelungsgefahr
Legende	kombiniert mit der Ebenendarstellung, in einem Tab aufrufbar
Maßstab/-sbalken	Balken halbtransparent in der Karte integriert; zusätzlich Maßstabsauswahlfeld
Werkzeugleiste	Umfangreich; übliche Funktionen, auffallend: send-this-view-to-a-friend-Button
Zoomnavigationsleiste	-
Pannavigationssteuerkreuz	-
Zooming allgemein	in max-/min-Zoomstufe werden passende Zoom-Buttons deaktiviert; insgeamt 5 Zooming-Werkzeuge (zoomIn, zoomBox, zoomOut, zoomReset, Maßtabsauswahlfeld);
Zooming per Doppelklick	-
Zooming per Mausrad	ja
Zooming per Zoombox	ja (mit extra ZoomBox-Werkzeug; farbige Unterlegung des aufgezogenen Bereichs)
Panning allgemein	freies, stufenloses panning per Drag&Drop; keine Navigationspfeile; Nachladen der Kacheln teilweise unangenehm verzögert
Zooming/panning über Tastatur	-
Tiling	ja

6. Weitere Features

Alle Untersuchungen beziehen sich auf die aktuelle CVS(!)-Demo: http://www.ominiverdi.org/ka-map/ka-map/htdocs/

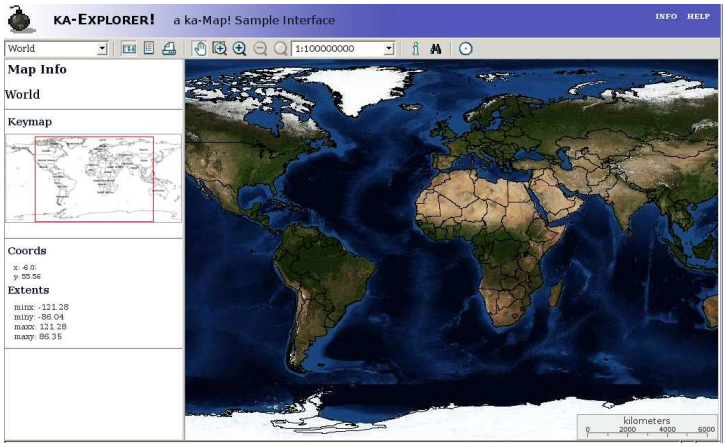
Analysefunktion	Sachdatenabfrage
Suchfunktion	Ländersuche
Hilfefunktion	über der Karte erscheint halbtransparente Kurzanleitung mit Icons und Erklärung, gut realisiert
Druckfunktion	speicherbar in diversen Formaten: pdf, png, jpg, gif, geotiff; mit Legende und Maßstab

7. Bemerkungen

AJAX u. MapServer-Einstellungen für ka-Map:
http://www.xml.com/lpt/a/1606

8. Screenshot

der untersuchten Demo



A.2.5 Mapbender

Stand: 15.05.2007

1. Allgemein

Name	**Mapbender**
URL	
Home	http://www.mapbender.org
Dokumentation	siehe Home
Download	http://www.mapbender.org/index.php/Download_Mapbender
Live-Demo	http://www.mapbender.org/index.php/Mapbender_Gallery
aktuelle Version	2.4.1
letztes Update	23.03.2007
Lizenz	GNU GPL
entwickelt von	WhereGroup, Bonn
Kurzbeschreibung	Mapbender ist ein webbasiertes GIS-Frontend implementiert als PHP-basierte Umgebung für das OGC-WMS/WFS-konforme Management der Darstellung, Wartung, Navigation und Abfrage.

2. System

Architektur	client-serverseitig
Programmiersprache	PHP
Voraussetzungen	∘ Webserver (Apache, MS IIS) ∘ PHP ∘ Datenbank (MySQL oder PostgreSQL)
unterstützte Browser	Mozilla/Firefox 1.x+, IE 6+
ggf. Integration anderer Software	-

3. Community

Revisionsverwaltung	SVN (http://www.mapbender.org/index.php/SVN)
Mailing Listen (URL)	http://www.mapbender.org/index.php/Mapbender_Mailing_Lists
Entwickler-ML	
Mails je Monat[1]	55
aktive Entwickler insgesamt[2]	46
Anwender-ML	
Mails je Monat[1]	107
aktive Anwender insgesamt[2]	98
kommerzieller Support	WhereGroup, Bonn

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	sehr gute Installationsanleitung (http://www.mapbender.org/index.php/Installation_de, deutsch); allgemeine Infos für Anwender; alles zweisprachig (englisch, deutsch)

5. Usability

Alle Untersuchungen beziehen sich auf die Demo http://www.mainz.de/mainzextern/geografischeinformationen/index.htm

Usability – Gesamteindruck	Auch ohne AJAX schnelle Ladezeiten; viele Funktionen; Zoomtiefenorientierung fehlt; Ebenenübersicht minimierbar – übersichtlich

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	beliebige Kartengröße per Drag&Drop einstellbar
Übersichtskarte	konstante Zoomstufe, dadurch keine Zentrierung nötig; Ausschnitt nur durch Klick verschiebbar, Aufziehen eines neuen Bereichs möglich
Ebenenübersicht	thematisch gegliedert und komplett minimierbar; Ebenen zusammenklappbar und de-/aktivierbar durch Checkboxen
Legende	Legendenbutton öffnet extra Fenster mit vielen Zusatzinfos; Nachteil: umständlich und verschachtelt
Maßstab/-sbalken	Balken in der Karte integriert (keine Transparenz); zusätzlich Maßstabsauswahlfeld und editierbares Maßstabstextfeld vorhanden
Werkzeugleiste	sehr umfangreich; auffallend: neuen Bildmittelpunkt bestimmen, Koordinaten anzeigen (auf Klick unter der Karte sichtbar), Abmelden (automatisch nach 15 min), Straßensuche, Koordinateneingabe
Zoomnavigationsleiste	-
Pannavigationssteuerkreuz	im Kartenrand
Zooming allgemein	Zoom-In/Out-Werkzeuge in max/min-Stufe weiterhin optisch aktiv (Funktion deaktiviert); Zoom-Reset; Zoom-History (previous, next)
Zooming per Doppelklick	-
Zooming per Mausrad	-
Zooming per Zoombox	ja (mit extra ZoomBox-Werkzeug)
Panning allgemein	freies, stufenloses panning per Drag&Drop; zusätzlich schrittweise über Navigationspfeile
Zooming/panning über Tastatur	-
Tiling	-

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo http://www.mainz.de/mainzextern/geografischeinformationen/index.htm

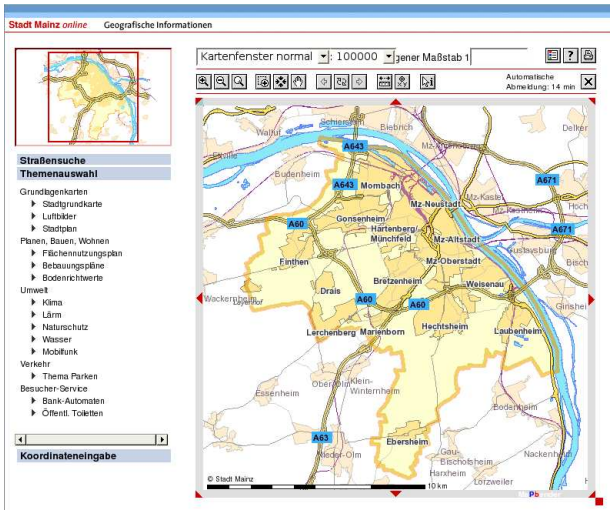
Analysefunktion	Sachdatenabfrage, Messfunktion, Koordinatenkreuz
Suchfunktion	Straßensuche
Hilfefunktion	Bedienungshinweisseite, Tooltipps
Druckfunktion	erzeugt HTML; mit Titel, Datum, Maßstab, Beschreibung

7. Bemerkungen

- Geo-CMS
- Securitymanagement
- schneller Kartenaufbau durch PNG-Interlacing

8. Screenshot

der untersuchten Demo



A.2.6 Mapbuilder

Stand: 15.05.2007

1. Allgemein

Name	**Mapbuilder**
URL	
Home	http://communitymapbuilder.org
Dokumentation	http://communitymapbuilder.org/display/MAP/User+Guide
Download	http://communitymapbuilder.org/display/MAP/Downloads
Live-Demo	http://communitymapbuilder.org/display/MAP/Examples
aktuelle Version	1.0.1
letztes Update	19.07.2006
Lizenz	GNU LGPL
entwickelt von	OSGeo
Kurzbeschreibung	Mapbuilder ist ein AJAX, webbasierter Karten-Client für die OGC Dienste WMS und transaktionalen WFS (WFS-T). Das modulare Design erlaubt den Einbau eigener Widgets.

2. System

Architektur	client-serverseitig
Programmiersprache	JavaScript, XML
Voraussetzungen	Apache/PHP oder Tomcat
unterstützte Browser	Firefox 1.0+, Internet Explorer 6.0+, Mozilla 1.3+, Navigator 6+ inkompatibel: IE 5.5, Safari, Netscape 4
ggf. Integration anderer Software	-

3. Community

Revisionsverwaltung	SVN (http://communitymapbuilder.org/display/MAP/SVN+Repository)
Mailing Listen (URL)	http://ka-map.maptools.org/index.phtml?page=mailinglist.html
Entwickler-ML	
Mails je Monat[1]	120
aktive Entwickler insgesamt[2]	47
Anwender-ML	
Mails je Monat[1]	65
aktive Anwender insgesamt[2]	78
kommerzieller Support	Kontaktaufnahme über die dev-ML

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	UserGuide gut gegliedert und kompakt geschrieben; Installation: http://communitymapbuilder.org/display/MAP/Installing+MapBuilder; Tutorials und QuickStart siehe Link unter (1)

5. Usability

Alle Untersuchungen beziehen sich auf die Demo http://geoservices.cgdi.ca/mapbuilder/demo/Demis/index.html

Usability – Gesamteindruck	übersichtlich und selbsterklärend; 'Zurück' und 'Vor'-Buttons auffallend; keine Rückmeldung über Ladezeitstatus während der Aktualisierungen, max/min-Zoomtiefe nicht definiert (beide enden bei hinreichend weitem Zoom in WMS-Fehlermeldung)

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	aktuelle Steuerkreuzposition in Grad, Min. und Sek. am Kartenrand
Übersichtskarte	-
Ebenenübersicht	de-/aktivierbar durch Checkboxen (siehe Time Series Demo http://geoservices.cgdi.ca/mapbuilder/demo/timeSeries/index.html)
Legende	-
Maßstab/-sbalken	nur editierbares Maßstabstextfeld vorhanden
Werkzeugleiste	übliche Funktionen, auffallend(1): Zoom-History (speichert die letzten aufgerufenen Kartenausschnitte); auffallend(2): *area of interest*-Funktion (genaue Funktion unklar – in der Demo nicht testbar)
Zoomnavigationsleiste	-
Pannavigationssteuerkreuz	-
Zooming allgemein	max-Zoomstufe nur am Maßstabstextfeld (Wert '0') erkennbar, bei weiterem Reinzoomen folgt WMS-Fehlermeldung; min-Zoomstufe nicht festgelegt, Karte verschwindet bei hinreichend weitem Rauszoomen, WMS-Fehlermeldung folgt – Zoomlupe(n) weiterhin aktiv
Zooming per Doppelklick	-
Zooming per Mausrad	-
Zooming per Zoombox	ja (auch mit gedrückter Shifttaste möglich; farbige Unterlegung des aufgezogenen Bereichs)
Panning allgemein	freies, stufenloses panning per Drag&Drop; keine Navigationspfeile
Zooming/panning über Tastatur	-
Tiling	-

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo http://geoservices.cgdi.ca/mapbuilder/demo/Demis/index.html

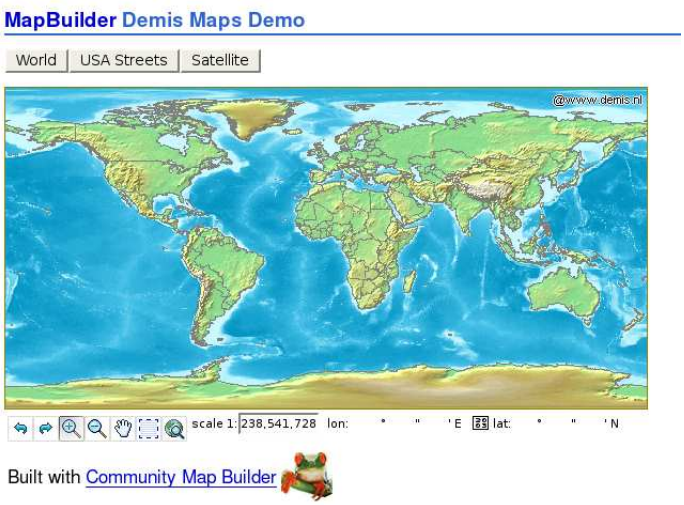
Analysefunktion	'area of interest'-Funktion (siehe Werkzeugleiste)
Suchfunktion	-
Hilfefunktion	-
Druckfunktion	-

7. Bemerkungen

- nutzt AJAX
- ModelViewController design pattern
- Zeichnet Karten von WMS, WFS, GeoRSS und Google Maps
- unterstützt Web Map Context (WMC) und Open Web Services Context
- SVG/VML-Rendering ab Version 1.5

8. Screenshot

der untersuchten Demo



A.2.7 MapGuide Open Source

Stand: 15.05.2007

1. Allgemein

Name	**MapGuide Open Source**
URL	
Home	https://mapguide.osgeo.org
Dokumentation	https://mapguide.osgeo.org/documentation.html
Download	https://mapguide.osgeo.org/downloads.html
Live-Demo	https://mapguide.osgeo.org/livegallery.html
aktuelle Version	1.1.0
letztes Update	09.12.2006
Lizenz	GNU LGPL
entwickelt von	Autodesk, USA; jetzt: OSGeo
Kurzbeschreibung	MapGuide Open Source ist eine Web Mapping Plattform für die Entwicklung und den Einsatz von raumbezogenen Anwendungen.

2. System

Architektur	client-serverseitig
Programmiersprache	PHP, für die Anwendungsentwicklung auch .NET oder Java möglich
Voraussetzungen	○ MapGuide Server (Linux oder Windows) ○ Webserver (Apache oder MS IIS) ○ Application Development (PHP 5.0.5+; .NET Framework 2.0 (optional); Java JDK 5.0 und Tomcat Servlet engine version 5.5.12 (optional)) ○ PROJ.4
unterstützte Browser	Mozilla Firefox, IE, Safari
ggf. Integration anderer Software	-

3. Community

Revisionsverwaltung	SVN (https://mapguide.osgeo.org/subversionconfig.html)
Mailing Listen (URL)	dev (http://lists.osgeo.org/mailman/listinfo/mapguide-internals) user (http://lists.osgeo.org/mailman/listinfo/mapguide-users)
Entwickler-ML	
Mails je Monat[1]	168
aktive Entwickler insgesamt[2]	42
Anwender-ML	
Mails je Monat[1]	527
aktive Anwender insgesamt[2]	297
kommerzieller Support	Autodesk, USA

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	umfangreiche Doku und Installationsanleitung, GettingStarted-Anleitung sehr gut; URL unter (1)

5. Usability

Alle Untersuchungen beziehen sich auf die Demo http://data.mapguide.com/mapguide/phpviewersample/ajaxtiledviewersample.php

Usability – Gesamteindruck	bewegbare Zoomnavigationsleiste; übersichtliches Layout; Statusleiste mit Koordinaten und Maßstab; Mauscursor passt sich nicht gewählten Funktionen an – irritierend

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	Kartengröße passt sich dynamisch an Browserfenstergröße an
Übersichtskarte	-
Ebenenübersicht	kombiniert mit Legende; de-/aktivierbar durch Checkboxen; thematisch gegliedert und zusammenklappbar
Legende	-
Maßstab/-sbalken	nur Maßstabszahl in Statusbar vorhanden
Werkzeugleiste	übliche Funktionen (ZoomIn/Out, Zoombox, Pan, Sachdatenabfrage, Entferungs- und Druckfunktion); auffallend: Zoom-Menü mit History-Buttons (previous, next) und Reset sowie eine umfangreiche 'Umkreis-funktion' (Buffer)
Zoomnavigationsleiste	frei auf der Karte platzierbar(!), zusammen mit Pannavigation
Pannavigationssteuerkreuz	unterhalb der Zoombar integriert
Zooming allgemein	gute Zoomtiefenorientierung durch Zoomnavigationsleiste; keine Deaktivierung der Zoombuttons bei Erreichen der Endstufe
Zooming per Doppelklick	-
Zooming per Mausrad	-
Zooming per Zoombox	ja (mit extra Zoombox-Werkzeug; auch mit gedrückter Shifttaste möglich; farbige Unterlegung des aufgezogenen Bereichs)
Panning allgemein	freies, stufenloses panning per Drag&Drop; zusätzlich schrittweise über Navigationspfeile der Zoomnavigationsleiste
Zooming/panning über Tastatur	-
Tiling	ja

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo http://data.mapguide.com/mapguide/phpviewersample/ajaxtiledviewersample.php

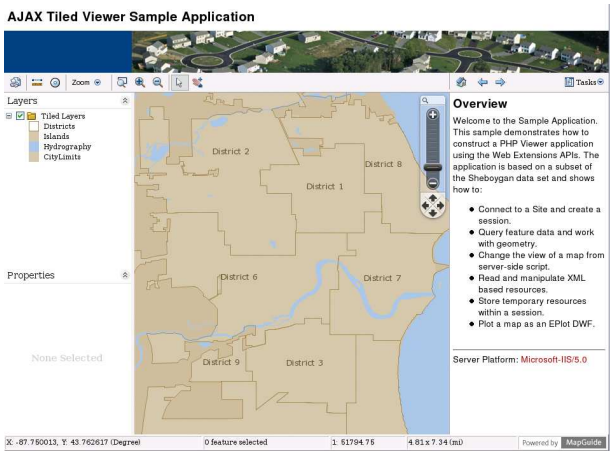
Analysefunktion	Sachdatenabfrage, Entfernungsfunktion, Umfangfunktion
Suchfunktion	Adress- und Eigentümersuche
Hilfefunktion	-
Druckfunktion	optionales Hinzufügen von Legenden, Titel, Nordpfeil; öffnet als HTML im Popup; gute Realisierung

7. Bemerkungen

- nutzt eigene XML-Datenbankstruktur
- 2 Viewer: AJAX und DWF (vektorbasiert, nur für Win, PlugIn erforderlich)
- kommerziell-proprietäre Version: Autodesk MapGuide Enterprise 2007
- passendes Authoring-Tool MapGuide Studio in MapGuide Web Server Extensions enthalten
- im November 2005 von Autodesk unter GNU LGPL gestellt
- http://www.heise.de/newsticker/meldung/66808

8. Screenshot

der untersuchten Demo



A.2.8 MappingWidgets

Stand: 15.05.2007

1. Allgemein

Name	**MappingWidgets**
URL	
Home	http://mappingwidgets.sourceforge.net
Dokumentation	http://mappingwidgets.sourceforge.net/manual
Download	http://sourceforge.net/project/showfiles.php?group_id=130528
Live-Demo	http://mappingwidgets.sourceforge.net/demo/mapserver
aktuelle Version	0.3.1
letztes Update	17.03.2006
Lizenz	GNU GPL
entwickelt von	k. A.
Kurzbeschreibung	Ein Karten-Widget (Zoom, Verschieben, Info etc.) für einfache OGC WMS Klienten. Dafür wird PHP Smarty um entsprechende Plugins erweitert.

2. System

Architektur	client-serverseitig
Programmiersprache	PHP, JavaScript
Voraussetzungen	∘ Smarty (PHP template framework) ∘ UMN Mapserver und PHP/MapScript (optional)
unterstützte Browser	k. A.
ggf. Integration anderer Software	basiert auf UMN MapServer (optional)

3. Community

Revisionsverwaltung	SVN (http://sourceforge.net/svn/?group_id=130528)
Mailing Listen (URL)	-
Entwickler-ML	
Mails je Monat[1]	-
aktive Entwickler insgesamt[2]	-
Anwender-ML	
Mails je Monat[1]	-
aktive Anwender insgesamt[2]	-
kommerzieller Support	k. A.

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	Installation, Usage, Design - sehr kurz und knapp, Beispiele fehlen, nicht sehr hilfreich; URL unter (1)

5. Usability

Alle Untersuchungen beziehen sich auf die Demo http://mappingwidgets.sourceforge.net/demo/mapserver

Usability – Gesamteindruck	keine Rückmeldung über Ladezeitstatus während der Aktualisierungen; Panning nur per Drag&Drop; endlose max/min-Zoomtiefe irritierend; Zoom-History-Buttons auffallend

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	einfach; Kartengröße nicht änderbar
Übersichtskarte	-
Ebenenübersicht	-
Legende	Scrollbar, thematisch geliedert
Maßstab/-sbalken	-
Werkzeugleiste	übliche Funktionen (ZoomIn/Out, Pan, Sachdatenabfrage, Entfernungsfunktion); auffallend: Zoom-History-Buttons (first, previous, next, last)
Zoomnavigationsleiste	-
Pannavigationssteuerkreuz	-
Zooming allgemein	endlose max/min-Zoomtiefe; fehlende Zoomtiefenorientierung
Zooming per Doppelklick	-
Zooming per Mausrad	-
Zooming per Zoombox	ja (mit ZoomIn-Werkzeug; farbige Unterlegung des aufgezogenen Bereichs)
Panning allgemein	freies, stufenloses panning per Drag&Drop; keine Navigationspfeile
Zooming/panning über Tastatur	-
Tiling	-

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo http://mappingwidgets.sourceforge.net/demo/mapserver

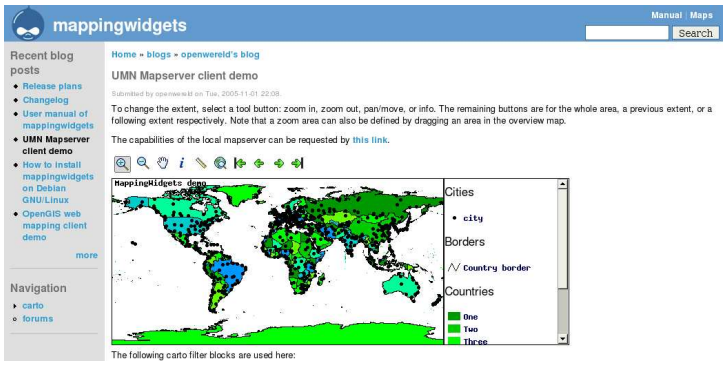
Analysefunktion	Sachdatenabfrage, Entfernungsfunktion
Suchfunktion	-
Hilfefunktion	-
Druckfunktion	-

7. Bemerkungen

Modul für CMS »Drupal« möglich

8. Screenshot

der untersuchten Demo



A.2.9 p.mapper

Stand: 15.05.2007

1. Allgemein

Name	**p.mapper**
URL	
Home	http://www.pmapper.net
Dokumentation	http://www.pmapper.net/documentation.shtml
Download	http://www.pmapper.net/download.shtml
Live-Demo	http://www.pmapper.net/demo.shtml
aktuelle Version	3.0.1 (stable)
letztes Update	30.12.2006
Lizenz	GNU GPL
entwickelt von	Armin Burger
Kurzbeschreibung	P.mapper ist ein Freies MapServer Templatesystem, das mit PHPMapScript realisiert ist und eine ansprechende Oberfläche zur Verfügung stellt.

2. System

Architektur	client-serverseitig
Programmiersprache	PHP
Voraussetzungen	UMN MapServer und PHP/MapScript
unterstützte Browser	Mozilla/Firefox 1.x+; IE 5/6; Opera 6.+: Netscape 6.1+
ggf. Integration anderer Software	basiert auf UMN MapServer

3. Community

Revisionsverwaltung	SVN (http://www.pmapper.net/download.shtml)
Mailing Listen (URL)	https://lists.sourceforge.net/lists/listinfo/pmapper-users
Entwickler-ML	
Mails je Monat[1]	-
aktive Entwickler insgesamt[2]	-
Anwender-ML	
Mails je Monat[1]	82
aktive Anwender insgesamt[2]	61
kommerzieller Support	u. a. Intevation GmbH, Osnabrück

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	Quick Install Anleitung und User manual; URL unter (1)

5. Usability

Alle Untersuchungen beziehen sich auf die Demo http://www.pmapper.net/demo.shtml (medium)

Usability – Gesamteindruck	lange Ladezeiten, kein AJAX-unterstütztes Panning; viele Funktionen; subjektiv schnell wirkender Zoomvorgang durch *continuous zooming* Effekt; Demo nur in Popup aufrufbar

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)

Hauptkarte	Kartengröße passt sich an Browserfenster an
Übersichtskarte	konstante Zoomstufe, dadurch keine Zentrierung nötig; Ausschnitt verschiebbar per Drag&Drop bzw. Mausklick
Ebenenübersicht	Kombiniert mit Legende; de-/aktivierbar durch Checkboxen; thematisch gegliedert und zusammenklappbar
Legende	kombiniert mit der Ebenendarstellung
Maßstab/-sbalken	Balken in der Karte integriert; zusätzlich editierbares Maßstabstextfeld vorhanden
Werkzeugleiste	umfangreich; auffallend: Punkt-hinzufügen (Marker setzen); Download; Selektieren (zur Mehrfachabfrage, Combobox-Auswahl unter der Karte beachten), Auto-Identify (=Infoabfrage)
Zoomnavigationsleiste	ja (bemerkenswert: *continuous zooming* mit Slider, d. h. on-the-fly skalieren der Karte; ermöglicht gute Zoomtiefenorientierung)
Pannavigationssteuerkreuz	-
Zooming allgemein	Max-Zoomstufe '1:0' unsinnig; ZoomIn/Out-Werkzeug in max/min-Stufe weiterhin aktiv; Zoom-Reset; Zoom-History (previous, next)
Zooming per Doppelklick	-
Zooming per Mausrad	ja (seit v3.1)
Zooming per Zoombox	ja (mit ZoomIn-Werkzeug; zusätzlich auch mit gedrückter Shifttaste möglich; farbige Unterlegung des aufgezogenen Bereichs)
Panning allgemein	freies, stufenloses panning per Drag&Drop; keine Navigationspfeile
Zooming/panning über Tastatur	ja
Tiling	-

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo http://www.pmapper.net/demo.shtml (medium)

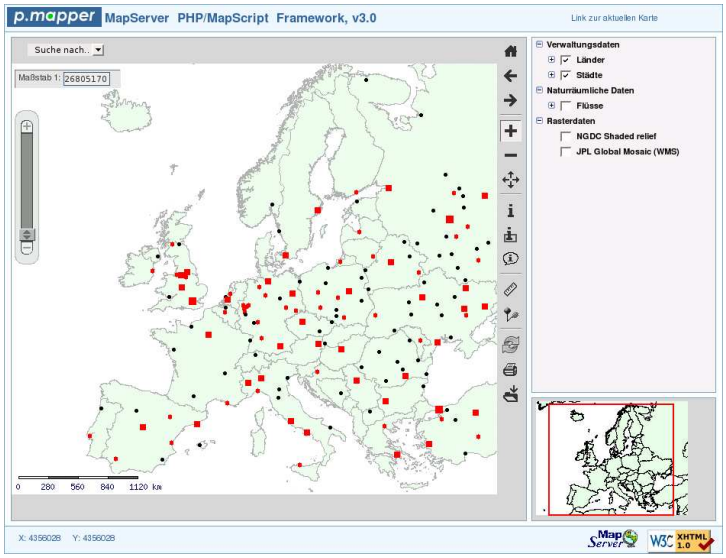
Analysefunktion	Sachdatenabfrage, Entfernungsmessung, Punkt hinzufügen
Suchfunktion	Länder- und Städtesuche
Hilfefunktion	-
Druckfunktion	HTML und PDF erzeugbar; mit Referenzkarte, Legende und Maßstab

7. Bemerkungen

-

8. Screenshot

der untersuchten Demo



A.3 Proprietär & clientseitig

Proprietäre Webmapping-Anwendung mit 100% JavaScript

A.3.1 Google Maps

Stand: 15.05.2007

1. Allgemein

Name	**Google Maps**
URL	
Home	http://maps.google.de
Dokumentation	API: http://www.google.com/apis/maps/documentation/ FAQ: http://www.google.com/apis/maps/faq.html
Download	-
Live-Demo	siehe Home
aktuelle Version	2.79 (API)
letztes Update	18.04.2007
Lizenz	*proprietär*
entwickelt von	Google, USA
Kurzbeschreibung	GoogleMaps ist eine JavaScript-API, um dynamische Karten in beliebigen Webseiten zu integrieren.

2. System

Architektur	clientseitig
Programmiersprache	JavaScript
Voraussetzungen	-
unterstützte Browser	Firefox 0.8+, IE 6.0+, Safari 1.2.4+, Netscape 7.1+, Mozilla 1.4+, Opera 8.02+
ggf. Integration anderer Software	k. A.

3. Community

Revisionsverwaltung	-
Mailing Listen (URL)	GoogleGroups: http://groups.google.de/group/Google-Maps-DE (deutsch) http://groups.google.de/group/Google-Maps-API (englisch)
Entwickler-ML	
Mails je Monat[1]	-
aktive Entwickler insgesamt[2]	-
Anwender-ML	
Mails je Monat[1]	-[3]
aktive Anwender insgesamt[2]	-[3]
kommerzieller Support	Google

4. Dokumentation

zur Installation/Entwicklung/Anwendung (Hinweise, Tutorials, URL)	ausführliche API-Referenz mit vielen praktischen Beispielen; URL unter (1)

5. Usability

Alle Untersuchungen beziehen sich auf die Demo http://maps.google.de

Usability – Gesamteindruck	GUI intuitiv, klar und attraktiv; überragendes Pan-Zoomverhalten; sehr flüssiges und komfortables Navigieren

Fortsetzung auf der nächsten Seite

[1] durchschnittliche monatliche Mailanzahl im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[2] Gesamtanzahl der aktiven Entwickler/Anwender im untersuchten Zeitraum 10/2006 - 03/2007 (6 Monate)
[3] Automatische Analyse in *Google Groups* nicht möglich.

Hauptkarte	fast alle Elemente/Werkzeuge werden auf der Karte platziert; Kartengröße passt sich dynamisch an Browserfenstergröße an
Übersichtskarte	minimierbar; Ausschnitt verschiebbar per Drag&Drop bzw. Doppelklick; automatisches, animiertes Zentrieren des Ausschnittes nach Verschieben; passt sich an Stil der Hauptkarte an
Ebenenübersicht	keine typische Ebenendarstellung; 3 Kartenansichten (Karte, Satellit und Hybrid) per Button wählbar
Legende	-
Maßstab/-sbalken	Balken in der Karte integriert
Werkzeugleiste	keine typische Werkzeugleiste: Drucken, Email, URL zu dieser Seite sowie umfangreiche Suchfunktion, Branchensuche, Routenberechnung; nur mit Google-Account: Standorte speicherbar und Setzen von persönlichen, georeferenzierten Kommentaren (über *Meine Karten*); schlicht und sehr klar gegliedert
Zoomnavigationsleiste	in der Karte integriert; inkl. Zoom-Reset-Button im Pansteuerkreuz
Pannavigationssteuerkreuz	oberhalb der Zoombar
Zooming allgemein	Wechsel zwischen Pan- und Zoommodus nicht notwendig; zoomen über Zoomnavigationsleiste oder Doppelklick; gute Zoomtiefenorientierung
Zooming per Doppelklick	ja
Zooming per Mausrad	ja (mit gutem optischen Effekt)
Zooming per Zoombox	-
Panning allgemein	freies, stufenloses panning per Drag&Drop; Nachladen der Karte im Hintergrund bewirkt absolut verzögerungsfreies(!) Bewegen der Karte
Zooming/panning über Tastatur	ja
Tiling	ja

6. Weitere Features

Alle Untersuchungen beziehen sich auf die Demo http://maps.google.de

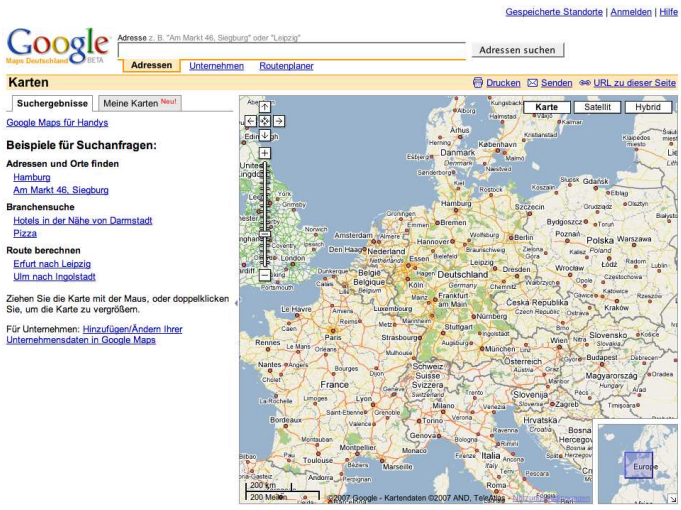
Analysefunktion	-
Suchfunktion	Adresssuche (Geocoding), Branchensuche
Hilfefunktion	sehr umfangreich
Druckfunktion	öffnet als HTML-Popup; Druckdialog öffnet automatisch; Notizen können über Formularfeld eingegeben werden; Satellitenkarten nicht druckbar

7. Bemerkungen

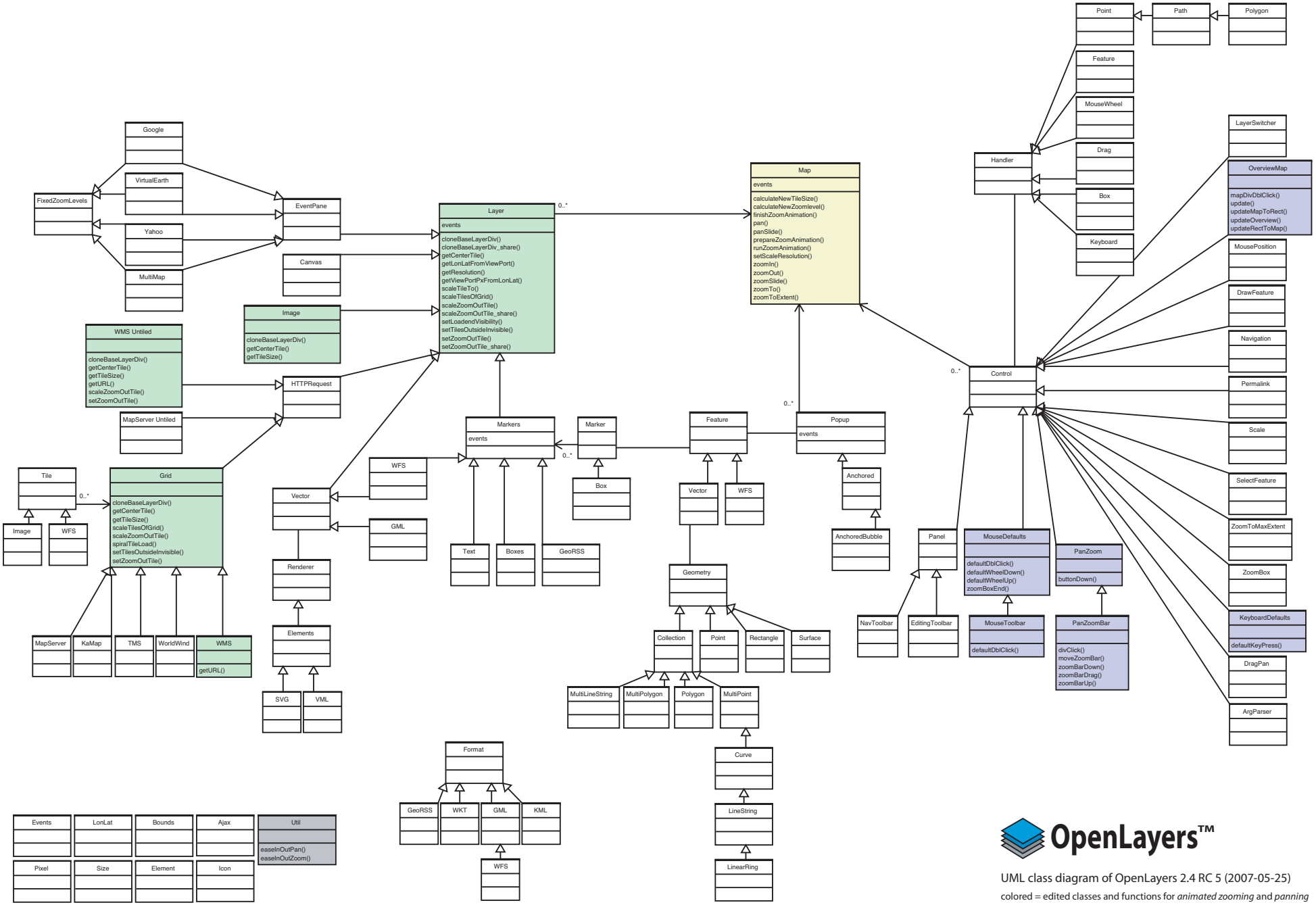
- proprietäre AJAX-Lösung
- gute API zur Einbindung von Karten in eigene Website (Integration nur durch ein aufwändiges Keyverfahren); keine kommerzielle Nutzung erlaubt
- gestartet am 8.2.2005
- changelogs: http://mapki.com/wiki/Changelog

8. Screenshot

der untersuchten Demo



B Klassendiagramm



OpenLayers™

UML class diagram of OpenLayers 2.4 RC 5 (2007-05-25)

colored = edited classes and functions for *animated zooming* and *panning*

C Testplan

Datum:	Testplanversion: 1.1
Betriebssystem:	Browser:
Tester:	Testdauer:
Kommentar:	

Dieser Testplan kann dazu genutzt werden, zu verifizieren, dass alle Funktionen des *animated zooming & panning* Features in *OpenLayers* wie erwartet funktionieren. Jede Testsuite kann separat ausgeführt werden, so dass es nicht notwendig ist, alle Tests hintereinander auszuführen.

Geben Sie für einen Fehlerbericht bitte die Versionsnummer dieses Dokuments (1.1) und die Nummer des Tests an, der fehlgeschlagen ist.

Für alle Tests soll die *OpenLayers*-Beispieldemo *controls.html* im Verzeichnis *examples* verwendet werden.

Überblick

Testabschnitt komplett erfolgreich?

Testsuite 1: Zooming über die *Zoombar* ..... $\square_{ja}$ $\square_{nein}$
Testsuite 2: Zooming über die Maus ..... $\square_{ja}$ $\square_{nein}$
Testsuite 3: Zooming über die Tastatur ..... $\square_{ja}$ $\square_{nein}$

Testsuite 4: Panning über die *Panbar* ..... $\square_{ja}$ $\square_{nein}$
Testsuite 5: Panning über die Maus ..... $\square_{ja}$ $\square_{nein}$
Testsuite 6: Panning über die Tastatur ..... $\square_{ja}$ $\square_{nein}$
Testsuite 7: Panning über die Übersichtskarte ..... $\square_{ja}$ $\square_{nein}$

Testsuite C.1: Zooming über die *Zoombar*

C.1.1 Klicken Sie auf den »+« Button am oberen Rand der Zoombar. Wird in die Hauptkarte hineingezoomt? ..... $\square_{ja}$ $\square_{nein}$
Erhöht sich die Zoomstufe um genau eine Stufe? ..... $\square_{ja}$ $\square_{nein}$
Läuft der ZoomIn-Prozess animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich der Zoomslider zeitgleich zum ZoomIn-Prozess um eine Zoomstufe nach oben? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird die skalierte Hauptkarte durch (nacheinander) neu gezeichnete Kacheln aktualisiert? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden der gesamten Karte die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.1.2 Klicken Sie auf den »–« Button am unteren Rand der Zoombar. Wird aus der Hauptkarte herausgezoomt? ..... $\square_{ja}$ $\square_{nein}$
Verringert sich die Zoomstufe um genau eine Stufe? ..... $\square_{ja}$ $\square_{nein}$
Läuft der ZoomOut-Prozess animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich der Zoomslider zeitgleich zum ZoomOut-Prozess um eine Zoomstufe nach unten? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird während der Zoomanimation der sichtbare Bereich um die kleiner werdende Originalkarte mit neuen Karteninformationen erweitert? ..... $\square_{ja}$ $\square_{nein}$
Wird die skalierte Hauptkarte durch (nacheinander) neu gezeichnete Kacheln aktualisiert? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden der gesamten Karte die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.1.3 Klicken Sie auf einen Punkt auf der Zoombar im Bereich des obersten Zoomlevelfeldes (direkt unter dem »+« Button) der Zoombar. Wird in die Hauptkarte bis zur maximalen ZoomIn-Stufe hineingezoomt? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich der Zoomslider zeitgleich zum ZoomIn-Prozess bis zum oberen Anschlag? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden aller Kacheln die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.1.4 Klicken Sie auf den »+« Button am oberen Rand der Zoombar. Bleibt die Hauptkarte, der Zoomslider und die Statusleiste unverändert? ..... $\square_{ja}$ $\square_{nein}$

C.1.5 Klicken Sie auf einen Punkt auf der Zoombar im Bereich des untersten Zoomlevelfeldes (direkt über dem »–« Button) der Zoombar. Wird aus der Hauptkarte bis zur maximalen ZoomOut-Stufe herausgezoomt? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich der Zoomslider zeitgleich zum ZoomOut-Prozess bis zum unteren Anschlag? ..... $\square_{ja}$ $\square_{nein}$

Symbolisiert die Statusleiste des Browsers nach vollständigem Laden aller Kacheln die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.1.6 Klicken Sie auf den »–« Button am unteren Rand der Zoombar. Bleibt die Hauptkarte, der Zoomslider und die Statusleiste unverändert? ..... $\square_{ja}$ $\square_{nein}$

C.1.7 Klicken Sie auf den Slider und ziehen Sie ihn langsam mit gedrückter (linker) Maustaste bis an das obere Ende der Zoombar. Lassen Sie die Maustaste im Bereich des obersten Zoomlevelfeldes los. Wird beim Bewegen des Sliders in die Hauptkarte hineingezoomt? . $\square_{ja}$ $\square_{nein}$
Wird beim Bewegen des Sliders in die Übersichtskarte hineingezoomt? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird die skalierte Hauptkarte nach dem Loslassen der Maustaste durch (nacheinander) neu gezeichnete Kacheln aktualisiert? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden aller Kacheln die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.1.8 Klicken Sie auf den Slider und ziehen Sie ihn langsam mit gedrückter (linker) Maustaste bis an das untere Ende der Zoombar. Lassen Sie die Maustaste im Bereich des untersten Zoomlevelfeldes los. Wird beim Bewegen des Sliders aus der Hauptkarte herausgezoomt? $\square_{ja}$ $\square_{nein}$
Wird beim Bewegen des Sliders aus der Übersichtskarte herausgezoomt? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird beim Bewegen des Sliders der sichtbare Bereich um die kleiner werdende Originalkarte mit neuen Karteninformationen erweitert? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden aller Kacheln die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.1.9 Klicken Sie auf den Slider und ziehen Sie ihn langsam mit gedrückter (linker) Maustaste bis an das obere und anschließend wieder zurück bis ans untere Ende der Zoombar. Lassen Sie die Maustaste im Bereich des untersten Zoomlevelfeldes los. Wird beim Bewegen des Sliders in die Hauptkarte zunächst hinein- und anschließend herausgezoomt? ..... $\square_{ja}$ $\square_{nein}$
Wird beim Bewegen des Sliders in die Übersichtskarte zunächst hinein- und anschließend herausgezoomt? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Bleibt die Hauptkarte und der Slider nach dem Loslassen der Maustaste in Zoomstufe 0 stehen? ..... $\square_{ja}$ $\square_{nein}$

C.1.10 Machen Sie einen Doppelklick auf den »+« Button am oberen Ende der Zoombar. Wird die Zoomstufe um genau *eine* Stufe erhöht? ..... $\square_{ja}$ $\square_{nein}$

C.1.11 Klicken Sie zunächst einmal auf den »+« Button. Machen Sie anschließend einen Doppelklick auf den »–« Button am unteren Ende der Zoombar. Betrachten Sie nur die Doppelklick-Auswirkung. Wird die Zoomstufe um genau *eine* Stufe verringert? ..... $\square_{ja}$ $\square_{nein}$

Testsuite C.2: Zooming über die Maus

C.2.1 Machen Sie einen Doppelklick auf eine beliebige Stelle in der Hauptkarte. Zentriert sich die Hauptkarte auf die Position des Doppelklicks? ..... $\square_{ja}$ $\square_{nein}$
Wird in die Hauptkarte hineingezoomt? ..... $\square_{ja}$ $\square_{nein}$
Erhöht sich die Zoomstufe um genau eine Stufe? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Pan-ZoomIn-Prozess animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich der Zoomslider zeitgleich zum Pan-ZoomIn-Prozess um eine Zoomstufe nach oben? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird die skalierte Hauptkarte durch (nacheinander) neu gezeichnete Kacheln aktualisiert? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden der gesamten Karte die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.2.2 Bewegen Sie den Mauszeiger bis zu einem beliebigen Punkt auf der Hauptkarte. Bewegen Sie anschließend das Mausrad um eine Stufe nach oben (vom Körper weg). Wird in die Hauptkarte hineingezoomt? ..... $\square_{ja}$ $\square_{nein}$
Befindet sich der geografische Punkt unter dem Mauszeiger nach dem Zoomen an der gleichen Position in der Hauptkarte, wie vor dem Zoomvorgang? ..... $\square_{ja}$ $\square_{nein}$
Erhöht sich die Zoomstufe um genau eine Stufe? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Pan-ZoomIn-Prozess animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich der Zoomslider zeitgleich zum Pan-ZoomIn-Prozess um eine Zoomstufe nach oben? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird die skalierte Hauptkarte durch (nacheinander) neu gezeichnete Kacheln aktualisiert? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden der gesamten Karte die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.2.3 Bewegen Sie den Mauszeiger bis zu einem beliebigen Punkt auf der Hauptkarte. Bewegen Sie anschließend das Mausrad um eine Stufe nach unten (zum Körper hin). Wird aus der Hauptkarte herausgezoomt? ..... $\square_{ja}$ $\square_{nein}$
Befindet sich der geografische Punkt unter dem Mauszeiger nach dem Zoomen an der gleichen Position in der Hauptkarte wie vor dem Zoomvorgang? ..... $\square_{ja}$ $\square_{nein}$
Verringert sich die Zoomstufe um genau eine Stufe? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Pan-ZoomOut-Prozess animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich der Zoomslider zeitgleich zum Pan-ZoomOut-Prozess um eine Zoomstufe nach unten? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird während der Zoomanimation der sichtbare Bereich um die kleiner werdende Originalkarte mit neuen Karteninformationen erweitert? ..... $\square_{ja}$ $\square_{nein}$
Wird die skalierte Hauptkarte durch (nacheinander) neu gezeichnete Kacheln aktualisiert? ..... $\square_{ja}$ $\square_{nein}$

Symbolisiert die Statusleiste des Browsers nach vollständigem Laden der gesamten Karte die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.2.4 Bewegen Sie den Mauszeiger auf einen beliebigen Punkt auf der Hauptkarte. Bewegen Sie anschließend das Mausrad um mehrere Stufen nach oben (vom Körper weg). Wird die Zoomstufe um genau *eine* Stufe erhöht? ..... $\square_{ja}$ $\square_{nein}$

C.2.5 Bewegen Sie den Mauszeiger auf einen beliebigen Punkt auf der Hauptkarte. Bewegen Sie anschließend das Mausrad um mehrere Stufen nach unten (zum Körper hin). Wird die Zoomstufe um genau *eine* Stufe verringert? ..... $\square_{ja}$ $\square_{nein}$

C.2.6 Ziehen Sie mit der Maus und gedrückter Shift-Taste einen Rahmen auf der Karte auf, deren Fläche maximal 1/4 der Haupkarte beträgt. Wird bis auf den (in etwa) markierten Ausschnitt hineingezoomt? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Pan-ZoomIn-Prozess animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich der Zoomslider zeitgleich zum Pan-ZoomIn-Prozess um mindestens eine Zoomstufe nach oben? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird die skalierte Hauptkarte durch (nacheinander) neu gezeichnete Kacheln aktualisiert? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden der gesamten Karte die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.2.7 Klicken Sie mit der Maus und gedrückter Shift-Taste auf einen beliebigen Punkt auf der Hauptkarte. Zentriert sich die Hauptkarte auf die (geografische) Position des Mausklicks? $\square_{ja}$ $\square_{nein}$
Wird in die Hauptkarte hineingezoomt? ..... $\square_{ja}$ $\square_{nein}$
Erhöht sich die Zoomstufe um genau eine Stufe? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Pan-ZoomIn-Prozess animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich der Zoomslider zeitgleich zum Pan-ZoomIn-Prozess um eine Zoomstufe nach oben? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird die skalierte Hauptkarte durch (nacheinander) neu gezeichnete Kacheln aktualisiert? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden der gesamten Karte die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

Testsuite C.3: Zooming über die Tastatur

C.3.1 Drücken Sie die Taste »+«. Wird in die Hauptkarte hineingezoomt? ..... $\square_{ja}$ $\square_{nein}$
Erhöht sich die Zoomstufe um genau eine Stufe? ..... $\square_{ja}$ $\square_{nein}$
Läuft der ZoomIn-Prozess animiert ab? ..... $\square_{ja}$ $\square_{nein}$

Bewegt sich der Zoomslider zeitgleich zum ZoomIn-Prozess um eine Zoomstufe nach oben? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird die skalierte Hauptkarte durch (nacheinander) neu gezeichnete Kacheln aktualisiert? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden der gesamten Karte die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.3.2 Drücken Sie die Taste »–«. Wird aus der Hauptkarte herausgezoomt? ..... $\square_{ja}$ $\square_{nein}$
Verringert sich die Zoomstufe um genau eine Stufe? ..... $\square_{ja}$ $\square_{nein}$
Läuft der ZoomOut-Prozess animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich der Zoomslider zeitgleich zum ZoomOut-Prozess um eine Zoomstufe nach unten? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich das rote Rechteck in der Übersichtskarte stets zentriert und zeitgleich mit dem Ausschnitt der Hauptkarte? ..... $\square_{ja}$ $\square_{nein}$
Wird während der Zoomanimation der sichtbare Bereich um die kleiner werdende Originalkarte mit neuen Karteninformationen erweitert? ..... $\square_{ja}$ $\square_{nein}$
Wird die skalierte Hauptkarte durch (nacheinander) neu gezeichnete Kacheln aktualisiert? ..... $\square_{ja}$ $\square_{nein}$
Symbolisiert die Statusleiste des Browsers nach vollständigem Laden der gesamten Karte die Beendigung des Ladevorgangs (z. B. durch die Meldung *done*)? ..... $\square_{ja}$ $\square_{nein}$

C.3.3 Drücken Sie zweimal kurz hintereinander die Taste »+«. Wird die Zoomstufe um genau *eine* Stufe erhöht? ..... $\square_{ja}$ $\square_{nein}$

C.3.4 Drücken Sie zweimal kurz hintereinander die Taste »–«. Wird die Zoomstufe um genau *eine* Stufe verringert? ..... $\square_{ja}$ $\square_{nein}$

Testsuite C.4: Panning über die *Panbar*

C.4.1 Klicken Sie auf den rechten Pfeil-Button. Bewegt sich der Ausschnitt der Hauptkarte nach Osten? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

C.4.2 Klicken Sie auf den unteren Pfeil-Button. Bewegt sich der Ausschnitt der Hauptkarte nach Süden? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

C.4.3 Klicken Sie auf den linken Pfeil-Button. Bewegt sich der Ausschnitt der Hauptkarte nach Westen? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

C.4.4 Klicken Sie auf den oberen Pfeil-Button. Bewegt sich der Ausschnitt der Hauptkarte nach Norden? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

Testsuite C.5: Panning über die Maus

C.5.1 Klicken Sie in die Mitte der Hauptkarte und ziehen Sie (mit gedrückter Maustaste) die Karte in eine beliebige Position innerhalb der Hauptkarte. Lassen Sie die Maustaste los. Bewegt sich die Hauptkarte beim Ziehen stets in die Richtung der aktuellen Mausposition? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$
Verbleibt die Hauptkarte und die Übersichtskarte nach dem Loslassen der Maustaste an der Position, an der die Taste losgelassen wurde? ..... $\square_{ja}$ $\square_{nein}$

Testsuite C.6: Panning über die Tastatur

C.6.1 Drücken Sie die Pfeiltaste →. Bewegt sich der Ausschnitt der Hauptkarte nach Osten? $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

C.6.2 Drücken Sie die Pfeiltaste ↓. Bewegt sich der Ausschnitt der Hauptkarte nach Süden? $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

C.6.3 Drücken Sie die Pfeiltaste ←. Bewegt sich der Ausschnitt der Hauptkarte nach Westen? $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

C.6.4 Drücken Sie die Pfeiltaste ↑. Bewegt sich der Ausschnitt der Hauptkarte nach Norden? $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

C.6.5 Drücken Sie die Taste »+«, warten Sie kurz bis die Hauptkarte sich aktualisiert hat und drücken Sie nocheinmal die »+«-Taste. Anschließend drücken Sie die Taste »Pos1«. Betrachten Sie nur die Auswirkung der Taste »Pos1«. Bewegt sich der Ausschnitt der Hauptkarte um ca. 75% der Kartenbreite nach
Westen? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

C.6.6 Drücken Sie die Taste »Ende«. Bewegt sich der Ausschnitt der Hauptkarte um ca. 75% der Kartenbreite nach
Osten? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

C.6.7 Drücken Sie die Taste »Bild ↑«. Bewegt sich der Ausschnitt der Hauptkarte um ca. 75% der Kartenhöhe nach
Norden? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Übersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

C.6.8 Drücken Sie die Taste »Bild ↓«. Bewegt sich der Ausschnitt der Hauptkarte um ca. 75% der Kartenhöhe nach
Süden? ..... $\square_{ja}$ $\square_{nein}$
Läuft der Panning-Vorgang animiert ab? ..... $\square_{ja}$ $\square_{nein}$
Passt sich der Ausschnitt in der Uebersichtskarte zeitgleich an die Änderungen der Hauptkarte an? ..... $\square_{ja}$ $\square_{nein}$

Testsuite C.7: Panning über die Übersichtskarte

C.7.1 Drücken Sie die Taste »+«, warten Sie kurz bis die Hauptkarte sich aktualisiert hat, und drücken Sie nocheinmal die »+«-Taste. Anschließend machen Sie einen Doppelklick auf eine beliebige Stelle in der Übersichtskarte *außerhalb* des roten Rechtecks. Betrachten Sie nur die Auswirkung des Doppelklicks. Bewegt sich das rote Rechteck animiert bis zur (geografischen) Position des
Doppelklicks? ..... $\square_{ja}$ $\square_{nein}$

Zentriert sich die Übersichtskarte auf die (geografische) Position des Doppelklicks? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich die Hauptkarte zeitgleich und animiert mit dem roten Rechteck? .... $\square_{ja}$ $\square_{nein}$

C.7.2 Klicken Sie auf den inneren Bereich des roten Rechtecks in der Übersichtskarte, und ziehen Sie (mit gedrückter Maustaste) den Ausschnitt an eine beliebige Stelle auf der Übersichtskarte. Lassen Sie die Maustaste los. Bewegt sich das rote Rechteck animiert bis zur aktuellen (geografischen) Position, an der die Maustaste losgelassen wurde? ..... $\square_{ja}$ $\square_{nein}$
Zentriert sich die Übersichtskarte auf die (geografische) Position, an der die Maustaste losgelassen wurde? ..... $\square_{ja}$ $\square_{nein}$
Bewegt sich die Hauptkarte zeitgleich und animiert mit dem Ausschnitt des roten Rechtecks? ..... $\square_{ja}$ $\square_{nein}$

D Creative Commons Lizenz

Commons Deed

Namensnennung-Weitergabe unter gleichen Bedingungen 2.0 Deutschland

Sie dürfen:

den Inhalt vervielfältigen, verbreiten und öffentlich aufführen

Bearbeitungen anfertigen

Zu den folgenden Bedingungen:

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

Weitergabe unter gleichen Bedingungen. Wenn Sie diesen Inhalt bearbeiten oder in anderer Weise umgestalten, verändern oder als Grundlage für einen anderen Inhalt verwenden, dann dürfen Sie den neu entstandenen Inhalt nur unter Verwendung identischer Lizenzbedingungen weitergeben.

- Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter die dieser Inhalt fällt, mitteilen.
- Jede dieser Bedingungen kann nach schriftlicher Einwilligung des Rechtsinhabers aufgehoben werden.

Die gesetzlichen Schranken des Urheberrechts bleiben hiervon unberührt.
Das Commons Deed ist eine Zusammenfassung des Lizenzvertrags in allgemeinverständlicher Sprache. Um den Lizenzvertrag einzusehen, besuchen Sie die Seite

http://creativecommons.org/licenses/by-sa/2.0/de

oder senden Sie einen Brief an Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

E Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Osnabrück, 1. Juni 2007

(Emanuel Schütze)

F CD-ROM

Der vollständige Inhalt der CD-ROM ist unter http://www.smartmapbrowsing.org/download.html verfügbar.