Automatische Erzeugung von JavaScript-Interaktionen für SVG-Karten

Urs Hensler

Diplomarbeit
am Geographischen Institut der Universität Zürich

Leitung:
Prof. Dr. Robert Weibel
Geographisches Institut der Universität Zürich

Betreuung:
Tobias Dahinden
Institut für Kartographie, ETH Zürich

Zürich, 2005

Zusammenfassung

Die vorliegende Diplomarbeit beschäftigt sich mit der automatischen Erzeugung von JavaScript-Interaktionen für SVG-Karten. Das Ziel besteht in der Realisierung eines Werkzeuges, welches Kartenautoren und -autorinnen bei der Programmierung von Interaktionen unterstützt bzw. diesen Vorgang vereinfacht und beschleunigt. Neben der Implementierung von ISIMAP wird eine Beurteilung des Werkzeuges vorgenommen. Die Untersuchung zeigt, dass sich durch den Einsatz des Werkzeuges die Produktivität in diesem Prozess steigern lässt und dass das Werkzeug Kartographen und Kartographinnen ermuntert, vermehrt Interaktionen in ihre Karten einzubauen.

Die Ausgangslage für die Fragestellung bildet die zunehmende Beliebtheit von SVG. Das Vektorformat SVG ist zur Zeit die am besten geeignete Methode, um qualitativ hochstehende und interaktive Kartographie im Internet zu betreiben. Auf der einen Seite besticht SVG durch seine graphischen Möglichkeiten und durch die umfangreiche Palette an Hilfsmittel, die für die Erstellung und Bearbeitung von SVG-Graphiken zur Verfügung steht. Auf der anderen Seite ist die Verwendung von SVG auch mit Nachteilen verbunden. Interaktionen müssen (immer noch) mühsam von Hand implementiert werden. Die manuelle Erstellung von Interaktionen setzt grosse Kenntnisse einer geeigneten Programmiersprache, z. B. JavaScript, voraus und ist ein zeitraubender Vorgang. Die vorliegende Arbeit soll einen Betrag zur Überwindung dieser gegenwärtig aktuellen Schwierigkeit in der Web-Kartographie leisten.

Als Grundlage für die Realisierung des Werkzeuges dient eine Anforderungsspezifikation. Diese ist in Zusammenarbeit mit Personen aus dem kartographischen Bereich erarbeitet worden. So ist gewährleistet, dass das Werkzeug auf ihre Bedürfnisse zugeschnitten ist. Die Anforderungen werden in einem zweiten Schritt in ein Computerprogramm namens ISIMAP umgesetzt. Dieses ermöglicht den Benutzenden statische SVG-Karten mit grundlegenden Interaktionen zu versehen. Dazu zählen hauptsächlich Interaktionen zum Vergrössern, Verkleinern und Verschieben des Kartenbildes sowie zur graphischen Hervorhebung von Kartenelementen.

Für die Beurteilung von ISIMAP ist eine Befragung unter potentiellen Benutzenden durchgeführt worden. Die Resultate dieser Evaluation zeigen ein überwiegend positives Urteil. ISIMAP ist ein sehr gutes Werkzeug für die Erzeugung von JavaScript-Interaktionen für SVG-Karten. Die einfach zu erlernende Funktionalität erlaubt es auch programmiertechnischen Laien, Karten mühelos und in kurzer Zeit mit Interaktivität zu versehen.

Abstract

The objective of this master thesis is the automatic generation of JavaScript interactions for SVG maps. Firstly a tool with the name ISIMAP will be developed for assisting authors of map in programming of interactions. In other words it should simplify and accelerate this process. Furthermore the tools efficiency is evaluated. It shows that the use of the implemented tool increases the productivity in the mentioned process and that the tool animates map authors to use interactive elements with their maps.

One can notice an increasing popularity of SVG. The use of the vector format SVG is currently the most suitable method to draw qualitative maps for the publication in the Internet. Its advantages are the rich graphic possibilities and the number of tools supporting the creation and the editing of SVG graphics. On the other hand it's disadvantage lies in the exhausting manual coding of interactive elements. This requires a good knowledge of a programming language like JavaScript and plenty of time. This study contributes to overcome these difficulties in web cartography.

The base for the development of this tool are the user requirements. These were compiled together with specialists in cartography. So, the tool will be specified to them. ISIMAP allows for adding basic interactions to SVG maps. These are namely zooming and panning the map and highlighting specific map elements.

The evaluation of ISIMAP is based on a survey with potential users. The results shows generally a positive rating. ISIMAP is a very good tool for the generation of JavaScript interactions for SVG maps. Its functionality is easy to learn and it allows people with even low programming skills to publish interactive maps in the Internet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

API Application Programming Interface.

CSS Cascading Style Sheets.

DIN Deutsches Institut für Normung.

DOM Document Object Model.

GIF Graphics Interchange Format.

GIS Geographisches Informationssystem.

GVT Graphic Vector Toolkit.

HTML Hypertext Markup Language.

ISO International Standardisation Organization.

JPEG Joint Photography Experts Group.

MoSCoW Must Should Could Would.

PDF Portable Document Format.

PNG Portable Network Graphic.

SMIL Synchronizised Multimedia Integration Language.

sRGB Standard Red Green Blue.

SVG Scalable Vector Graphics.

TIFF Tagged Image File Format.

W3C Word Wide Web Consortium.

WWW World Wide Web.

XML Extensible Markup Language.

1 Einleitung

1.1 Problemstellung

SVG (Scalable Vector Graphics) ist momentan der State-of-the-art Vektorstandard für das World Wide Web (WWW) und derzeit die am besten geeignete Methode, um qualitativ hochstehende und interaktive Kartographie in Internet zu betreiben [Dahinden et al., 2003]. Da es sich um ein offenes und gut dokumentiertes Format handelt, sind zahlreiche Methoden und Hilfsmittel entwickelt worden, um auf einfache Art und Weise SVG-Dokumente zu generieren [Neumann und Winter, 2001]. Diesem Umstand verdankt SVG die zunehmende Beliebtheit bei Kartographen und Kartographinnen. Sie verfügen nun über die Möglichkeit, Karten im Internet mit einer befriedigenden kartographischen Qualität zu publizieren.

In einer für die Benutzenden geeigneten Darstellungsform von räumlichen Daten auf dem Web sind Interaktionen ein wesentlicher Bestandteil. Den Nachteilen von Graphiken am Bildschirm gegenüber Karten im Printmedium, wie der geringeren Grösse der nutzbaren Bildfläche oder der weit geringeren Auflösung [Brunner, 2001], kann so entgegengewirkt werden. Interaktionen können aber auch zur verbesserten Informationsvermittlung eingesetzt werden [Dickmann, 2002]. Kartographische Anwendungen reichen vom einfachen Ein- und Ausblenden von Elementen und Ebenen, über das Anzeigen von Objektdaten beim Überfahren von Kartenelementen mit der Maus bis hin zu Applikationen, die den Anwendenden zum Beispiel einfache Digitalisierschritte erlauben [Neumann und Winter, 2003a].

Die Möglichkeiten zur Erstellung von Interaktionen in Webseiten lassen sich mit Skriptsprachen deutlich steigern [Dickmann, 2002]. Um SVG-Graphiken mit Interaktivität zu versehen, wird am häufigsten die Skriptsprache JavaScript verwendet [Neumann und Winter, 2003b]. Mit ihr können sämtliche SVG-Elemente am Client-Computer neu erstellt und verändert werden. JavaScript ist im Vergleich zu anderen Programmiersprachen relativ einfach zu erlernen [Dickmann, 2002]. Graphisch orientierten Kartographen und Kartographinnen ist der Aufwand zur Einarbeitung in eine Programmiersprache trotzdem zu gross. Ausserdem stellt die manuelle Programmierung von Interaktionen auch bei guten JavaScript-Kenntnissen einen nicht zu unterschätzenden Zeitaufwand dar. Kartenautoren und -autorinnen verzichten deshalb oftmals lieber auf den Mehrwert, welchen Interaktionen mit sich bringen, und veröffentlichen ihre Karten in einer traditionellen, statischen Form.

1.2 Zielsetzung

Die Erstellung von interaktiven SVG-Karten ist immer noch mit Schwierigkeiten verbunden, insbesondere die mühsame und zeitaufwändige Programmierung der Interaktionen bereitet

vielen Kartenautoren und -autorinnen Probleme. Es besteht ein Bedürfnis nach einem Werkzeug zur Vereinfachung dieses Herstellungsteilprozesses. Die vorliegende Arbeit widmet sich dieser Problematik. Es wird ermittelt und festgehalten, wie ein geeignetes Hilfsmittel aussehen könnte. In einem zweiten Schritt werden die Ergebnisse anhand eines Prototypen umgesetzt. Abschliessend wird eine Beurteilung vorgenommen und die folgende Hypothese untersucht: Mit dem implementierten Werkzeug kann die Produktivität bei der Erstellung von Interaktionen gesteigert werden und programmiertechnisch ungeübte Benutzer und Benutzerinnen werden ermuntert, vermehrt JavaScript-Interaktionen in ihre SVG-Karten einzubauen.

Aus dieser allgemeinen Formulierung werden folgende konkreten Ziele abgeleitet:

- Es soll eine Zusammenstellung der am meisten eingesetzten Interaktionen aufgestellt werden.
- Es soll ein Werkzeug zur automatischen Erzeugung von JavaScript-Interaktionen für SVG-Karten erstellt werden.
- Das Werkzeug soll zu einer Vereinfachung bei der Erstellung der grundlegenden JavaScript-Interaktionen führen.
- Das Werkzeug soll die Erstellung von JavaScript-Interaktionen beschleunigen.
- Das Werkzeug soll benutzerfreundlich sein.
- Das Werkzeug soll evaluiert werden.

Da das Werkzeug für den Einsatz im kartographischen Bereich vorgesehen ist, werden Fachleute mit dem entsprechenden Wissen in den Herstellungsprozess mit einbezogen. Ihre Bedürfnisse und Ansprüche werden mit einer geeigneten Methode ermittelt. Eine Übersicht der gestellten Anforderungen ist im Abschnitt 4.1 zu finden. Diese sind als weitere Ziele für die Implementierung des Prototypen zu betrachten.

1.3 Aufbau der Arbeit

1.3.1 Gliederung der Arbeit

Die Arbeit ist in sechs Abschnitte eingeteilt. Nach der Einleitung wird in Kapitel 2 die Bedeutung von Interaktionen in der Web-Kartographie besprochen. Nach einem kurzen geschichtlichen Überblick wird der aktuelle Stand der Technik vorgestellt. Die grundlegenden Interaktionen in SVG-Karten und die Vorgehensweise bei der Erstellung von diesen ist ein weiterer Bestandteil des Kapitels. Aus diesen Ausführungen wird die Idee für das Werkzeug abgeleitet.

In Kapitel 3 wird das Grundlagenwissen zu den eingesetzten Techniken vermittelt. Nach einer Einführung zum Graphikformat SVG und zur Skriptsprache JavaScript wird der Fokus auf das Zusammenspiel dieser beiden Techniken gerichtet. Des Weiteren wird Java, die für die Programmierung des Werkzeuges verwendete Sprache, und Batik, die eingesetzte Java-Bibliothek zur Manipulation von SVG-Dokumenten, kurz beschrieben.

Nach den theoretischen Grundlagen wenden wir uns dem praktischen Teil zu. Die Realisierung des Computerprogramms ISIMAP steht im Mittelpunkt von Kapitel 4. Als erstes werden

die Anforderungen der Kartographen und Kartographinnen präsentiert. Anschliessend wird auf das Design und die Implementierung eingegangen. Der Schwerpunkt liegt dabei auf der Erklärung von ausgewählten Konzepten zur Umsetzung der Anforderungsspezifikation. Der Abschluss des Kapitels bildet die Beschreibung des JavaScript-Codes für die mit ISIMAP erstellbaren Interaktionen.

Das Kapitel 5 widmet sich der Beurteilung des implementierten Werkzeuges. Neben der formalen Überprüfung der Umsetzung der Anforderungsspezifikation wird eine Evaluation des Computerprogramms durch Fachleute aus dem kartographischen Bereich durchgeführt. Ein wesentlicher Bestandteil in diesem Abschnitt ist die Diskussion der Resultate.

Die Schlussfolgerungen aus dieser Arbeit werden in Kapitel 6 gezogen. Dies umfasst die Überprüfung der Hypothese und der Ziele sowie eine Einordnung der Arbeit im Kontext der interaktiven Web-Kartographie. Einen Ausblick auf mögliche weiterführende Arbeiten wird im letzten Anschnitt der Arbeit gegeben.

1.3.2 Inhalt der CD

Auf der beiliegenden CD sind alle für diese Arbeit relevanten digitalen Dokumente zu finden. Die Dateien sind in drei Verzeichnisse eingeteilt:

- *Diplomarbeit*: Hier ist die vorliegende Arbeit als pdf-Dokument abgelegt.
- *isiMap*: Hier sind die Bestandteile des erstellten Werkzeuges gespeichert. Sie sind in drei Unterverzeichnisse eingeordnet:
  - *bin*: Die Datei *isiMap.jar* ist das kompilierte Programm. Die restlichen Dateien in diesem Verzeichnis gehören zur Batik-Bibliothek.
  - *doc*: Das Javadoc ist die Dokumentation des Quellcodes.
  - *src*: Der Quellcode von ISIMAP liegt in diesem Verzeichnis.
- *Tutorial*: Das Tutorial bietet eine Hilfestellung für die Benutzenden von ISIMAP.

2 Interaktive Web-Kartographie

2.1 Stand der Technik

2.1.1 Geschichtlicher Hintergrund der Web-Kartographie

Mit der Einführung des Wold Wide Webs in der ersten Hälfte der 1990er Jahre und den graphisch orientierten Browsern hat die Attraktivität des Internets sprunghaft zugenommen. Kartographen und Kartographinnen unternahmen schon früh erste Versuche ihre Produkte im Internet zu veröffentlichen. Sie waren nun erstmals in der Lage raumbezogene Daten unmittelbar zu visualisieren und interaktiv beeinflussbare kartographische Darstellungen zur Informationsvermittlung zu nutzen. [Dickmann, 1999]

Ein erste Anwendung im Internet war der 1993 vom Xerox's Palo Alto Research Center entwickelte *Map Viewer*. Dieser generierte sehr einfache Karten aus öffentlichen Daten und beinhaltete ein interaktive Benutzeroberfläche [Plewe, 1997]. Eine nach den eigenen Wünschen erstellte kartographische Darstellung liess sich als GIF-File übertragen. Obwohl die Karten nur aus Liniendaten bestanden, jegliche textlichen oder topographischen Informationen fehlten, wurde dieses System ein grosser Erfolg [Dickmann, 1999]. Die Idee der Veröffentlichung von Karten im Internet wurde schnell in anderen Ländern aufgenommen, und innerhalb wenigen Monaten wurden in vielen Ländern nationale Karten online gestellt [Plewe, 1997].

In den folgenden Jahren wurden mehrere Projekte für die Verbreitung von räumlichen Daten im Internet gestartet. 1995 wurden mehrere Internet Map Server entwickelt, die das Prinzip des *Map Viewers* erheblich ausbauten [Dickmann, 1999]. Das wohl bekannteste System dieser Generation von Map Servern bildete der *TIGER Map Service* [38] des U. S. Census Bureau [Dickmann, 1997]. Ziel dieser Anwendung war es, detaillierte Strassen- und Themenkarten der USA bereitzustellen, die von Internetnutzern kostenfrei verwendet werden konnten. Die Benutzenden dieses frühen kartographischen Informationssystems konnten verschiedene Themenlayer selber auswählen und sich die Karte in Form eines GIF-Bildes anzeigen lassen. Neben weiteren Behörden traten nun auch verstärkt kommerzielle Unternehmen wie MapQuest [18] oder MapBlast [21] in den Vordergrund, die selber aufgenommene Datensätze in verschiedenen kartographischen Systemen zur Lokalisierung von Strassen verwendeten. [Dickmann, 1999]

Die nächsten Schritte der Entwicklung zielten darauf ab, eine weitreichende GIS-Funktionalität im Internet verfügbar zu machen. Ein Vertreter dieser Klasse war *GRASSLinks*, das an der Universität von Berkeley entwickelt wurde. Zur Online-Bearbeitung wurden neben flächenbezogenen Grundkarten in Form von Rasterdarstellungen auch Layer mit Punkt- und Linieninformationen angeboten. Als GIS-Funktionen standen z. B. Flächenberechnungen oder Reklassifizierungen zur Verfügung. Die Durchführung der Operationen wurde durch die ge-

ringen Einflussmöglichkeiten der Benutzenden auf die kartographischen Gestaltungsvorgaben des Servers sowie durch den systembedingten hohen Zeitaufwand stark eingeschränkt. Neben diesem Beispiel wurde eine grosse Anzahl an ausgereiften kommerziellen Online-GIS entwickelt, z. B. *MapGuide* [3] von Autodesk oder *Internet Map Server* [8] von ESRI. Auch in diesen Produkten erfolgte die Übermittlung der Abfrageresultate mit Hilfe von GIF- oder JPEG-Bildern. [Dickmann, 1999]

Auch die Darstellungsform der Bildschirmkarten hatte infolge der technischer Entwicklungen interessante Erweiterungen erfahren. So bot zum Beispiel das Programm *Descartes* interaktive Werkzeuge zur graphisch unterstützten Auswertung abgerufener Karten an. Die Möglichkeiten erstreckten sich von der Veränderung der Einfärbung, über die cursorgesteuerte Hervorhebung von Flächen bis zur Einblendung von zugeordneten Sachdaten. Die Bedingungen für eine visuelle Datenexploration konnten dadurch entscheidend verbessert werden. [Dickmann, 1999]

Bei der Handhabung der vorgestellten Systeme machte sich ein Nachteil besonders bemerkbar. Die langen Übertragungszeiten vom Server zum Browser bei hochauflösenden Bildern, Multimedia-Elementen oder komplexen Abfragesituationen waren im Internet nicht zufriedenstellend. Zum Teil wurde daher die Strategie verfolgt, einen Teil der Rechnerleistung auf die lokalen Rechner (clients) zu übertragen. Jedes Zoomen oder Verschieben der Karte bedeutet schliesslich eine neue Abfrage an den Server, der eine neue Karte generieren und zum Client zurücksenden muss. Die Entwicklung der Java- und der Plugin-Technologien hat es jedoch ermöglicht, dieses Defizit aufzufangen. [Dickmann, 1999]

Applets sind kleine Java-Programme die direkt im HTML-Dokument eingebetet sind und automatisch auf den Client-Rechner geladen werden, sobald die Seite aufgerufen wird. Sie bleiben nicht dauerhaft auf dem Client-Rechner und müssen bei jeder erneuten Nutzung abermals geladen werden [Dickmann, 1999]. Für die Web-Kartographie eignen sich Applets insbesondere durch ihren grösseren Funktionsumfang, ihre flexiblen Darstellungsmöglichkeiten und die Portabilität von Java [Cecconi et al., 2000]. Insbesondere *Java2D* wurde für die qualitativ hochwertige und einfach Programmierung von Vektorgraphiken entworfen [Neumann und Winter, 2003b]. Anwendungen sind zum Beispiel das oben erwähnte Programm *Descartes*, das *THMApplet* [Winkler, 2000] oder *EVisA* [Shenton, 2000].

Eine weitere Variante besteht darin, mit Plugins zu arbeiten. Plugins sind Software-Bausteine und erweitern die Funktionalität von Programmen. Sie müssen einmal auf dem Client installiert werden und stehen anschliessend dauerhaft zur Verfügung. Durch die Verwendung von Plugins können einzelne Arbeitsschritte von Server auf den Client verlagert werden. Dies erhöht die Übertragungsgeschwindigkeit. [Dickmann, 1999] In kartographischer Hinsicht ist auch von Bedeutung, dass mit Hilfe von Plugins auch die Verarbeitung von Vektorgraphiken in Browsern möglich wurde.

2.1.2 Aktueller Stand

Die Möglichkeiten des Internets für die dynamische und interaktive Weitergabe kartographisch aufbereiteter Geoinformation hat mittlerweile grosse Bedeutung erlangt. Es sind die verschiedensten Formen kartographischer Darstellungen anzutreffen, und die **Zahl von Web-Karten** ist in den letzten zehn Jahren immens gestiegen [Dickmann, 2004]. Peterson [Peter-

son, 1999] ging bereits 1999 von einer Nutzung von 40 Mio. Karten täglich im Internet aus, und vier Jahre später revidierte er seine Schätzung der täglichen Nutzung auf 200 Mio. täglich [Peterson, 2003]. Der Wert dürfte sich auch weiterhin rasant zunehmen.

Obwohl die Nutzung gedruckter Karten ein fester Bestandteil im Alltag vieler Leute ist, besteht offensichtlich ein Bedürfnis nach Karten im Web. Dieses Phänomen liegt in **den Vorteilen von Web-Karten** gegenüber gedruckten Karten begründet. Im Internet sind Informationen rund um die Uhr verfügbar. Die Benutzenden können im Prinzip jederzeit von zu Hause oder am Arbeitsplatz auf eine enorme Fülle von räumlichen Informationen zugreifen. Durch die Mobilfunktechnologie erweitert sich dieser Raum sogar um alle Orte, an denen eine Funknetzverbindung besteht. Dank dieser Verfügbarkeit müssen sie sich nicht mehr in einen Laden mit begrenzten Öffnungszeiten zur Suche einer für ihre Zwecke geeigneten Karte begeben. Ein anderer Aspekt der Verfügbarkeit und ein grosser Vorteil für die Betrachter und Betrachterinnen ist die immer noch kostenfreie Benutzung der Karten im Internet. [van Elzakker, 2001a] Eines der erheblichsten Probleme der traditionellen Kartographie ist die Nachführung von Karten. Bedingt durch den langen Produktionsprozess waren die Informationen einer gedruckten Karte oder einer CD-ROM-Karte beim Verkauf manchmal bereits veraltet. Das WWW macht es möglich, wirklich aktuelle Daten bereitzustellen. [van Elzakker, 2001a] Beispiele dafür sind Webseiten mit aktuellen Wetterkarten [19] oder mit Karten zur aktuellen Verkehrslage [23]. Ausserdem ist das Drucken auf Papier teuer, speziell in grossen Formaten und in Farbe. Viel günstiger ist es, Farbgraphiken im Web zu publizieren. Unter Berücksichtigung der Kosten für den Versand und den Vertrieb von gedruckten Karten werden die Kostenvorteile der online veröffentlichten Karten noch offensichtlicher. [Peterson, 2003]

Bemerkenswert ist nicht nur die grosse Anzahl von Karten im Internet, sondern auch die unterschiedlichen **Gestaltungsvarianten** von diesen. Allen gemeinsam ist, dass sie in ein interaktives Umfeld, d. h. in die Webseite, eingebunden sind. Sie weisen aber unterschiedliche Formen von Interaktivität mit der Karte bzw. den Kartenelementen selbst auf [Gartner, 2003]. Kraak [Kraak, 2001] unterscheidet die Karten in diesem Zusammenhang nach den Zustandsmöglichkeiten und trennt zwischen statischen und interaktiven Karten. Peterson [Peterson, 2003] erweitert dieses Modell um die Kategorie der animierten Karten. Statische Karten präsentieren wie gedruckte Karten nur eine Ansicht der Informationen. Diese technisch einfach herstellbaren Karten sind in den meisten Fällen gescannte Papieroriginale und werden auf dem Bildschirm in Form von Bitmaps visualisiert [Dickmann, 2000a]. Eine umfangreiche Kartensammlung von statischen Karten stellt zum Beispiel die University of Texas [27] zur Verfügung. Interaktive Karten erlauben es den Nutzenden die Darstellung auf Wunsch zu verändern (siehe Abschnitt 2.2). In animierten Karten wird schliesslich eine Serie von Karten mit dem Ziel der Darstellung der zeitlichen Veränderung eines Themas aneinander gereiht. Ein Beispiel einer animierten Karte erstellte Isakowski [13].

Ein weiteres Kriterium zur Einteilung von Karten im Internet ist das verwendete Format. Es kann grundsätzlich zwischen **Raster- und Vektorformat** unterschieden werden. Bilder im Rasterformat sind aus einem Gitter von farbigen oder grauen Bildpunkten (sogenannten Pixeln) aufgebaut [Peterson, 2003]. Dem Vektorformat liegt ein objektorientierter Ansatz zu Grunde, es werden die Merkmale der graphischen Figuren gespeichert. Gegenüber Vektorgraphiken weisen Rastergraphiken jedoch entscheidende Nachteile auf [Winter, 2000, van Elzakker, 2001b]:

- Massstabsabhängikeit
- ungenügende Druckqualität
- fehlende Objekthierarchie
- Abhängigkeit des Datenvolumens von der Grösse des Ausschnittes statt von der Informationsdichte
- fehlende Möglichkeiten zur Textsuche und zur Indexierung durch Suchmaschinen

Die Darstellung von Vektorgraphiken in Browsern ist jedoch erst seit einigen Jahren möglich und die neuen Formate setzen sich nur langsam gegen die weit verbreiteten Rasterformate durch. Auf die zwei leistungsstärksten Vertreter, SVG und Flash, wird im nächsten Abschnitt eingegangen.

Der Schwerpunkt bei der Entwicklung von Web-Karten ist in den Anfängen eindeutig auf die Erweiterung der Funktionalität gesetzt worden. Auf die graphische Qualität wurde und wird dagegen oft nicht genügend Wert gelegt und gegenwärtig wird im Web nur wenig vom kartographischen Potential ausgeschöpft [van Elzakker, 2001a]. Die Gründe für die **mangelnde graphische Qualität** sind zum einen im technischen Bereich zu suchen. Schon in einer frühen Generation von Browsern konnten Rasterbilder dargestellt werden. Die allermeisten Webkarten werden deshalb auch heute noch mit Hilfe dieser, für die Kartographie an sich ungeeigneten Technik veröffentlicht [Räber und Jenny, 2003]. Ein anderer Aspekt ist, dass die Kartenautoren und -autorinnen keine vollständige Kontrolle über die schlussendliche Anzeige besitzen. Auch wenn sie ihre Karten in einem plattform-unabhängigen Format (GIF, JPEG oder PDF) speichern, werden die Graphiken nicht für alle Benutzenden gleich dargestellt [van Elzakker, 2001b]. Die Effekte des kartographischen Designs können abhängig von den verschiedenen Konfigurationen der Ausgabegeräte unterschiedlich erscheinen. Nicht zu unterschätzen ist auch die Verfügbarkeit von Hilfsmitteln. Heutzutage kann jeder Karten gestalten und im Web verbreiten, auch wenn er nicht über das nötige kartographische Wissen verfügt [van Elzakker, 2001b]. Viele Geographische Informationssysteme (GIS) erlauben eine einfache Veröffentlichung von Karten im Internet. GIS sind jedoch in erster Linie für die Analyse von räumlichen Daten und nicht für deren Präsentation geschaffen. Mit GIS erstellte Karten zeigen oft mangelhafte Symbolisierungen, eine schlechte Schriftplatzierung oder eine Überlagerung unharmonisierter Datensätze [Räber und Jenny, 2003].

Räber und Jenny [Räber und Jenny, 2003] plädieren deshalb für eine mediengerechte Kartengraphik. Sie stellen fest, dass sich im Internet nur selten gelungene Karten finden lassen, welche die graphischen Möglichkeiten des jungen Mediums ausschöpfen. Speziell für das Web erstellte Karten genügen noch zu selten kartographischen Ansprüchen. Van den Worm [van den Worm, 2001] sieht deshalb die **Herausforderung der Web-Kartographie** auch in der Kombination von Funktionalität mit visueller Attraktivität und einer Gestaltung, die dem Medium Internet gerecht wird.

2.1.3 Die Rolle von SVG

Im Jahre 2000 wurde SVG als offizieller Standard vom World Wide Web Consortium (W3C) verabschiedet und zwei Jahre später in überarbeiteter Form als Version 1.1 veröffentlicht. In

der Spezifikation vom W3C [41] wird das Format folgendermassen definiert:

> „SVG ist eine Sprache zur Beschreibung von zweidimensionalen Graphiken in XML. SVG erlaubt drei Arten von graphischen Objekten: Vektorgraphiken, Bilder und Text. [...] SVG-Zeichnungen können interaktiv und dynamisch sein.“
> (Übersetzung des englischsprachigen Originals)

Diese Definition lässt das Potential von SVG erahnen. In den letzten Jahren wurden mehrere Untersuchungen durchgeführt, die sich mit der Eignung von SVG für die Web-Kartographie beschäftigen. Insbesondere die Arbeiten von [Winter, 2000], [Neumann und Winter, 2003a] und [Wilfert, 2003] dringen tief in die Thematik ein.

Die relevanten Eigenschaften von SVG in Bezug auf die Web-Kartographie werden von Neumann und Winter [Neumann und Winter, 2003a] zusammengefasst. Die wichtigsten Punkte seien an dieser Stelle erwähnt.

Vektorformat

Vektorformate bieten bei einem relativ kleinen Speicherbedarf qualitativ sehr hochwertige Darstellungsmöglichkeiten. Die Daten können theoretisch in beliebiger Vergrösserungsstufe ohne graphischen Qualitätsverlust dargestellt werden. Die Elemente der Karte können objektweise bearbeitet werden. Dies ermöglicht die Verknüpfung von Sachdaten mit einzelnen Objekten und die präzise Steuerung von Interaktionen.

Renderingkonzept (Bildschirmaufbau)

SVG-Graphiken werden nach dem „painter's algorithm“ dargestellt, was vereinfacht bedeutet, dass sich überlappende Bereiche „übermalt“ werden oder entsprechend der Transparenz-Werte die darunter liegenden Bildschirm-Pixel durchscheinen lassen.

Koordinatensysteme und Transformationen

SVG verwendet ein kartesisches Koordinatensystem, dessen Ursprung sich in der linken oberen der Zeichenfläche befindet. Bei rechtwinkligen Kartendarstellungen muss deshalb die y-Achse invertiert werden, was jedoch einfach mit einer 3x3-Matrixoperation zu bewerkstelligen ist. Diese können auch für elementare geometrische Transformationen eingesetzt werden.

Formatierungen

In SVG werden sehr umfangreiche Möglichkeiten zur Darstellung angeboten. Farben werden im sRGB-Standard angegeben. Weitere Optionen sind u. a. Füllmuster, Farbverläufe, Transparenzen oder Strichtypen. Mit Hilfe von Stylesheets (CSS-Objekte) lassen sich Darstellungen effizient verwalten und ändern.

Geometrische Grundelemente

Die geometrischen Grundelemente Rechteck, Kreis, Ellipse, Linie, Polylinie und Polygon sind als eigene Objekte definiert. Mit dem flexiblen Path-Objekt kann man offene und geschlossene Linienobjekte zeichnen, die sowohl aus absoluten als auch relativen Koordinaten bestehen können. Um allen Anforderungen gerecht zu werden, können auch kubische und quadratische Bezierkurven sowie elliptische Kurvenelemente einen Pfad bilden.

Texte

Texte können in allen gängigen Formaten angezeigt werden. Ein herausragendes Merkmal ist, dass sich Textelemente von Suchmaschinen indizieren lassen. Schriften lassen sich direkt in der SVG-Datei einbetten.

Animationen

Mit Animationen werden zeitliche Änderungen dargestellt. Die verwendete Technik beruht auf der bekannten Metapher der Zeitlinien. Dabei werden sogenannte „key frames“, fixe Zustände im zeitlichen Ablauf, definiert. Die Zwischenschritte werden bei der Ausführung einer Animation laufend berechnet. Die vorhandenen Variablen zur Steuerung der Animationen gewähren einen kontrollierten Ablauf. Animationen können entweder mit Hilfe von SMIL-Anweisungen oder mit Skriptsprachen, z. B. JavaScript, realisiert werden.

Interaktionen

Hyperlinks können dazu verwendet werden, auf andere Dateien oder auf beliebige Objekte in des Dokumentes zu verweisen. Komplexere Anwendungen können mit Hilfe von Skriptsprachen oder Java-Applets programmiert werden. Prinzipiell lassen sich alle SVG-Elemente und alle Attribute der SVG-Elemente über das DOM (siehe Abschnitt 3.3) ansprechen und manipulieren.

Metadaten

Metadaten ermöglichen, Informationen über das Dokument (z. B. Autor, Titel, Publikationsdatum, Kurzbeschreibung) in strukturierter Form abzulegen. Sie können ebenso wie Fremdcode (z. B. JavaScript) in die SVG-Datei eingebettet werden.

Erweiterbarkeit

Da SVG ein XML-Dialekt ist, können beliebige auf XML beruhende Standards in SVG eingebettet werden. Auf diese kann unbeschränkt zugegriffen werden.

SVG verfügt über alle Vorteile, um den Datenaustausch im Allgemeinen und die Visualisierung im Speziellen zu fördern [Dickmann, 2001]. Neumann und Winter [Neumann und Winter, 2003a] gehen sogar einen Schritt weiter. Sie kommen zu folgendem Schluss:

> „Bis dato war jeder ernsthafte Versuch, Kartographie im Internet zu betreiben, an sehr strikte anzeigetechnische Vorgaben gebunden, die graphisch hochqualitative Kartographie unmöglich machten. Das Gros der Kartographen an dieser Front war vorrangig damit beschäftigt Notlösungen zu kreieren, anstatt sich den eigentlichen kartographischen Inhalten zu widmen. SVG erlaubt es nun erstmals die Last der Sorgen um die graphische Ausgabe zu verringern, und sich mehr den Interaktionen, die die Bildschirmkartographie immer noch mit sich bringt, zuzuwenden.“

Für eine umfassende Beschreibung von SVG sei auf [Fibinger, 2001], [Eisenberg, 2002] und [Watt und Lilley, 2002] verwiesen. Eine umfassende Sammlung von Web-Karten im SVG-Format ist auf der Web-Seite von carto:net [5] zu finden.

Zur Darstellung von SVG-Dokumenten muss in den meisten Browsern das Plugin *SVG Viewer* [1] von Adobe installiert sein. Im Browser *Opera* [28] ab der Version 8 sowie in speziellen Version von *Mozilla* [36] und *Firefox* [36] können SVG-Graphiken direkt, d. h. ohne Plugin, angezeigt werden. Alternativ dazu lässt sich die Java-Bibliothek Batik [9] zur Einbindung von SVG in Webseiten oder als eigenständige Applikation verwenden.

Als weiteres Vektorformat für Graphiken im Internet sei an dieser Stelle noch das weit verbreitete Format ShockWave Flash von Macromedia erwähnt. Dieses konnte sich insbesondere in der Werbe- und Multimedia-Branch durchsetzen [Neumann und Winter, 2003a]. Ullrich [Ullrich, 2003] zeigt mit seiner Arbeit, dass auch mit Flash ansprechende Web-Karten erstellt werden können. Und auch in einigen kommerziellen, kartographischen Unternehmen, z. B. Intermaps [12], wird Flash eingesetzt.

Insbesondere die Möglichkeiten zum Schutz des Quellcodes und der räumlichen und thematischen Daten ist ein ausschlaggebendes Argument für Flash und gegen das Klartextformat SVG. Es hat sich jedoch gezeigt, dass die Möglichkeiten von SVG diejenigen von Flash erweitern, SVG im Gegensatz zu Flash auf offenen Standards basiert und wesentlich besser mit anderen Technologien integrierbar ist [Dahinden et al., 2003]. Für einen Vergleich der beiden Formate sei auf [25] und [Hurni et al., 2001] verwiesen.

2.2 Interaktive Karten

Interaktive Karten erlauben es den Benutzenden mit einer Karte zu interagieren und Entscheidungen über den Inhalt und das Design selber zu treffen [Cartwright, 2003]. Sie sind gekennzeichnet durch eine graphische Bedienungsoberfläche (auch Benutzerschnittstelle genannt), die überlicherweise aus systematisch strukturierten graphischen Schaltflächen (Menü), einem Zeigergerät (Maus) und einer nahezu unverzüglichen Bildschirmanzeige besteht [Hake und Grünreich, 2002].

Für die Verwendung von Interaktionen gibt es viele Gründe. Eine erste Einsatzmöglichkeit besteht in der **Überwindung von bildschirmspezifischen Problemen**. Brunner [Brunner, 2001] legt die Probleme von Kartengraphiken am Bildschirm ausführlich dar. Die beträchtlichsten Einschränkungen sieht er in der limitierten Grösse der nutzbaren Fläche und der geringen Auflösung. Zur Behebung dieser Probleme können, neben der richtigen Wahl des Graphikformates, Interaktionen eingesetzt werden.

Zur **Vermittlung von ergänzenden und weiterführenden Informationen** in Form von Fotos, Texten, Graphiken sowie Bild- oder Tonsequenzen können ebenfalls Interaktionen eingesetzt werden [Cartwright, 2003]. Interaktive Karten können als graphische Schnittstelle, mit der Auskunftssysteme online betrieben werden, dienen. Ein Beispiel hierfür ist die Verbreitung von Reiseinformation [30] über ein Land oder eine Stadt [Dickmann, 2000a].

Gartner [Gartner, 2003] spricht einen weiteren Grund zur Notwendigkeit des Einsatzes von Interaktivität in Webkarten an. Die Attraktivität von Inhalten im Internet ist geprägt von zunehmend bildhaften, interaktiven, animierten und verlinkten Informationspräsentationen und weniger von textlichen, linear zu lesenden und statischen Darstellungen. Diese Tatsache führt dazu, dass das Anklicken eines Textes oder eben einer Karte als **Selbstverständlichkeit** von den Benutzenden gefordert wird. Interaktivität ist deshalb eine notwendige Eigenschaft, unabhängig davon, ob damit tatsächlich Informationserschliessung oder sonstige Vorteile ermöglicht werden. Karten in einer interaktiven und animierten Form können aber auch in einem grösseren Ausmass Leute für Karten begeistern, als dies gedruckte Erzeugnisse vermögen [Peterson, 2003].

In Abschnitt 2.1.1 wurde aufgezeigt, dass Interaktivität bereits früh in Web-Karten eingebaut wurde. Im Laufe der Zeit wurde ein reiches Spektrum an unterschiedlichen Funktionen entwickelt. Abhängig vom **Grad der Interaktivität** können die Karten auf dem Web in drei Klassen aufgeteilt werden. Cecconi et al. [Cecconi et al., 2000] klassieren sie ähnlich wie Kraak und Peterson (siehe S. 6). Neben den Karten ohne Manipulationsmöglichkeit (statische Karten), sind Karten mit limitierter Manipulationsmöglichkeiten und Karten mit erweiterter Manipulationsmöglichkeiten anzutreffen. Die Entwicklung bewegt sich in zunehmendem Mass in Richtung höherer Interaktivität.

Limitierte Manipulationsmöglichkeiten dienen primär einer benutzerfreundlichen Gestaltung. In Karten mit **erweiterten Manipulationsmöglichkeiten** übersteigen die Interaktionen die einfache Abfragefunktionalität. Der Benutzer hat einerseits die Gelegenheit, verschiedene Parameter und Variablen seiner Kartendarstellung zu spezifizieren und sich am Bildschirm anzeigen zu lassen [Cecconi et al., 2000]. Zum anderen verfügen solche Karten auch über GIS-Funktionalität, d. h. sie sind mit Fähigkeiten zur komplexen Datenabfrage und -analyse ausgestattet [Dickmann, 2000a]. Diese Art der Interaktivität verlangt von den Anwendenden aber Expertenwissen. Sie ist deshalb meistens nur für ein Fachpublikum geeignet.

Die vorliegende Arbeit befasst sich insbesondere mit Interaktivität für limitierte Manipulationsmöglichkeiten. Im Abschnitt 2.2.1 werden die grundlegenden, d. h. die oft eingesetzten Interaktionen dieser Klasse zusammengefasst.

2.2.1 Grundlegende Interaktionen

Viele der Karten mit limitierten Manipulationsmöglichkeiten weisen eine ähnliche Grundfunktionalität auf. Dabei handelt es sich in erster Linie um eine „ansichtsverändernde“ Form der Interaktion, um die Defizite des Mediums Bildschirm auszugleichen. Mit Hilfe dieser Funktionen können nicht oder schlecht sichtbare Elemente durch nutzerdefinierte Aktionen erschlossen werden. Interaktionen zur einfachen Steuerung von Karten sind in den letzten Jahren für viele Benutzer und Benutzerinnen zur Selbstverständlichkeit geworden. Aus Sicht der Kartenautoren und -autorinnen sind der graphischen Gestaltung der Benutzerführung kei-

ne Grenzen gesetzt. Die Interaktionen werden zwar unterschiedlich ausgelöst und gesteuert, das Resultat ist jedoch meistens das gleiche. Insbesondere neuere Techniken, die im Internet zum Einsatz kommen, tragen zu dieser flexiblen Anwendung von Steuerelementen bei.

Die folgende Aufstellung liefert einen detaillierten Überblick der am häufigsten anzutreffenden Grundfunktionen.

Zoom

Die Zoom-Interaktion wurde wahrscheinlich am frühesten implementiert. Sie wurde bereits im *Map Viewer* im Jahre 1993 den Nutzenden zur Verfügung gestellt.

Mit der Zoom-Interaktion können Kartenbetrachter und -betrachterinnen den Massstab verändern. Durch das Einzoomen, d. h. durch die Vergrösserung des Massstabs, werden kleine Elemente der Karte besser sichtbar. Im Gegenzug kann durch Auszoomen ein besserer Überblick über die dargestellte Szene gewonnen werden. Graphische Variationen der Implementierung sind folgende:

- Im Navigationsteil der Karte sind Symbole vorhanden, die das Ein- und Auszoomen ermöglichen [24].
- Durch das Aufziehen eines Rechteckes im Kartenbild wird der neue sichtbare Ausschnitt definiert [44].
- Mit Hilfe eines Schiebereglers kann die Zoomstufe eingestellt werden [44].
- In einem Auswahlmenü können vordefinierte Vergrösserungsfaktoren ausgewählt werden [31].
- Die gewünschte Massstabszahl kann per Tastatur eingegeben werden.

Der *Adobe SVG Viewer* bietet standardmässig die Möglichkeit zum Zoomen. Wenn mit der rechten Maustaste (Windows) resp. bei gedrückter ctrl-Taste (Macintosh) auf die SVG-Graphik geklickt wird, erscheint ein Auswahlmenü, in welchem die Zoom-Funktion ausgewählt werden kann. Der grosse Nachteil liegt allerdings darin, dass die ganze Graphik gezoomt wird. Karten bestehen jedoch aus dem Kartenbild und den Randangaben. In solchen Applikationen soll nur das Kartenbild, nicht aber der Rest der Graphik vergrössert bzw. verkleinert werden. In anspruchsvollen Anwendungen muss die Zoom-Funktion daher mit einer der oben beschriebenen Möglichkeiten implementiert.

Pan

Unter Pan bzw. Panning versteht man das Navigieren in der Karte durch Verschieben des Bildausschnittes. In einer gezoomten Karte sind nicht alle Bereiche sichtbar. Mit Hilfe einer Pan-Interaktion können sich die Benutzenden solche Bereiche anzeigen lassen. Normalerweise kann der Kartenausschnitt in die vier Quer- und die vier Diagonalrichtungen verschoben werden. In einigen Karten ist auch eine Möglichkeit zum Zentrieren des Kartenbildes anzutreffen. Die Implementierungsmöglichkeiten sind:

- Der Kartenausschnitt verschiebt sich bei gedrückter Maustaste zusammen mit dem Cursor [44].
- Im Navigationsteil der Karte können Pfeilsymbole angeklickt werden [24].
- Die Pfeilsymbole sind am Rand des Kartenausschnittes angebracht [26].
- Ein im Kartenbild angeklickter Punkt wird ins Zentrum der Karte verschoben [44].
- In einer Übersichtskarte (siehe unten) zeigt ein Rechteck den sichtbaren Kartenausschnitt an. Dieses kann mit der Maus verschoben werden, der Bildausschnitt verschiebt sich analog dazu [24].

Auch die Interaktion zum Verschieben des angezeigten Bereiches ist im SVG Viewer integriert. Sie verfügt jedoch über das gleiche Manko wie die Zoom-Interaktion.

Übersichtskarte

Neuere Web-Karten verfügen oftmals über eine kleine Übersichtskarte, in welcher immer die ganze Ausdehnung der Karte dargestellt wird. Ein transparentes oder farblich nicht gefülltes Rechteck zeigt den im Kartenausschnitt sichtbaren Bereich an. Dank dieser Interaktion behalten die Benutzenden stets die Übersicht, welcher Ausschnitt der Karte momentan angezeigt wird. Diese Funktion war erstmals in einer SVG-Karte [10] von Gaëtan Gaborit im Jahr 2001 anzutreffen.

Massstab

Mit jeder Änderung der Zoomstufe verändert sich der Massstab einer Karte. In interaktiven Karten passt sich die Beschriftung der Massstabsleiste bzw. die Massstabszahl in einem numerischen Massstab automatisch an [20].

Themenwahl

In thematischen Web-Karten kann in vielen Fällen aus verschiedenen Themen, welche sich auf dieselbe Basiskarte beziehen, gewählt werden. Der Vorteil dieser Methode liegt darin, dass die Benutzenden nicht für jedes Thema eine eigene Karte aufrufen müssen. Sie können sich die Karte auf den Client laden und in einer Auswahlliste bestimmen, welche Information sie betrachten möchten [31].

Bei einer grossen Anzahl von dargestellten Klassen können die Betrachtenden diese wegen der mangelhaften farblichen Abstufung oder dem mit Informationen überfüllten Kartenbild oft nicht mehr gut unterscheiden. Ein beliebtes Hilfsmittel zur deutlicheren Darstellung einer einzelnen oder einer bestimmten Auswahl von Klassen ist das Ein- und Ausblenden von diesen. Dies wird in der Regel durch das Anklicken des Legendeneintrages einer Klasse bzw. einer dafür bestimmten Markierung ermöglicht. Die Benutzenden erlangen damit die Kontrolle über die Zusammenstellung der angezeigten Daten in einer Karte [24].

Hervorhebungen

Zur besseren Unterscheidung von Kartenelementen werden von den Benutzenden gesteuerte, graphische Hervorhebungen (meist durch eine Änderung der Farbe) eingesetzt. Sie dienen in den meisten Fällen zur einfacheren Zuordnung eines Kartenelementes zu einer bestimmten Klasse. Diese Funktion wird normalerweise durch das Überfahren eines Kartenelementes mit der Maus ausgelöst. Im Internet wird eine grosse Vielfalt dieser Interaktionsmöglichkeit angetroffen:

- Beim Überfahren eines Kartenelementes ändert sich dessen Aussehen [14].
- Beim Überfahren eines Kartenelementes ändert sich das Aussehen aller zu dieser Klasse gehörenden Kartenelemente.
- Beim Überfahren eines Kartenelementes ändert sich das Aussehen des entsprechenden Legendeneintrages [32].
- Beim Überfahren eines Legendeneintrages ändert sich das Aussehen der zur Klasse gehörenden Kartenelemente [32].

Informationsanzeige

Viele Informationen können nicht in kartographische Symbole umgewandelt werden. Das Medium Internet bringt die Möglichkeit mit sich, Informationen auf Verlangen der Benutzenden aufzurufen. Diese Methode wird auch bei Web-Karten angewandt. Beim Überfahren oder Anklicken eines Kartenelementes werden Informationen zu diesem in einer der folgenden Varianten dargestellt:

- Eine Beschriftung wird neben dem überfahrenen Element angezeigt [14].
- Ein Text wird in einem bestimmten Bereich des Kartenlayouts eingeblendet [32].
- Informationen oder weiterführende Links werden in einem neuen Browserfenster aufgerufen [30].
- Multimediaelemente wie Bilder, Musik oder Filme werden angezeigt resp. abgespielt [14].

2.3 Erstellung von interaktiven SVG-Karten

Dieser Abschnitt beschäftigt sich mit der Herstellung von interaktiven SVG-Karten und den Problemen in diesem Arbeitsablauf. Diese Beschreibung ist bewusst sehr allgemein gehalten. Da jede Karte ihre ganz besonderen Merkmale und Eigenheiten aufweist, ist es an dieser Stelle unmöglich, eine Schritt für Schritt Anleitung für die Aufbereitung von räumlichen Daten für die Publikation im Internet zu geben.

Der erste Schritt beginnt bei der Identifizierung des Nutzungskontextes. Daraus resultiert, wie das Kartenlayout gestaltet sein soll und welche Interaktionen integriert werden sollen. Das erstellte Grobkonzept sollte möglichst umfassend sein, um Fehler im Herstellungsprozess

bereits von Anfang an zu vermeiden. In grossen Projekten lohnt es sich, kleine Prototypen und Beispielapplikationen zu erstellen. [Dahinden et al., 2003]

Anschliessend folgt die Erstellung einer statischen Karte unter Berücksichtigung kartographischer Gesichtspunkte und die Generierung der SVG-Datei. Für die Erzeugung von SVG-Graphiken stehen eine Reihe an geeigneten Werkzeugen zu Verfügung. Dahinden et al. [Dahinden et al., 2003] sprechen von folgenden Möglichkeiten:

Texteditoren

SVG ist textbasiert und kann grundsätzlich von jedem herkömmlichen Texteditor bearbeitet werden. Für eine übersichtliche Handhabung des Quellcodes wird idealerweise ein Editor mit Syntax-Highlighting sowie komfortablen Such- und Ersetzfunktionen verwendet. Dieser Ansatz eignet sich jedoch höchstens für die Erstellung kleiner Graphiken und zur Optimierung von bestehendem SVG-Code, da der Aufwand beim manuellen Editieren des SVG-Codes schnell stark anwachsen kann.

Autorensysteme

Mit Hilfe von Autorensystemen können SVG-Graphiken in einem WYSIWYG-Modus (und oft auch einem Quelltext-Modus) bearbeitet werden. Vorteilhaft gegenüber reinen Texteditoren sind das einfachere Editieren und das schnellere Auffinden von Fehlern. Die Vertreter dieser Werkzeuge sind oftmals noch in einer frühen Entwicklungsstufe, durch die Spezialisierung auf SVG produzieren sie dafür spezifikationskonformen Code. Erfolgsversprechende Projekte sind *WebDraw* von Corel, *XStudio* von Evolgrafix und *Sketsa* von Kiyut. Auch im Open-Source-Bereich sind diverse SVG-Editoren am entstehen, so zum Beispiel *Inkscape*, *Sodipodi* und *Glips Graffiti* von ITRIS.

Vektorzeichenprogramme

Kartographen und Kartographinnen verwenden seit langem Vektorzeichenprogramme in der Kartenproduktion. In neueren Programmversionen ist oftmals eine Exportfunktion für das SVG-Format integriert worden. Es ist somit möglich, eine Grosszahl von bestehenden Graphiken in SVG umzuwandeln. Die bekanntesten Vektorzeichenprogramme, die SVG-Dateien erzeugen können, sind Adobe *Illustrator* und *Corel Draw*. Die im kartographischen Bereich beliebte Software Macromedia *Freehand* verfügt über keine SVG-Exportfunktion. Der Export kann aber über den Umweg Adobe *Illustrator* vorgenommen werden [Kellenberger, 2005].

Konverter

Mit Konvertern können diverse Graphikformate in SVG umgewandelt werden. Es existieren u. a. Konverter für die Formate PostScript (.ps), Encapsulated PostScript (.eps), Portable Document Format (.pdf), Macromedia Flash (.swf) und Shapefile (.shp).

Programmierhilfen

Softwareentwickler und -entwicklerinnen können auf existierende XML-Parser oder Bibliotheken zurückgreifen, welche sie leicht in ihre Applikationen einbauen können, um SVG-Code zu lesen, zu verändern und zu speichern. Es gibt einige Bibliotheken für die Sprachen C/C++, Java und Perl. Diese Hilfen erlauben es auch, SVG-Dokumente direkt aus Datenbanken zu generieren.

Der durch die Programme generierte SVG-Code ist für kartographische Zwecke nicht immer ideal aufgebaut. Bei der Erstellung von SVG für den optimalen Einsatz in interaktiven Web-Karten muss deshalb auf einige Aspekte besonderen Wert gelegt werden. Dies ist hauptsächlich aus zwei Gründen notwendig. Zum einen kann durch eine optimale Struktur des SVG-Dokumentes die Dateigrösse verringert werden. Zum anderen muss der Quellcode auf die Programmierung der Interaktionen angepasst sein. Je nach gewählter Methode bei der Generierung ist die manuelle Nachbereitung des SVG-Codes mehr oder weniger umfangreich. Die wichtigsten Punkte, die es für eine optimale SVG-Struktur zu beachten gilt, sind die folgenden [Dahinden et al., 2003, Neumann und Winter, 2003b]:

- Gestaltung nach kartographischen Prinzipien
- Kennzeichnung der Kartenelemente mit `ids`
- Sinnvolle Unterteilung der Kartenelemente in Ebenen
- Verwendung von Stylesheets
- Verwendung von Symbolvorlagen
- Löschen von unnötigen Attributen
- Koordinatenangabe nur mit der benötigten Genauigkeit
- Möglichst wenige Schriften im SVG-Dokument einbetten

Steht die SVG-Graphik in ihrer endgültigen Version, können die JavaScript-Interaktionen programmiert und mit dem SVG-Code verknüpft werden. Dieser Schritt ist nicht zu unterschätzen und macht meistens den Hauptteil der Arbeit aus.

Der Skript-Code ist im Prinzip eine Textdatei, die eine Abfolge von Befehlen enthält und die mit Hilfe eines einfachen Texteditors erstellt werden kann. JavaScript wird auch zur Steuerung von Webseiten eingesetzt und mit Programmen wie zum Beispiel Macromedia *Dreamweaver* lassen sich solche Befehle auf Knopfdruck erzeugen. Für spezielle kartographische Funktionen lassen sie sich aber nicht verwenden.

Kartographische Interaktionen müssen letztlich immer noch von Hand programmiert werden. Viele der Funktionen müssen nur einmal geschrieben, bzw. sie können aus dem Internet kopiert und an das Projekt angepasst werden, Voraussetzung dafür sind allerdings umfassende JavaScript-Kenntnisse. Diese sind bei kartographischen Fachpersonen, graphisch orientierten Spezialisten und Spezialistinnen, nicht oder nicht in genügendem Umfang vorhanden. Im Abschnitt 3.3 wird gezeigt, dass eine simple Interaktion relativ einfach zu programmieren ist. Bei einem umfangreicheren Projekt umfasst der Code aber schnell mehrere 1000 Zeilen. Die Komplexität der Programmierung nimmt mit zunehmender Interaktivität stark zu.

Ein weiterer Nachteil von JavaScript besteht darin, dass die Skripte nicht von allen Browsern gleichermassen akzeptiert werden und der Code an jeden einzelnen Browser angepasst werden muss [Dickmann, 2002]. Dies bringt einen überdurchschnittlichen Aufwand beim Testen der Interaktionen mit sich. Erschwert wird die Programmierung von JavaScript auch durch das Fehlen von geeigneten Hilfsmitteln. Es können zwar Texteditoren mit Syntax-Highlighting eingesetzt werden, konfortable Debugger für JavaScript und SVG sind allerdings nicht vorhanden.

2.4 Idee für das Werkzeug

Wie wir in den bisherigen Ausführungen gesehen haben, stehen im kartographischen Sektor diverse Möglichkeiten zur Verfügung, um Produkte online zu veröffentlichen. Hilfsmittel für die einfache Herstellung von qualitativ hochstehenden Web-Karten sind bis dato jedoch nicht entwickelt worden. Im graphischen Bereich kann mit SVG nun seit einigen Jahren immerhin ein Format für ansprechende Graphiken eingesetzt werden. Trotz viel versprechender Anwendungen in der Web-Kartographie ist SVG jedoch kein „kartographisches Format“, sondern nach wie vor ein Graphikstandard. Deswegen müssen grundlegende Anforderungen des Webmapping's, wie die Navigation, teilweise mühsam implementiert werden. Die Erstellung einer derartigen, durch relativ komplexes Scripting zu erzeugende Funktionalität, ist eine der gegenwärtigen Schwierigkeit im kartographischen Bereich [Überschär et al., 2003]. Räber und Jenny [Räber und Jenny, 2003] fordern deshalb eine Möglichkeit, interaktive Karten zu erstellen, ohne den Quellcode editieren oder Routinen programmieren zu müssen.

Diese Forderung wird in der vorliegenden Arbeit aufgegriffen. Mit der Implementierung eines Werkzeuges zur automatischen Erzeugung von JavaScript-Interaktionen für SVG-Karten soll ein Werkzeug für Kartographen und Kartographinnen geschaffen werden, das diesen erlaubt, sich in Zukunft nicht mehr oder in geringerem Umfang mit der Programmierung von Interaktivität zu beschäftigen. Diese Arbeitserleichterung soll ein weiterer Baustein in der effizienten und effektiven Herstellung von interaktiven und graphisch anspruchsvollen Web-Karten sein.

3 Technische Grundlagen

3.1 SVG (Scalable Vector Graphics)

Im Abschnitt 2.1.3 wurde eine Einschätzung über den Stellenwert von SVG in der Web--Kartographie vorgenommen und die relevanten Eigenschaften von SVG für einen gelungenen Einsatz wurden erläutert. Es wurde gezeigt, dass sich SVG für die Darstellung von qualitativ hochstehende Graphiken im Internet eignet. In diesem Abschnitt werden wir uns mit den technischen Details von SVG auseinandersetzen.

Abbildung 3.1 zeigt eine einfache SVG-Graphik, bestehend aus einer grauen Grundfläche, einem roten Quadrat und einem blauen Kreis. Das Listing 3.1 zeigt den entsprechenden SVG-Code. Da SVG eine in XML formulierte Sprache ist, muss die Syntax entsprechend XML-konform sein [Sieber, 2002]. Gleich nach dem XML-Prolog, welcher Angaben zu Definitionen, Versionsinformationen etc. enthält, beginnt der eigentliche SVG-Code mit dem `<svg>`-Tag. Ein einzelner Tag besteht in den meisten Fällen aus dem Elementtyp und einer Reihe von Attributen mit Attributwerten.

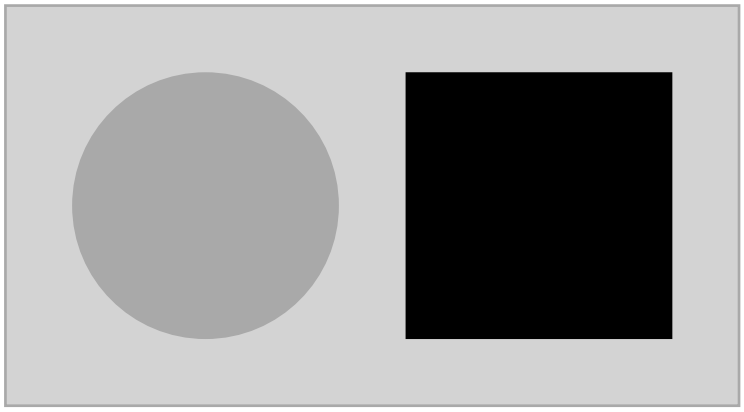

Abbildung 3.1: Einfache SVG-Graphik

Über das Attribut `id` kann einem Element eine eindeutige Identifikation zugewiesen werden. Über dieses Attribut kann ein Objekt mit JavaScript (Abschnitt 3.3) via DOM (Abschnitt 3.2) angesprochen und beeinflusst werden. [Joray, 2001]

Die Attribute `width` und `height` des `<svg>`-Tags definieren den Anzeigebereich am Bildschirm; die `viewBox` bestimmt das innere Koordinatensystem (Abbildung 3.2). Die Angaben erfolgen in beliebigen Masseinheiten. Die angegebenen Werte stammen im Normalfall vom Masssystem des Zeichenprogramms, von dem aus exportiert wurde. Innerhalb einer SVG-Graphik können weitere SVG-Graphiken mit einem eigenen Koordinatensystem eingebettet

```
<svg width="550px" height="300px" viewBox="0 0 550 300">
   <script xlink:href="scripts.js" language="Javascript"/>
   <rect id="border" width="550" height="300" fill="lightgray" stroke="darkgray"/>
   <g id="graphics" visibility="visible">
      <circle id="circle" r="100" cx="150" cy="150" fill="darkblue"
         onclick="changeColor(evt)"/>
      <rect id="redRect" x="300" y="50" width ="200" height="200" fill="darkred"/>
   </g>
</svg>
```

Listing 3.1: SVG-Code

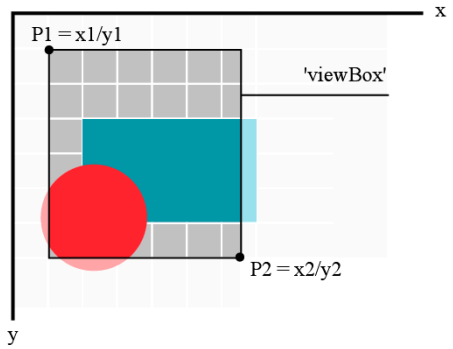

Abbildung 3.2: Der viewBox-Anzeigebereich
(Quelle: [Fibinger, 2001])

werden.

Mit dem `<g>`-Tag wird eine Gruppe von Elementen eingeleitet. In Listing 3.1 sind der Gruppe `graphics` ein Kreis (`circle`) und ein Rechteck (`redRect`) untergeordnet. Diese erben die Attribute des übergeordneten Elementes, in diesem Fall das Attribut `visibility`. Ist ein Attribut im über- und im untergeordneten Element angegeben, so hat der Attributwert des untergeordneten Elementes grundsätzlich höhere Priorität. Gruppen können beliebig viele Untergruppen enthalten.

Die JavaScript-Anweisungen können direkt in die SVG-Datei eingebettet werden oder wie im obigen Beispiel in einer separaten Datei gespeichert und mittels `<script>`-Tag referenziert werden. Die JavaScript-Datei wird im Attribut `xlink:href` bestimmt.

Ebenfalls als Attribut eines Elementes werden eventuelle Event-Handler angegeben. Diese reagieren auf ein bestimmtes Ereignis und dienen zur Auslösung von Skripts. Im obigen Beispiel wird bei einem Klick mit der Maus auf ein Element der Gruppe `graphics` der Event-Handler `onclick` benachrichtigt, das Skript `changeColor(evt)` zu starten. In der *SVG candidate recommendation* [41] ist im Anhang B die genaue Auflistung der unterstützten Event-Handler zu finden. In Tabelle 3.1 sind die im Zusammenhang mit SVG-Karten gebräuchlichsten Event-Handler aufgelistet.

Event-Handler	Wird ausgelöst, wenn
onload	das SVG-Dokument vollständig geladen und gerendert ist.
onresize	sich die Grösse der Ansicht verändert hat.
onclick	mit der Maus auf ein Element geklickt wurde.
onmousedown	die Maustaste über einem Element gedrückt wird.
onmouseup	die Maustaste über einem Element losgelassen wird.
onmouseover	sich der Mauszeiger über einem Element befindet.
onmouseout	der Mauszeiger ein Element verlässt.
onmousemove	sich der Mauszeiger bewegt.

Tabelle 3.1: Die gebräuchlichsten Event-Handler

3.2 DOM (Document Object Model)

Bereits im Abschnitt 3.1 wurde die Manipulation von SVG-Elementen per DOM angesprochen. Beim Document Object Model (DOM), seit 1. Oktober 1998 als W3C-Empfehlung vorliegend, handelt es sich dabei um eine plattform- und sprachneutrale Schnittstelle, die es Programmen und Skriptsprachen erlaubt, dynamisch auf den Inhalt von Dokumenten zuzugreifen und ihren Inhalt, ihre Struktur und ihren Darstellungsstil zu modifizieren [42]. SVG unterstützt das DOM Level 2 und eine spezifische Erweiterung für SVG, das sogenannte SVGDom. Die Integration von DOM in SVG ist einer der Grundsteine für das effiziente Arbeiten mit Elementen in einem SVG-Dokument [Neumann und Winter, 2003a]. Über diese Schnittstelle ist der Zugriff auf die Elemente mit Programmiersprachen wie Java, Javascript oder ECMAScript möglich.

Das Modell beschreibt die Zusammenhänge von Elementen, man spricht auch von Objekten, mittels einer Baumstruktur, wie sie in XML generell vorhanden ist (Abbildung 3.3) [Fibinger, 2001]. Für ein Element in der Baumstruktur wird grundsätzlich der englische Begriff *node* (Knoten) verwendet, in Bezug auf SVG ist auch *element* gebräuchlich. Ein übergeordnetes Element nennt man *parent*, bei einem untergeordneten Element spricht man von *child*.

Die einzelnen Objekte können zum einen über das Attribut `id` angesprochen werden. Zum

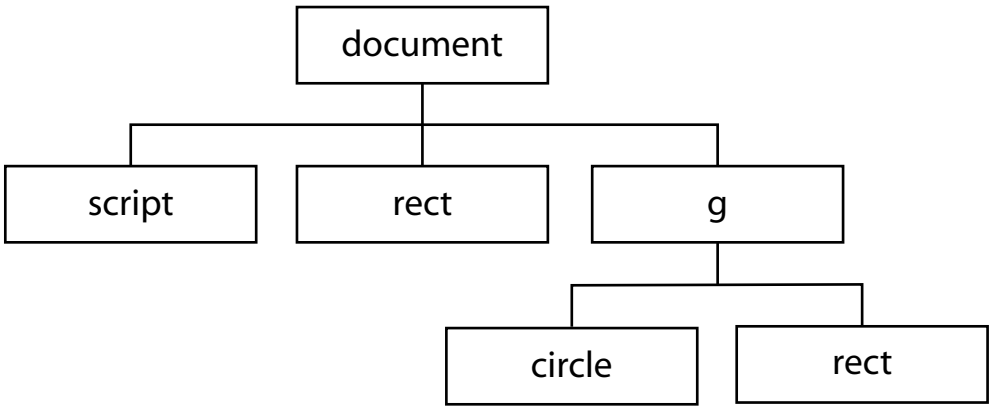

Abbildung 3.3: DOM-Stuktur der SVG-Graphik in Abbildung 3.1

Funktion	Beschreibung
getElementById()	spricht ein Element über das Attribut `id` an.
getElementsByTagName()	gibt eine Liste aller Elemente eines Typs zurück.
createElement()	erstellt ein beliebiges neues Element.
createTextNode()	erstellt ein neues Textelement.

Tabelle 3.2: Die wichtigsten *document*-Funktionen

Funktion	Beschreibung
getNodeType()	gibt den Typ eines Knoten zurück, z. B. Element oder Text.
getTagName()	gibt den Namen eines Knoten zurück, z. B. `circle` oder `rect`.
hasChilds()	ermittelt, ob ein Knoten untergeordnete Knoten besitzt.
getChildNodes()	gibt eine Liste der untergeordneten Knoten zurück.
getFirstChild()	gibt den ersten untergeordneten Knoten zurück.
getLastChild()	gibt den letzten untergeordneten Knoten zurück.
getParentNode()	gibt den übergeordneten Knoten zurück.
appendNode()	fügt einen untergeordneten Knoten an letzter Stelle an.
insertBefore()	fügt einen Knoten an einer bestimmten Position an.
removeChild()	löscht einen untergeordneten Knoten.
replaceChild()	ersetzt einen untergeordneten Knoten.
cloneNode()	dupliziert einen Knoten.
hasAttribute()	ermittelt, ob ein Knoten ein bestimmtes Attribut besitzt.
getAttribute()	gibt den Wert eines bestimmten Attributes zurück.
setAttribute()	ändert den Wert eines Attributes.
removeAttribute()	löscht ein bestimmtes Attribut.

Tabelle 3.3: Die wichtigsten *element*-Funktionen

anderen kann das DOM beliebig von oben nach unten und umgekehrt traversiert werden, um ein oder mehrere Elemente zu bearbeiten. Jeder Zugriff auf einzelne Knoten innerhalb eines SVG-Dokumentes erfordert meistens eine Referenz auf das Dokument selber. Dieses wird im DOM als *document* (oder *root*) bezeichnet und ist zuoberst in der DOM-Hierarchie zu finden.

Zur Erzeugung von Interaktivität in SVG-Graphiken und zur Manipulation der SVG-Elemente steht eine Vielzahl von Funktionen zur Verfügung. In der Tabelle 3.2 ist eine Beschreibung von ausgewählten Methoden des *document*-Elementes angegeben. Die wichtigsten Funktionen zur Manipulation eines *elements* sind in der Tabelle 3.3 aufgelistet. Wie die skript-gesteuerten Interaktionen auf das DOM zugreifen, wird im nächsten Abschnitt gezeigt. Eine komplette Liste der DOM-Schnittstelle für ECMAScript ist im Anhang E der DOM-Spezifikation [40] zu finden, die spezifischen SVGDom-Funktionen sind im Anhang E der SVG-Spezifikation [41] definiert. Beim Einsatz dieser Funktionen ist jedoch zu berücksichtigen, dass in keinem der existierenden SVG-Viewer die gesamte Spezifikation implementiert ist.

3.3 JavaScript

Die Interaktionen werden, wie bereits mehrmals erwähnt, mit einer Programmier- oder Skriptsprache realisiert. Unter den eingesetzten Sprachen ist JavaScript wahrscheinlich die populärste. Sie ist objektorientiert und wird wie alle Skriptsprachen eingesetzt, um die Einrichtung eines existierenden Systems zu verändern, anzupassen oder zu automatisieren [Berger, 2003]. Dabei läuft eine Skriptsprache immer in einer bestimmten Umgebung ab, bei Web-Karten ist dies der Browser, und verwendet die von dieser Umgebung bereitgestellten Funktionalitäten. Der Code wird zusammen mit der HTML- oder SVG-Datei übertragen und auf dem Client-Computer ausgeführt. JavaScript bietet eine breite Palette an Funktionen und Anweisungen zur Ausführung mathematischer Operationen, zur Manipulation von Zeichenketten, zur Steuerung von Browserfenstern und vielem mehr [Sieber, 2002]. Weitere Funktionen können selber programmiert werden. Skriptsprachen sind relativ einfach zu erlernen, um sie jedoch optimal einsetzen zu können, braucht es vertiefte Kenntnisse.

JavaScript ist ursprünglich eine Gemeinschaftsentwicklung der Firmen Netscape und Sun Microsystems und wurde 1995 mit der Herausgabe des Browsers *Netscape Navigator 2* zum ersten Mal veröffentlicht [Berger, 2003]. Als Antwort darauf entwickelte Microsoft die Skriptsprache JScript und integrierte sie in ihren Browser *Internet Explorer*. Die beiden Sprachen sind mehr oder weniger kompatibel, von einem einheitlichen Modell kann jedoch nicht die Rede sein. Dies hatte für Entwickler die unangenehme Folge, Codes auf beide Browser anpassen zu müssen. [Joray, 2001] Um dieser unbefriedigenden Situation ein Ende zu bereiten, unternahm die European Computer Manufacturer's Assocation eine Normierung vor. Es resultierte der Standard ECMA-262 [European Computer Manufacturer's Assocation, 1999] oder ECMAScript. JavaScript ist ab der Version 1.4 voll kompatibel zu ECMAScript [22]. Bei der Verwendung von JavaScript sollte man sich demnach an die Vorgaben von ECMAScript halten.

Zur Illustration der Syntax von JavaScript werfen wir noch einen kurzen Blick auf ein Code-Fragment. Das Beispiel in Listing 3.2 bezieht sich auf die Abbildung 3.1 und zeigt den JavaScript-Code für die Funktion `changeStyle(evt)`. Durch das Anklicken des Kreises ändert sich die Farbe des Quadrates auf Dunkelgrün. Das Resultat dieser Interaktion ist in Abbildung 3.4 zu sehen.

Der JavaScript-Code ist aus vier Anweisungen aufgebaut. In der ersten Zeile wird die Funktion mit dem Namen `changeColor(evt)` definiert. In den Klammern können dabei beliebig viele Argumente übergeben werden. In diesem Fall wird das auslösende `Event`-Objekt `evt` für die Abarbeitung der Anweisungen benötigt. Zwischen den geschwungenen Klammern stehen

```
1 function changeColor(evt) {
2    var svgdoc = evt.target.getOwnerDocument();
3    var ele = svgdoc.getElementById('redRect');
4    ele.setAttributeNS(null, 'fill', 'white');
5 }
```

Listing 3.2: Code der JavaScript-Funktion changeStyle(evt)

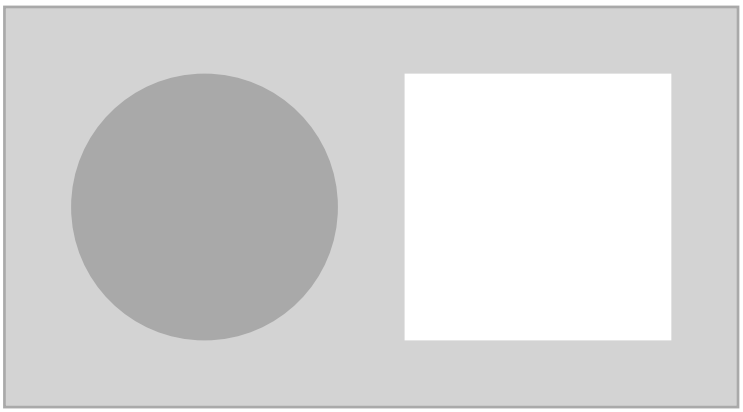

Abbildung 3.4: SVG-Graphik nach der Interaktion

die Anweisungen der Funktion. In der Zeile 2 wird das Objekt *document* ermittelt und in der Variable `svgdoc` gespeichert. Das *document* ist bedeutend, da es das oberste Objekt in der DOM-Hierarchie darstellt und als Ausgangspunkt für den Zugriff auf die anderen Elemente benötigt wird. In JavaScript dienen Punkte (.) für den Zugriff auf Attribute oder Funktionen eines Objektes. Von dieser Syntax wird in diesem Fall Gebrauch gemacht. Es wird das Attribut `target`, welches wiederum ein Objekt ist, des Objektes `evt` angesprochen und dessen Funktion `getOwnerDocument()` aufgerufen. Der Rückgabewert dieser Funktion ist das Objekt *document*. Es wird jedoch nicht das Objekt *document* selber, sondern das rote Rechteck benötigt. Dieses wird mit der Funktion `getElementByID('redRect')` referenziert und in der Variable `ele` abgelegt (Zeile 3). In den Klammern dieser Funktion kann die `id` des gewünschten Elementes angegeben werden. In der letzten Anweisung wird die eigentliche Farbänderung durchgeführt. In der Funktion `setAttributeNS(null, 'fill', 'darkgreen')` stehen drei Argumente. Das erste bezieht sich auf den Namespace und wird in diesem Beispiel nicht benötigt. Als zweites Argument wird das zu ändernde Attribut (`fill`) spezifiziert und im dritten der neue Wert (`darkgreen`), den das Attribut erhalten soll, angegeben.

Ein breiterer Hintergrund kann an dieser Stelle nicht vermittelt werden; dies würde den Rahmen dieser Arbeit sprengen. Zur Vertiefung in die Thematik steht aber ein grosses Spektrum an Büchern und Webseiten zur Verfügung. Einen guten Einstieg vermitteln [Berger, 2003] sowie die Webseiten [45] und [17] . Es sei jedoch noch einmal bemerkt, dass das Zusammenspiel von SVG, dem DOM und JavaScript ein reichhaltiges Set an Möglichkeiten für die Erzeugung von Interaktivität in graphisch ansprechenden Web-Karten bietet.

3.4 Java

Ein Ziel dieser Arbeit besteht in der Implementierung eines Werkzeuges zur automatischen Erzeugung von JavaScript-Interaktionen für SVG-Karten. Das Zielpublikum sind in erster Linie Kartographen und Kartographinnen. Diese arbeiten oftmals in heterogenen Computerumgebungen, d. h. sie arbeiten auf verschiedenen Betriebssystemen wie Microsoft Windows, Mac OS oder Linux. Dies ist der Hauptgrund, warum Java als Programmiersprache für das

Werkzeug verwendet wird. Durch die Portabilität von Java ist der Einsatz auf verschiedenen Plattformen gewährleistet. Die Benutzenden sollen nicht an ein bestimmtes Betriebssystem gebunden sein.

Dies ist aber nicht das einzige Argument für den Einsatz von Java. Sun Microsystems, der Entwickler von Java, fasst die charakteristischen Eigenschaften dieser Sprache zusammen [34]:

> „[Java ist] eine einfache, objektorientierte, verteilte, interpretierte, robuste, sichere, architekturunabhängige, portable, hoch leistungsfähige, multithread-fähige und dynamische Sprache."
> (Übersetzung des englischsprachigen Originals)

Diese Eigenschaften haben dazu beigetragen, dass Java eine hohe Akzeptanz bei Entwicklern geniesst und in unzähligen Projekten erfolgreich eingesetzt wurde. Für die Verwendung in dieser Arbeit sind sicher die Punkte Einfachheit und Robustheit von entscheidender Bedeutung. Unter Einfachheit ist in diesem Zusammenhang die geringe Anzahl von Sprachkonstrukten zu verstehen, was es Programmierern ermöglicht, die Sprache schnell zu lernen [Flanagan, 1996]. Dies kommt einer produktiven Entwicklung des Werkzeuges zu gute. Die Robustheit ermöglicht eine einfachere Programmierung zuverlässiger Software, als dies mit anderen Sprachen möglich wäre [Flanagan, 1996].

Hilfreich für diese Arbeit sind auch geeignete Bibliotheken zur Manipulation von SVG- und JavaScript-Dokumenten. Grundsätzlich ist dies nicht unbedingt notwendig, da sowohl SVG wie auch JavaScript reine Textdateien sind. Diese können standardmässig mit Java geöffnet, bearbeitet und gespeichert werden. Die Anforderungen an das Werkzeug sind allerdings umfassender. So soll u. a. die graphische Ansicht des SVG-Dokumentes, die einfache Auswahl von Elementen der SVG-Datei mit der Maus und die Vorschau der interaktiven SVG-Karte möglich sein. Eine solche Funktionalität von Grund auf zu programmieren ist mit einem enormen Aufwand verbunden und im Rahmen dieser Arbeit nicht möglich. Dies ist auch gar nicht notwendig, da die gewünschte Funktionalität durch den Einsatz der Java-Bibliothek Batik zu bewerkstelligen ist. Zum Zeitpunkt der Niederschrift dieser Arbeit ist Batik die umfangreichste Java-Bibliothek zur Handhabung von SVG-Dateien. Andere Bibliotheken sind meistens auf die Konversion in das SVG-Format, z. B. der *AnotherSimpleShapefile Converter* [37], oder auf die Darstellung von SVG-Graphiken, z. B. *SVG Salamander* [15], spezialisiert.

3.4.1 Batik

Das Batik-Projekt [9] ist ein open-source-Prokjekt der Apache Software Foundation. Das Ziel ist die Bereitstellung eines Sets von Kernmodulen, die einzeln oder zusammen zur Unterstützung von SVG-Anwendungen eingesetzt werden können [16]. Die Version 1.0 wurde am 21. Mai 2001 veröffentlicht und wurde kontinuierlich weiterentwickelt.

Batik selber ist ein Java-Toolkit für Programme und Applets zur Darstellung, Generierung oder Manipulation von SVG-Graphiken [9]. Das Toolkit besteht aus mehreren wiederverwendbaren Komponenten die für die Integration in server- oder clientseitigen Applikationen entworfen wurden. So sind z. B. eine Komponente zur Darstellung von SVG-Dokumenten oder

ein Modul zur Konvertierung von SVG-Dokumenten in Rasterformate wie JPEG oder PNG vorhanden [11]. Bei der Implementierung wurde darauf geachtet, die SVG- und die DOM Level 2-Spezifikation so präzise wie möglich abzubilden.

Die Architektur von Batik besteht im Wesentlichen aus drei Arten von Modulen, den sogenannten Application Modules, den Core Modules und den Low Level Moduls (Abb 3.5).

Application Modules

Die Application Modules sind Anwendungen des eigentlichen Toolkits und demonstrieren die Funktionalität von Batik. Zusammen mit der Distribution werden vier Beispielprogramme, die das Potential von Batik aufzeigen, mitgeliefert:

- Squiggle SVG Browser

  Der Squiggle SVG Browser kann zur Betrachtung von SVG-Dokumenten verwendet werden. Er verfügt über Möglichkeiten zum Zoomen, Pannen und Rotieren der Graphik oder zur Auswahl von Textelementen. Dank der spezifikationskonformen SVG-Implementierung und dem Anzeigen von Fehlermeldungen kann dieser Browser auch zum Debugging von SVG-Dokumenten eingesetzt werden.

- SVG Rasterizer

  Der SVG Rasterizer ist ein Programm zur Konvertierung von SVG-Dateien in ein Rasterformat. Das Werkzeug kann ein oder mehrere Ausgangsdokumente umwandeln. Die unterstützten Formate sind JPEG, PNG und TIFF. Das Design erlaubt jedoch eine einfache Erweiterung um weitere Rasterformate.

- SVG Pretty Printer

  Der SVG Pretty Printer erlaubt es SVG-Code zu formatieren. So können Tabulatoren oder andere kosmetische Parameter in Ordnung gebracht werden. Er kann auch zur Modifikation des `DOCTYPE` verwendet werden.

- SVG Font Converter

  Der SVG Font Converter dient zur Konvertierung von True-Type-Schriften in SVG-Schriften und bettet diese in ein SVG-Dokument ein. Dies ermöglicht die vollständige Eigenständigkeit eines SVG-Dokumentes. Es wird dadurch auf allen Systemen exakt gleich dargestellt.

Core Modules

Die Core Modules sind das Herzstück der Batik-Architektur. Sie können einzeln oder zusammen für verschiedene Zwecke eingesetzt werden.

- SVG Generator

  Dieses Modul beinhaltet SVGGraphics2D, eine abstrakte Klasse zum Rendering von Graphiken. Es ermöglicht allen Java-basierten Applikationen und Applets auf einfache Art und Weise die Konvertierung ihrer Graphiken in das SVG-Format unter Verwendung des Java2D APIs.

- SVG DOM

  Dies ist die Implementierung des SVG DOM API der SVG-Spezifikation. Es ermöglicht die Manipulation eines SVG-Dokuments in Java.

- JSVGCanvas

  Die JSVGCanvas ist eine Benutzerschnittstellenkomponente. Sie erlaubt es den Benutzenden mit der Graphik zu interagieren (Zoom, Pan, Rotation, Textauswahl,...).

- Bridge

  Dieses Modul wird nicht oft direkt benutzt. Es ist verantwortlich für die Erzeugung und Wartung eines Java-Objektes, das dem SVG-Element entspricht. Die Bridge konvertiert ein SVG-Dokument in eine Batik-interne Repräsentation von Graphiken, das sogenannte Graphic Vector Toolkit (GVT).

- Transcoder

  Dieses Modul stellt ein generisches API für die Umwandlung einer Eingabe in eine Ausgabe dar.

Low Level Modules

Die Low Level Modules sind nicht für den direkten Gebrauch durch Entwickler, die das Batik Toolkit einsetzen, gedacht. Sie dienen als Grundlage und zur Unterstützung der Core Modules.

- Graphic Vector Tookit (GVT)

  Das Graphic Vector Toolkit repräsentiert eine Ansicht auf die Baumstruktur des DOM, welche sich besser für das Rendering und die Handhabung von Events eignet. Es beschreibt die Baumstruktur des DOM als ein Baum von Java-Objekten.

- Renderer

  Dieses Modul trägt die Verantwortung für das Rendering des GVT-Baumes und allen dazugehörenden Vorgängen.

- SVG Parser

  In diesem Modul ist eine Vielfalt von 'Micro Parsern' zusammengefasst. Diese sind Parser für komplexe SVG-Attribute wie *transform* oder *color*. Höhere Module beruhen zum Teil auf dem Parser-Modul.

Bei der Implementierung des Werkzeuges zur automatischen Erzeugung von JavaScript-Interaktionen für SVG-Karten ist auf zwei Module zurückgegriffen worden. Zur graphischen Anzeige der SVG-Datei kommt eine JSVGCanvas zum Einsatz. Für die Manipulation des SVG-Dokumentes und die Vorschau auf die interaktive Karte wird das SVG DOM eingesetzt. Auf ein Beispiel für die Verwendung von Batik wird hier verzichtet, einführende Code-Fragmente sind auf der Webseite von Batik [9] zu finden. Im Abschnitt 4.3 wird jedoch die Integration der oben genannten Module erläutert.

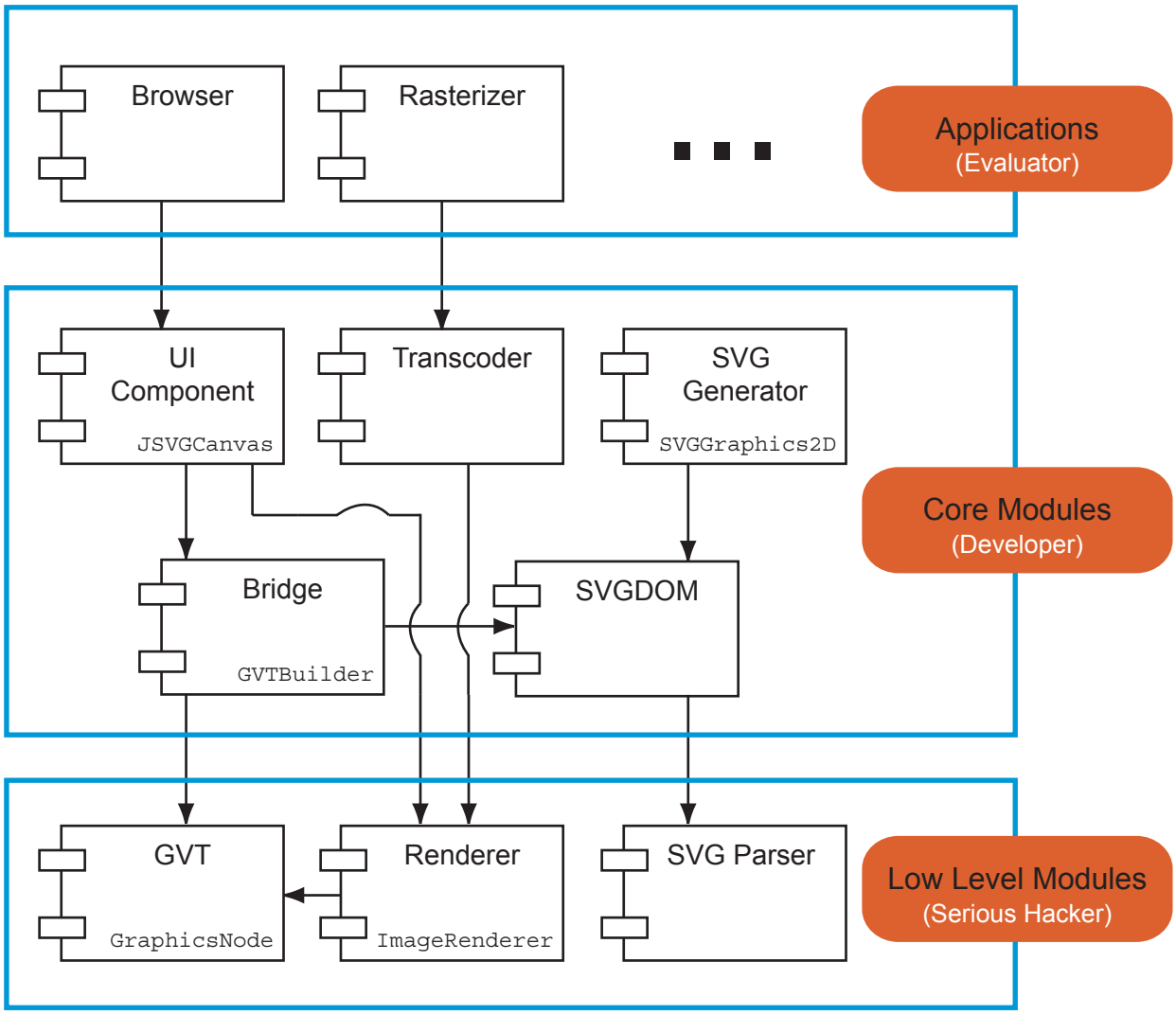

Abbildung 3.5: Die drei Modultypen von Batik
Quelle: [9]

4 Realisierung

Wie bereits in der Zielsetzung erwähnt, wurde im Rahmen dieser Arbeit ein Werkzeug für die automatische Erzeugung von JavaScript-Interaktionen für SVG-Karten implementiert. Für das erstellte Computerprogramm ist der Name ISIMAP gewählt worden, der auch im weiteren Verlauf dieses Textes verwendet wird. Die Funktionalität von ISIMAP ist auf Kartographen ausgerichtet, welche über keine oder nur geringe Erfahrungen mit der Programmierung von interaktiven SVG-Karten verfügen. Diese können den grössten Nutzen aus ISIMAP ziehen und werden deshalb als zukünftige Benutzer identifiziert.

Um sich auf die wesentliche Funktionalität konzentrieren zu können und um den zeitlichen Umfang der Implementierung von ISIMAP in Grenzen zu halten, wurden zuerst klar definierte Anforderungen festgelegt. Zur Ermittlung von klaren Anforderungen wurden Gespräche mit Personen der Zielgruppe durchgeführt. Dies ist die übliche Vorgehensweise, um die Wünsche und Vorstellungen an eine Software in Erfahrung zu bringen [Zuser et al., 2004]. Aus den Angaben dieser Befragung ist eine sogenannte Anforderungsspezifikation abgeleitet worden. Dabei wurden die grundlegenden Ziele der Arbeit (Abschnitt 1.2) miteinbezogen. Die resultierende Anforderungsspezifikation dieser Befragungen ist im nächsten Abschnitt zu finden. Aus den Anforderungen wurde das Design von ISIMAP abgeleitet und der Architekturentwurf erstellt. Dieser Schritt umfasst die Konzeption der Komponenten des Programms, ihre Schnittstellen und ihr Zusammenspiel (Abschnitt 4.2). Ausgehend von dieser Basis wurde ISIMAP implementiert und getestet. Auf Besonderheiten und Schwierigkeiten dieses Vorganges wird im Abschnitt 4.3 eingegangen. Die Vorgehensweise dieser Arbeitsschritte lehnen sich an den vorgeschlagenen Ablauf zur Herstellung von Software nach Glinz [Glinz, 2003] an.

4.1 Anforderungen

Eine grundlegende Anforderung an moderne Computerprogramme ist Benutzerfreundlichkeit (engl. *usability*). Dieses Thema ist in den letzen Jahren immer mehr in den Fokus der Diskussion rund um die Entwicklung von Software gerückt. Doch was versteht man genau unter dem Begriff Benutzerfreundlichkeit? Nach der Norm DIN EN ISO 9241 ist eine Software benutzerfreundlich, wenn die drei Kriterien Effizienz, Effektivität und Zufriedenheit für einen bestimmten Nutzungskontext erfüllt sind. [Vogt und Heinsen, 2003] Effektivität meint, dass sich mit einer Software die durch den Nutzungskontext gegebenen Aufgaben überhaupt ausführen lassen und dass die Arbeitsergebnisse die notwendige Mindestqualität aufweisen. Effizient bedeutet, dass die Aufgaben des Nutzungskontextes mit angemessenem bzw. möglichst geringem Aufwand erledigt werden können. Der dritte zentrale Anspruch ist die Zufriedenstellung der Benutzenden. Diese müssen subjektiv den Eindruck haben, mit der Software effektiv und effizient arbeiten zu können. [Baggen, 2003]

Die aktive Mitarbeit echter Benutzer und Benutzerinnen ist also das Herzstück bei der Entwicklung von benutzerfreundlichen Produkten [Vogt, 2003]. Durch ihre Mitarbeit bei der Konzeption einer Software wird die Chance auf ein breit akzeptiertes und einsetzbares Produkt stark gesteigert. Aus diesem Grund wurden sieben Personen mit guten kartographischen Kenntnissen nach ihren Bedürfnissen und Anforderungen an ein Computerprogramm zur automatischen Erzeugung von Interaktionen befragt. Zum einen konnte so ein möglicher Aufbau und eine sinnvolle Funktionalität von ISIMAP ermittelt werden. Da mit ISIMAP Neuland im Bereich der Web-Kartographie betreten wird, lieferten diese Angaben hilfreiche Tips für das Design des Programms. Zum anderen wurden aber auch Informationen zu wichtigen und oft eingesetzten Interaktionen gesammelt. Die grundlegenden Interaktionen der Web-Kartographie sollten ja mit ISIMAP erstellt werden können.

Als Methode für die Befragung des Zielpublikums wurden mündliche, halbstandardisierte Interviews eingesetzt. Bei dieser Technik gibt ein Gesprächsleitfaden das Grundgerüst für das Interview vor. Dank diesem wird auf alle relevanten Aspekte des Themas eingegangen. Die Flexibilität dieser Methodik lässt aber auch genügend Freiraum, um weiterführende Interessen und das Umfeld der potentiellen Benutzenden zu besprechen. Die Erwartungen an das Programm können von den Interviewteilnehmern und -teilnehmerinnen in ihren eigenen Worten geäussert werden. So ist die Chance zur Ermittlung einer möglichst umfassenden und vollständigen Anforderungsspezifikation gegeben. Auf der anderen Seite kann das gewählte Verfahren auch zu Schwierigkeiten bei der exakten Ausformulierung der Spezifikation führen. Bei einer unüberlegten Auswahl der Gesprächspartner oder der falschen Wahl der Methodik besteht die Gefahr einer mangelhaften Datengrundlage. Insbesondere durch mangelndes Interesse oder einen limitierten Ideenreichtum kann dieser Fall eintreten. Somit müsste die Anforderungsspezifikation auf eigenen Ideen basieren.

Für ein erfolgreiches Projekt müssen die einzelnen Interviews analysiert und in einzelne Anforderungen umgewandelt werden. Diese sind anschliessend aufgrund ihrer Bedeutung mit Prioritäten zu versehen. Bei der Definition des Designs der Software können die einzelnen Anforderungen schrittweise integriert werden. Unabhängig von der Methode für die Einteilung der Anforderungen müssen klar formulierte Punkte vorhanden sein. Eine Möglichkeit zur Klassifizierung ist die Verwendung eines Nummernsystems. Dies resultiert jedoch oft in einem fehlerhaften Resultat, da die einzelnen Anforderungen gerne mit einer zu hohen Priorität versehen werden. Geeigneter ist der Einsatz eines Verfahrens mit einer bestimmten Anzahl von Begriffen, die eine Bedeutung haben. In dieser Arbeit wurde deshalb die sogenannte MoSCoW-Methode eingesetzt. Das Akronym MoSCoW steht dabei für die vier englischen Wörter *must*, *should*, *could* und *would* (oder *won't*). Mit *must* bezeichnete Anforderungen sind unbedingt zu erfüllen; ohne deren Implementierung ist das Projekt zum Scheitern verurteilt. *Should*-Anforderungen sind wichtig, um ein qualitativ hochstehendes Produkt zu erhalten und geniessen ebenfalls einen grossen Stellenwert. Nice-To-Have-Funktionen werden mit *could* bezeichnet. Sie sind wünschenswert, können aber bei Bedarf vernachlässigt werden. Die *would*-Gruppe beinhaltet Anforderungen, die nicht berücksichtigt werden müssen, aber in einer zukünftigen Version des Programms wieder ins Auge gefasst werden können. [6]

Die in die Prioritätenklassen eingeteilten Anforderungen an ISIMAP sind in Tabelle 4.1 aufgelistet. Des Weiteren sind auch die erwünschten Interaktionen mit dem MoSCoW-System klassifiziert worden. Diese Aufstellung ist in den Tabellen 4.2 und 4.3 zu finden.

Priorität *must*	
1	Das Werkzeug muss SVG-Karten, welche mit Adobe *Illustrator* exportiert wurden, laden können.
2	Die zu ladende SVG-Datei muss vom Benutzer gewählt werden können.
3	Die bearbeitete SVG-Datei muss gespeichert und mit Adobe *Illustrator* geöffnet werden können.
4	Die geladenen Karten müssen graphisch dargestellt werden.
5	Einzelne oder mehrere Kartenelemente müssen per Mausklick ausgewählt werden können.
6	Dem/n ausgewählten Kartenelement/en muss/müssen eine oder mehrere JavaScript-Interaktion/en zugeordnet werden können.
7	Die Parameter der Interaktion müssen vom Benutzer eingestellt werden können.
8	Der JavaScript-Code muss in einer separaten Datei gespeichert werden.
9	Das Programm muss intuitiv und benutzerfreundlich sein.
10	Es muss ein visuelles Feedback über den Programmstatus angezeigt werden.
11	Der JavaScript-Code muss bei der Ansicht der SVG-Karte in einem Browser schnell geladen sein.
Priorität *should*	
1	Die Benutzerschnittstelle soll sich an gängige Programme anlehnen.
2	Das Programm soll möglichst viele JavaScript-Interaktionen anbieten.
3	Die Interaktionen sollen browserunabhängig sein.
4	Der JavaScript-Code soll kurze Kommentare enthalten.
5	Das Programm soll über eine undo-Funktion verfügen.
6	Die ausgeführten Aktionen und Fehler sollen in eine log-Datei geschrieben werden.
Priorität *could*	
1	Das Programm könnte einen integrierten Codeeditor beinhalten.
2	Das Programm könnte über eine Vorschaufunktion verfügen.
3	Die SVG-Karten könnten mit den Adobe *Illustrator* Versionen 8, 9, 10 und CS hergestellt worden sein.
4	Die JavaScript-Funktionen könnten gut verständlich sein.
Priorität *would*	
1	Das Werkzeug wäre als Adobe Illustrator Plugin verfügbar.
2	Die Interaktionen würden per Drag&Drop den Kartenelementen zugeordnet werden.
3	Das Programm würde über eine editierbare History-Funktion verfügen.
4	Das Programm könnte *namespaces* der SVG-Karte berücksichtigen.

Tabelle 4.1: Die Anforderungen an ISIMAP

In den Interviews wurden mehrere Punkte als besonders relevant betont. Grossen Wert legen die Befragten auf Benutzerfreundlichkeit und ein intuitives Verstehen der Funktionalität. Es muss auf den ersten Blick klar sein, was gemacht werden kann und wie es gemacht werden kann. Die einzelnen Befehle sollen selbsterklärend und einprägsam sein. Für die Erstellung von statischen SVG-Karten benützen die meisten Interviewpartner das Vektorzeichenprogramm *Illustrator* von Adobe. Bei der Implementierung soll deshalb darauf geachtet werden, dass die SVG-Datei mit Adobe *Illustrator* erstellt sein kann, und die mit Interaktivität versehene Karte problemlos in diesem Programm weiterverarbeitet werden kann. Sehr wichtig ist auch die graphische Darstellung des SVG-Dokumentes in ISIMAP, damit die einzelnen Elemente einfach per Mausklick auswählbar sind. Ausserdem nehmen die Nützlichkeit und die Anwendungsmöglichkeiten von ISIMAP mit der Anzahl und der Bandbreite der erstellbaren Interaktionen zu.

Priorität *must*	
1	Durch Anklicken eines Symbols werden die Kartenelemente vergrössert (zoom).
2	Durch Anklicken eines Symbols werden die Kartenelemente verkleinert (zoom).
3	Durch Anklicken eines Symbols werden die Kartenelemente in der ursprünglichen Grösse angezeigt.
4	Der graphische Massstab muss sich automatisch der Zoomstufe anpassen.
5	Durch Anklicken eines Symbols wird der Bildausschnitt wahlweise in eine der acht Grundrichtungen verschoben (pan).
6	Durch Anklicken eines Punktes in der Karte soll dieser in das Zentrum des Ausschnittes verschoben werden (pan).
7	Beim Anklicken eines Ebenennamens (oder eines dazugehörigen Symbols) wird die Ebene aus- bzw. eingeblendet.
8	Beim Überfahren eines Kartenelementes mit der Maus wird das Kartenelement graphisch hervorgehoben.
9	Beim Überfahren eines Kartenelementes mit der Maus werden zum Kartenelement gehörende Informationen in einem bestimmten Bereich ausserhalb der Karte angezeigt.
10	Beim Überfahren eines Legendenelementes mit der Maus werden die zum Legendenelement gehörenden Kartenelemente graphisch hervorgehoben.
Priorität *should*	
1	Durch das Aufziehen eines Rechteckes in der Karte soll der so gewählte Ausschnitt auf die Grösse des Kartenbildes vergrössert werden (zoom).
2	Beim Überfahren eines Kartenelementes mit der Maus werden zum Kartenelement gehörende Informationen neben dem Kartenelement angezeigt (Label / Tooltip).
3	Durch das Verschieben eines Rechteckes in einer Übersichtskarte wird der Kartenausschnitt analog dazu verschoben.
Priorität *could*	
1	Durch Eingabe eines Prozentwertes wird die Karte auf diesen Wert gezoomt.
2	(De-) Aktivierung der *SVGViewer*-Funktionen.
3	Durch Anklicken eines bestimmten Bereiches am Kartenrand wird das Kartenbild verschoben.
4	Beim Überfahren eines Kartenelementes wird der dazugehörige Legendeneintrag hervorgehoben.
5	Die Koordinaten der Position des Mauszeigers werden in einem bestimmten Bereich angezeigt.
6	Beim Anklicken eines Kartenelementes wird eine Webseite in einem neuen Browserfenster geöffnet.
7	Für thematische Daten kann ein Histogramm angezeigt werden.

Tabelle 4.2: Gewünschte mit ISIMAP erstellbare Interaktionen (Teil 1)

Priorität *would*	
1	Durch Anklicken eines Punktes in der Karte würde der Kartenausschnitt vergrössert werden.
2	Beim Vergrössern und Verkleinern des Kartenbildes würden Symbolgrössen nicht proportional verändert werden.
3	Beim Überfahren eines Kartenelementes mit der Maus würden zum Kartenelement gehörende Informationen in der Statusleiste des Browsers angezeigt werden.
4	Beim Anklicken eines Symbols würden Informationen in einem PopUp-Dialog angezeigt werden.
5	Teilebenen könnten zusammengefasst werden.
6	Es könnte eine Selektionsliste, deren Einträge abhängig von der Auswahl in einer anderen Selektionsliste sind, hergestellt werden.
7	Bei einer thematischen Karte könnte der Kartenbenutzer die Anzahl Klassen und deren Klassengrenzen selber einstellen.
8	Der Kartenbenutzer könnte die Farben der Karte selber einstellen.
9	Pfade könnten animiert werden.
10	Symbole oder Bilder würden sich entlang eines Pfades verschieben.

Tabelle 4.3: Gewünschte mit ISIMAP erstellbare Interaktionen (Teil 2)

4.2 Design

4.2.1 Benutzeroberfläche

Benutzende sehen Programme als Hilfsmittel, die ihnen bei der täglichen Arbeit nützlich sein sollen. Gute Software zeichnet sich nicht nur durch möglichst fehlerfreie Funktionen aus, sondern auch dadurch, dass die Benutzeroberfläche ergonomisch gestaltet ist. [Müller-Prove, 2003] Die Benutzeroberfläche ist die eigentliche Schnittstelle zwischen einer Software und deren Benutzenden. Im Designprozess ist es deshalb von enormer Wichtigkeit, der Gestaltung der Benutzeroberfläche genügend Gewicht zu geben. Der Funktionenkatalog legt die internen Anforderungen an das Design der Benutzeroberfläche fest. Demgegenüber setzen die Programmierumgebung und die damit zur Verfügung stehenden Klassenbibliotheken die externen Grenzen [Shenton, 2000].

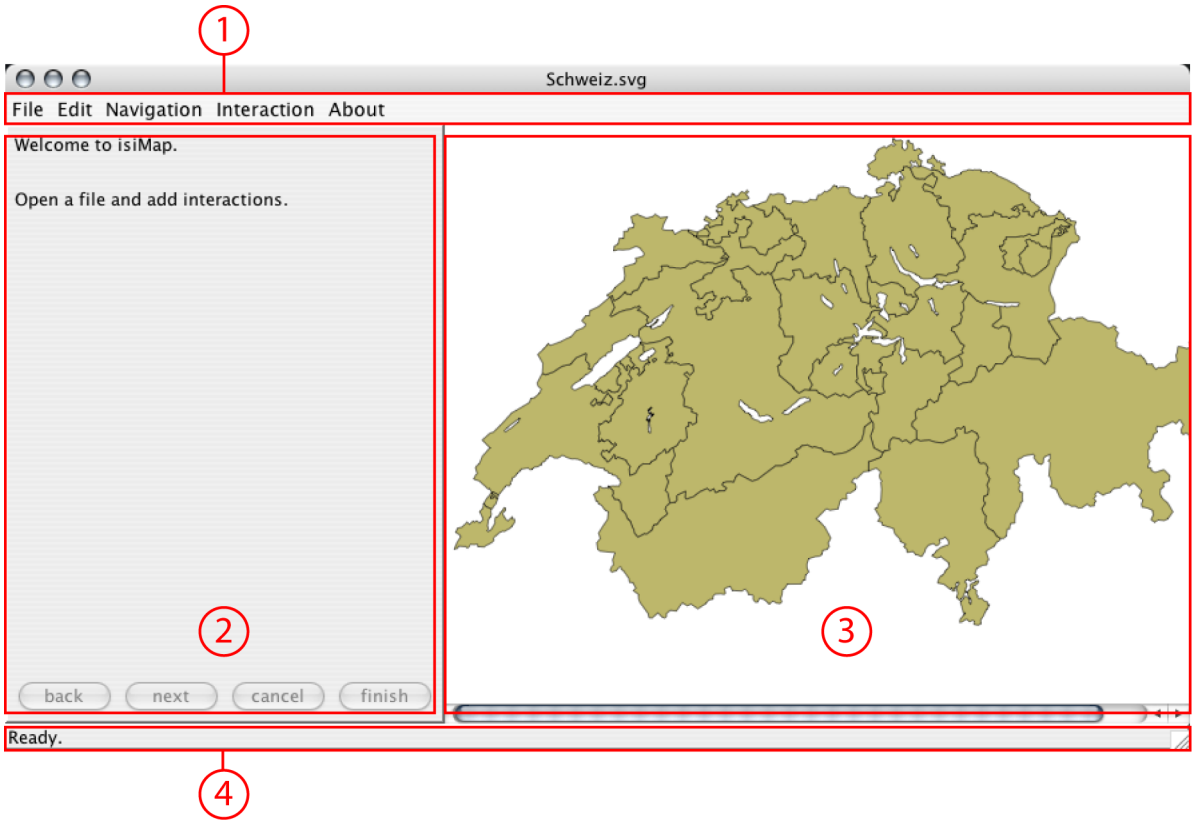

Abbildung 4.1: Layout von ISIMAP

Bei der Gestaltung der Benutzeroberfläche von ISIMAP wurde ein möglichst schlichtes und einfach zu bedienendes Design angestrebt. Das Layout (Abbildung 4.1) besteht aus vier Bereichen. In der Menü-Leiste (1) am oberen Rand der graphischen Oberfläche sind alle verfügbaren Befehle möglichst übersichtlich und in funktionale Klassen eingeteilt angeordnet. Der linke Bereich (2) dient zur Einstellung von Parametern der Interaktionen. Die grösste Fläche im Layout nimmt die Anzeige des SVG-Dokumentes (3) ein. In der Statusbar (4) wer-

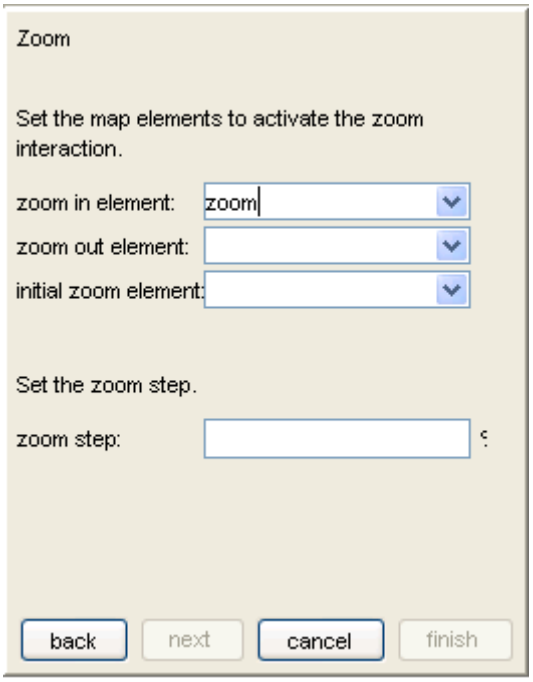

Abbildung 4.2: Der Wizard zur Eingabe vom Parametern

den Informationen über den Programmzustand für den Benutzer angezeigt. Auf Fehler wird in PopUp-Dialogen aufmerksam gemacht.

Insgesamt sind in der Menüleiste 17 Befehle, eingeteilt in fünf Menüs, zu finden. Bei der Benennung der Befehle wurde auf die Verwendung von unmittelbar verständlichen Begriffen grossen Wert gelegt. Die Befehle können durch das Anwählen mit der Maus oder durch das Drücken einer Tastenkombination ausgelöst werden. Eine kurze Beschreibung der einzelnen Funktionen von ISIMAP ist in Tabelle 4.4 vorhanden. Die meisten Befehle sind aus anderen Programmen bestens bekannt und bedürfen keiner weiteren Erklärung. Einzigartig ist dagegen das Menü *Interaction*. In diesem sind die einzelnen erstellbaren Interaktionen für die SVG-Karte vorhanden.

Vor der Erzeugung einer Interaktion müssen wie erwähnt einige Angaben von den Benutzenden eingegeben werden. Dies geschieht mit Hilfe eines sogenannten Wizards (Abbildung 4.2). Dies sind mehrseitige Dialoge, in welchem die Benutzenden Einstellungen vornehmen können. In ISIMAP können so die Parameter der Interaktionen angegeben werden. Zu jeder Einstellmöglichkeit werden Hilfestellungen in Form von kurzen Erklärungen angezeigt. Die Steuerung des Wizards geschieht über vier Buttons. Mit dem Button *next* kann vor, mit dem Button *back* kann zurück geblättert werden. Wenn alle Parameter eingestellt sind, kann mit dem Button *finish* eine Interaktion generiert werden. Durch das Drücken des Buttons *cancel* wird die Erzeugung der Interaktion abgebrochen. Um Fehlern vorzubeugen, sind die einzelnen Buttons nur aktiv, sofern dies einen Sinn ergibt.

Das Konzept des Wizards hat sich in der Software-Entwicklung mehrfach bewährt. Es wird zum Beispiel bei der Installation und Konfiguration von Programmen verwendet.

In einem Wizard in ISIMAP können alle Eingaben mit der Tastatur getätigt werden. Für

Menü *File*	
Open...	Öffnen einer SVG-Datei
Save	Speichern der SVG-Datei
Save As...	Speichern der SVG-Datei unter einem neuen Namen
Close	Schliessen der SVG-Datei
Quit	Beenden von ISIMAP
Menü *Edit*	
undo	Löschen der letzten erstellten Interaktion
redo	Erneute Herstellung der zuletzt gelöschten Interaktion
Menü *Navigation*	
Initial View	Darstellung der SVG-Graphik in vollem Umfang
Zoom In	Einzoomen der SVG-Graphik
Zoom Out	Auszoomen der SVG-Graphik
Menü *Interaction*	
Set Map Extent	Einstellung, welcher Teil des SVG-Dokumentes den Kartenbereich darstellt
Zoom	Auswahl einer Zoom-Interaktion
Pan	Auswahl einer Pan-Interaktion
Change Style	Auswahl der Interaktion zur Hervorhebung eines SVG-Elementes
Highlight Layer	Auswahl einer Interaktion zur Hervorhebung von Kartenebenen
Layer On/Off	Auswahl der Interaktion zum Ein- und Ausblenden von Kartenebenen
Menü *About*	
About isiMap...	Anzeige von Angaben über ISIMAP

Tabelle 4.4: Beschreibung der Befehle in ISIMAP

die Eingabe der `ids` von SVG-Elementen steht eine zweite Möglichkeit zur Verfügung. `ids` müssen relativ oft angegeben werden, da durch sie die Elemente zur Auslösung von Interaktionen und die zu verändernden Bereiche der Karte referenziert werden. Da die Kartenautoren und -autorinnen bei einer grossen Anzahl von Kartenelementen nicht immer alle `ids` der SVG-Elemente im Kopf behalten können, ist die realisierte Erleichterung sicher sinnvoll. Sie besteht aus dem Anklicken eines Elementes in der angezeigten Karte. Die `id` des Elementes wird ausfindig gemacht und im Wizard eingetragen. Oftmals besteht ein Navigationselement jedoch aus einer Gruppe von SVG-Elementen. Deshalb erscheint unterhalb des Wizard-Feldes eine Auswahlliste (Abbildung 4.3), in welcher die DOM-Hierarchie vom angeklickten Element bis zum *root*-Element angezeigt wird. So kann eine Gruppe oder ein anderes übergeordnetes Element angegeben werden.

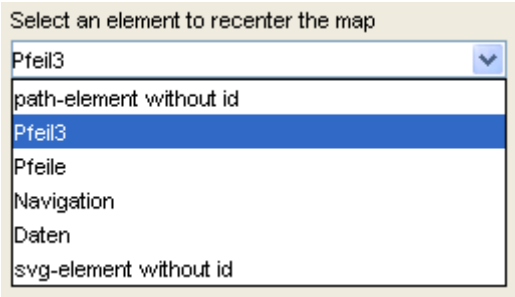

Abbildung 4.3: Die Auswahlliste

4.2.2 Architektur

Die Umsetzung der Anforderungen an ISIMAP ist mit Java realisiert worden. In dieser objektorientierten Sprache ist ein Programm aus Klassen aufgebaut. Um den minimalen Funktionsumfang von ISIMAP zu gewährleisten, sind 29 Klassen implementiert worden. Die Klassen sind aufgrund ihrer funktionalen Verwandtschaft in drei Packages organisiert worden. Ein Package ist grundsätzlich ein Mittel zur Strukturierung von Java-Projekten und beinhaltet mehrere zusammengehörende Klassen. Im Package `ch.unizh.geo.isimap` sind die Klassen mit übergreifender Funktionalität zu finden. Dazu gehören in erster Linie die Komponenten für den Aufbau des Layouts und zur Steuerung des Programms. Für die Manipulation des SVG- und JavaScript-Codes sind die Klassen im Package `ch.unizh.geo.isimap.js` verantwortlich. Die Klassen im Package `ch.unizh.geo.isimap.gui` stellen kleine, mehrmals eingesetzte Elemente der Benutzeroberfläche von ISIMAP dar. In den Tabellen 4.5, 4.6 und 4.7 sind die Aufgaben sämtlicher Klassen dieser Applikation kurz beschrieben. Für eine umfassendere Beschreibung der Funktionalität, der Attribute sowie der Methoden einer Klasse sei auf die Java-Dokumentation Javadoc auf der beiliegenden CD verwiesen.

Im Quellcode von ISIMAP sind die Klassen `ISIMapFrame`, `JSDocument` und `AbstractJSC` mit all ihren Subklassen besonders umfangreich. In ihnen ist ein grosser Teil der Funktionalität von ISIMAP implementiert. `isiMapFrame` definiert das Layout und die grundlegenden Elemente der Benutzeroberfläche. Ausserdem beinhaltet diese Klasse die Steuerung durch die Menübefehle, und sie ist für die Anzeige des SVG-Dokumentes zusändig. Die Klasse `JSDocument` übernimmt die vollständige Handhabung des JavaScript-Codes inkl. der Speicherung. Die grundlegenden Eigenschaften und Funktionen eines Wizards sind in der Klasse `AbstractJSC` abgelegt. Die Realisierungen (Abbildung 4.4) bestimmen exakt die vom Benutzer benötigten Angaben für eine Interaktion und nehmen die Änderungen am SVG-Dokument bei der Erzeugung einer solchen vor. Alle anderen Klassen sind Hilfsklassen, die kleine Bausteine in der Programmarchitektur darstellen. Spezielle Konzepte in der Umsetzung der Vorgaben werden im Abschnitt 4.3.1 vorgestellt.

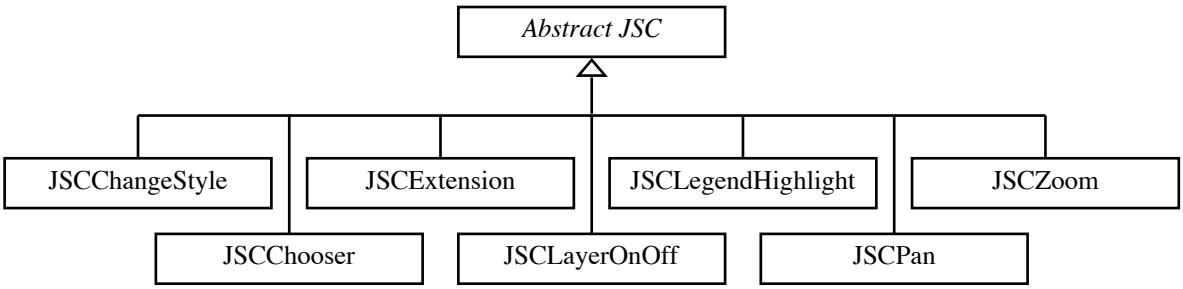

Abbildung 4.4: Die abstrakte Klasse *AbstractJSC* mit ihren Realisierungen

Klasse	Funktionalität
ISIMap	Starten von ISIMAP
ISIMapFrame	Benutzeroberfläche, Steuerung, Laden und Speichern von SVG-Dateien, Anzeigen der SVG-Graphik
ISIMapLog	Handhabung und Speicherung der log-Datei
UndoFragment	Stellt die Änderung eines Tags für die *undo*- und *redo*-Funktionen dar
UndoStep	Stellt alle Änderungen eines *undo*- oder *redo*-Befehles dar
UndoStepPanOverview	Stellt alle Änderungen eines *undo*- oder *redo*-Befehles der Interaktion *pan by mooving a rectangle in the overview* dar
UndoStepZoomRect	Stellt alle Änderungen eines *undo*- oder *redo*-Befehles der Interaktion *zoom by drag a rectangle* dar

Tabelle 4.5: Klassen des Package ch.unizh.geo.isimap

Klasse	Funktionalität
AbstractJSC	Grundklasse aller Wizards: Layout, Funktionalität der Buttons
JSCChangeStyle	Eingabe der Parameter für die Interaktion zur Hervorhebung eines Elementes
JSCExtension	Eingabe der Elemente des Kartenbereiches
JSCChooser	Standardanzeige, wenn kein Wizard aktiv ist
JSCLayerOnOff	Eingabe der Parameter für die Interaktion zum Ein- und Ausblenden von Kartenebenen
JSCLegendHighlight	Eingabe der Parameter für die Interaktionen zur Hervorhebung mehrerer Kartenelemente
JSCPan	Eingabe der Parameter für die Interaktionen zum Verschieben des Kartenbildes
JSCZoom	Eingabe der Parameter für die Interaktionen zum Zoomen
JSDocument	Handhabung und Speicherung des JavaScript-Codes
SVGAttributeHandler	Änderungen von Attributen von SVG-Elementen

Tabelle 4.6: Klassen des Package ch.unizh.geo.isimap.js

Klasse	Funktionalität
AttributeDeclaration	Layoutelement zur Spezifizierung von Werten von SVG-Attributen
AttributeDeclarationEvent	Ausgelöster *Event* bei korrekter Angabe der Werte in einer AttributeDeclaration
AttributeDeclarationListener	Reagiert, wenn ein AttributeDeclarationEvent ausgelöst wurde
ElementList	Layoutelement zur Auswahl mehrerer SVG-Elemente
ElementListEvent	Ausgelöster *Event* bei korrekter Angabe der Werte in einer ElementList
ElementListListener	Reagiert, wenn ein ElementListEvent ausgelöst wurde
LayerTable	Layoutelement zur Angabe von Kartenebenen und den dazugehörenden Legendeneinträgen
LayerTableEvent	Ausgelöster *Event* bei korrekter Angabe der Werte in einer LayerTable
LayerTableListener	Reagiert, wenn ein LayerTable Event ausgelöst wurde
StatusBar	Layoutelement zur Anzeige von Mitteilungen an den Benutzer

Tabelle 4.7: Klassen des Package ch.unizh.geo.isimap.gui

4.3 Implementierung

Die Implementierung von ISIMAP erfolgte auf einem Apple iBook G4 mit einem 800MHz-Prozessor und 640 MB RAM. Als Betriebssystem kam Mac OS X.3.8 zum Einsatz. Die Wahl der Programmierumgebung fiel auf *Eclipse 3.0* [7], welche bei der Erstellung von ISIMAP wertvolle Dienste leistete. Java lag in der Version 1.4.2 vor. Bei Fragen zur Programmierung ist hauptsächlich auf das Internet zurückgegriffen worden. Insbesondere das offizielle Java-Tutorial von Sun Microsystems [35] diente oft als reichhaltiges Nachschlagewerk. Bei Fragen zu Batik konnte in den meisten Fällen das Archiv der Mailing-List der Batik-User [2] weiterhelfen.

Die Java-Klassenbibliothek umfasst eine Vielzahl von Klassen und Methoden zur Erstellung von Applikationen. So konnte der grösste Teil der Implementierung mit Methoden aus dem grundlegenden Java-Funktionsumfang erfolgen. Bei der Erstellung von ISIMAP sind Klassen aus den Packages `java.awt`, `java.lang`, `java.io`, `java.text`, `java.util` eingesetzt worden. Für des Layout ist Swing verwendet worden. Zur Manipulation von SVG kamen Klassen aus den Packages `javax.xml`, `org.w3c.dom` und `org.xml.sax` zum Einsatz. Allerdings reichten die zur Verfügung stehenden Möglichkeiten nicht ganz aus. Deshalb musste, wie bereits im letzten Abschnitt angetönt, auf die Klassen der Bibliothek Batik zurückgegriffen werden. Diese ermöglichten einen einfachen Umgang mit SVG-Dokumenten.

Bei der Implementierung von ISIMAP stand die Erfüllung der Anforderungsspezifikation (Abbildung 4.1) im Zentrum. Diese diente als Anleitung beim Vorgehen der Programmierung. Viele der gestellten Anforderungen treffen auch für andere Applikationen, die in einem Zusammenhang mit SVG stehen, zu. Die weiteren Ausführungen in diesem Abschnitt beschäftigen sich deshalb mit neuen, vielfältig einsetzbaren Konzepten bei der Handhabung von SVG in Java-Programmen.

4.3.1 Ausgewählte Konzepte der Umsetzung

Auf den nächsten Seiten werden Ausschnitte aus dem Sourcecode von ISIMAP abgebildet und beschrieben, um ausgewählte Konzepte zur Umsetzung der Anforderungsspezifikation zu erklären. Die einzelnen Codefragmente umfassen alle relevanten Anweisungen der Implementierung eines Konzeptes. Sie können als Beispiele für die Handhabung von interaktiven SVG-Dokumenten angesehen werden und als Ausgangsbasis für weitere Computerprogramme in diesem Bereich dienen. Sie sind jedoch, wie gesagt, nur Ausschnitte. Um die gesamte Funktionalität von ISIMAP zu verstehen, kann der gesamte Sourcecode auf der beiliegenden CD studiert werden.

Laden, Anzeigen und Speichern von SVG-Dokumenten

Eine *must*-Anforderung an ISIMAP lautet: „Die geladenen Karten müssen graphisch dargestellt werden.“ Für die Anzeige von SVG-Dokumenten kann in Batik die Swing-Komponente `JSVGCanvas` eingesetzt werden. Diese beinhaltet die gleiche Funktionalität wie die Superklasse `JSVGComponent`, entspricht jedoch der JavaBean-Spezifikation [33].

```
import java.io.*;
import org.apache.batik.dom.svg.*;
import org.apache.batik.util.*;
import org.w3c.dom.svg.*;

private void loadSVG(File f) {
   try {
      String parser = XMLResourceDescriptor.getXMLParserClassName();
      SAXSVGDocumentFactory svgFactory =
         new SAXSVGDocumentFactory(parser);
      SVGDocument doc =
         svgFactory.createSVGDocument(f.toURL().toString());
      svgCanvasDynamic.setSVGDocument(doc);
   } catch (Exception exc) {

   }
}
```

Listing 4.1: Laden eines SVG-Dokumentes

```
import java.io.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;

private void saveSVG(File f) {
   try {
      FileOutputStream fos = new FileOutputStream(f);
      TransformerFactory tFactory = TransformerFactory.newInstance();
      Transformer transformer = tFactory.newTransformer();
      transformer.setOutputProperty(OutputKeys.INDENT, "yes");
      DOMSource source = new DOMSource(svgDocumentToSave);
      StreamResult result = new StreamResult(fos);
      transformer.transform(source, result);
      fos.close();
   } catch (Exception exc) {

   }
}
```

Listing 4.2: Speichern eines SVG-Dokumentes

Zur Darstellung einer SVG-Datei muss der `JSVGCanvas` ein `SVGDocument` übergeben werden. Schauen wir uns also zuerst die Umwandlung einer SVG-Datei in ein `SVGDocument` an. Für diesen Vorgang kann eine `SAXSVGDocumentFactory` eingesetzt werden (siehe Listing 4.1). In ISIMAP kann die gewünschte Datei in einem Öffnen-Dialog ausgewählt werden. Diese wird als Argument in die `SAXSVGDocumentFactory` eingespiesen, welche die Umwandlung automatisch und ohne weitere Angaben vornimmt. Das resultierende `SVGDocument` wird der `JSVGCanvas` zugewiesen, welche die Graphik aufbaut und anzeigt.

Der Vorgang der Übergabe des `SVGDocument`es bis zur Darstellung am Bildschirm (engl. *rendering*) kann abhängig vom Prozessor und dem Betriebssystem, insbesondere auf Mac OS X, einige Sekunden dauern. Dieser Prozess ist ein Bestandteil der `JSVGComponent` und läuft in fünf sequentiellen Phasen ab. Die einzelnen Schritte sind: Aufbau des DOM-Baumes, Aufbau des GVT-Baumes, Ausführung des `SVGLoadEvent`-Handlers, Darstellung des GVT-Baumes und Starten des Update-Threads bei dynamischen Dokumenten. Je nach gewählter Methode zur Anzeige des `SVGDocument`es müssen nicht alle Phasen durchlaufen werden. In ISIMAP kann jedoch auf keinen der Schritte verzichtet werden.

Die Anweisung `svgCanvasDynamic.setURI(f.toURL().toString());` kann als Alternative zum Laden und zur Anzeige des SVG-Dokumentes verwendet werden. Sie fasst die Schritte der Zeilen 8 bis 13 zusammen. Der Vorteil der verwendeten Methode liegt jedoch in der Möglichkeit zur Manipulation des `SVGDocument`es bevor es der `JSVGCanvas` zur Darstellung übergeben wird.

Das Speichern der SVG-Datei geschieht unter der Verwendung der XML-Packages der Java-Klassenbibliothek. Mit Hilfe eines `Transformers` kann der SVG-Code vor dem Speichern formatiert werden. In der Zeile 11 des Listings 4.2 wird zum Beispiel angegeben, dass der DOM-Hierarchie entsprechende Tabulatoren am Anfang jeder Zeile des SVG-Dokumentes eingefügt werden sollen. Für die Verarbeitung im `Transformer` wird aus dem modifizierten `svgDocumentToSave` zuerst eine `DOMSource` generiert. Der eigentliche Speichervorgang läuft dann über einen gewöhnlichen `FileOutputStream` ab.

Veränderung der Darstellung von SVG-Dokumenten

In ISIMAP kann die angezeigte SVG-Graphik gezoomt und der sichtbare Bildausschnitt verschoben werden (Abbildung 4.5). Durch die Implementierung dieser beiden Funktionen können bestimmte Bereiche der Graphik detaillierter dargestellt werden. Die vergrösserte Darstellung einzelner Elemente hilft den Benutzenden bei der Eingabe von `ids`. Die Elemente können so präzise mit der Maus angeklickt werden.

Die `JSVGCanvas` beinhaltet prinzipiell diese Funktionalität. Durch die Vererbung von der Klasse `JSVGComponent` erhält sie die entsprechenden Methoden. Die Anzeige einer SVG-Graphik innerhalb einer `JSVGCanvas` kann mit Hilfe von `Actions` der Swing-Bibliothek gesteuert werden. Eine `Action` kann grundsätzlich eingesetzt werden, wenn Funktionen durch verschiedene Handlungen der Benutzenden ausgelöst werden können. Sie können zum Beispiel durch das Drücken einer Tastenkombination oder durch die Auswahl eines Menüeintrages gestartet werden. In der Klasse `JSVGCanvas` sind u. a. `Actions` für Zoom-, Pan-, Rotations- oder Transformations-Funktionen vorhanden.

In Listing 4.3 ist ein Ausschnitt der Funktion `buildMenus()`, welche im Konstruktor für

```
import javax.swing.*;
import org.apache.batik.swing.*;

private void buildMenus() {
   JMenuBar menuBar = new JMenuBar();
   JMenu navigation = new JMenu("Navigation");
   Action zoomInAction = (Action)svgCanvasDynamic.getActionMap().get(
      JSVGCanvas.ZOOM_IN_ACTION);
   JMenuItem zoomIn = new JMenuItem(zoomInAction);
   zoomIn.setText("Zoom In");
   navigation.add(zoomIn);
   menuBar.add(navigation);
   this.setJMenuBar(menuBar);
}
```

Listing 4.3: Zoomen eines SVG-Dokumentes

```
import javax.swing.*;
import org.apache.batik.swing.*;

public class ISIMapFrame extends JFrame {

   private JPanel mainPanel = new JPanel(new BorderLayout());
   private JSVGScrollPane svgScrollPane;
   private JSVGCanvas svgCanvasDynamic;

   public ISIMapFrame() {

      svgCanvasDynamic = new JSVGCanvas(isiMapUserAgent, true,false);
      svgCanvasDynamic.setDocumentState(JSVGCanvas.ALWAYS_DYNAMIC);
      this.svgScrollPane = new JSVGScrollPane(svgCanvasDynamic);
      mainPanel.add(svgScrollPane, BorderLayout.CENTER);

   }

}
```

Listing 4.4: Darstellung von interaktiven SVG-Dokumenten mit Batik

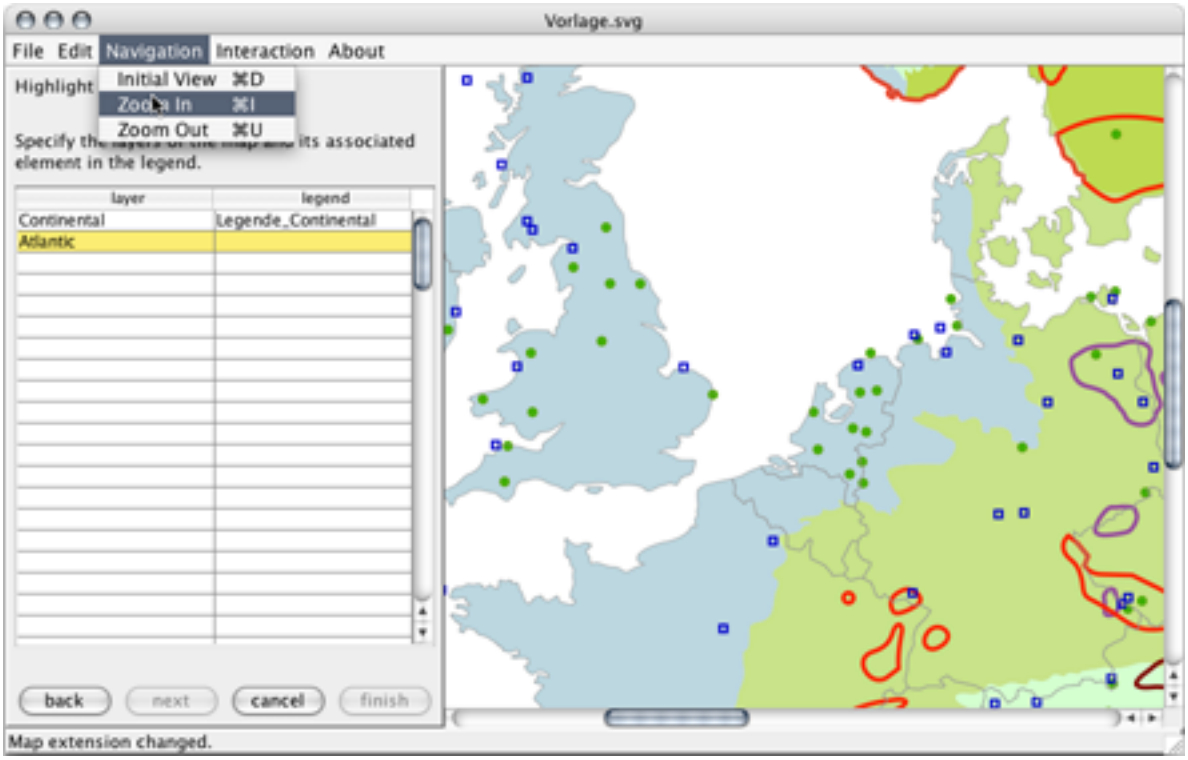

Abbildung 4.5: Vergrösserte Darstellung einer SVG-Graphik in ISIMAP

den Aufbau der Menüs aufgerufen wird, abgebildet. Die Menüs werden mit Swing-Klassen implementiert. Interessant für die Erzeugung einer Zoom-Funktion sind die Zeilen 7 und 8. Sie zeigen wie eine `Action` für den Aufruf einer Zoom-Funktion eingesetzt werden kann. Zuerst wird die vorgefertigte Funktion zum Einzoomen in der Variable `zoomInAction` abgelegt. Anschliessend wird diese `Action` einem Menüeintrag mit der Beschriftung *Zoom In* zugeordnet. Dies erlaubt es den Benutzenden vom ISIMAP durch die Auswahl dieses Menüeintrages die SVG-Graphik zu vergrössern. Auf dieselbe Art und Weise sind auch Funktionen für das Auszoomen (*Zoom Out*) und für die Anzeige in der Originalgrösse (*Initial View*) programmiert worden.

Für die Implementierung der Funktionalität zum Verschieben des Kartenbildes bietet sich eine elegantere Variante an. Das Code-Fragment in Listing 4.4 illustriert die Einbindung einer `JSVGCanvas` in der Klasse `ISIMapFrame`. Beachtenswert in diesem Zusammenhang ist insbesondere die Zeile 14. Die `JSVGCanvas` wird im Konstruktor eines Objektes der Klasse `JSVGScrollPane` aufgerufen. Das `JSVGScrollPane` ist ein Layoutelement mit Rollbalken (engl. *scroll bar*), durch welche der angezeigte Bereich einer SVG-Graphik verschoben werden kann. Diese Möglichkeit ist in vielen Computerprogrammen anzutreffen. Somit wird also die Anforderung der Anlehnung der Benutzeroberfläche an gängige Programme berücksichtigt.

Bei der Verwendung einer `JSVGCanvas` ist weiter die Voreinstellung zur Ausführung von interaktiven Elementen zu beachten. Die drei Modi `ALWAYS_DYNAMIC`, `ALWAYS_INTERACTIVE` und `ALWAYS_STATIC` legen den Status für das gewünschte Verhalten fest. Der Modus muss

```
import org.w3c.dom.svg.*;

public boolean changeInstruction(String id, String attribute, String value
    ) {
   this.svgID = id;
   this.svgAttribute = attribute;
   this.instruction = value;
   try {
      SVGElement svgElement = (SVGElement)svgDocumentToSave.getElementById
          (id);
      svgElement.setAttributeNS(null, attribute, value);
      isiMapFrame.getSvgCanvas().getUpdateManager().getUpdateRunnableQueue
          ().invokeLater(new Runnable () {
         public void run() {
            SVGElement e = (SVGElement) isiMapFrame.getSvgCanvas().
                getSVGDocument().getElementById(svgID);
            e.setAttributeNS(null, svgAttribute, instruction);
      }});
      return true;
   } catch (Exception exc) {
      return false;
   }
}
```

Listing 4.5: Ändern eines SVG-Attributes

beim Erzeugen des `JSVGCanvas` angegeben werden. Er kann zu einem späteren Zeitpunkt nicht mehr umgestellt werden, das Ein- und Ausschalten von Interaktionen ist also nicht mehr möglich. Im Modus `ALWAYS_DYNAMIC` sind alle Interaktionen aktiv. Dieser wird in ISIMAP verwendet, um der Anforderung einer Vorschau nachzukommen.

Manipulation von SVG-Dokumenten

Für die Manipulation der SVG-Elemente können dank Batik die DOM-Funktionen verwendet werden. Im Gegensatz zu JavaScript muss allerdings der Typ der Variablen angegeben werden. Ansonsten können die Befehle gleich eingesetzt werden. Sofern man sich mit dem DOM API auskennt, bereitet dieser Vorgang keine Schwierigkeiten.

Für die Handhabung der SVG-Dokumente durch ISIMAP ist allerdings ein Problem von entscheidender Bedeutung (Listing 4.5). Die `JSVGCanvas` dient nicht nur zur Anzeige des SVG-Dokumentes, sondern auch als Vorschau. Die angefügten Interaktionen sind voll aktiv: DIE Benutzenden können sie ausführen und somit die SVG-Graphik entsprechend verändern. Wenn nun aber das in der `JSVGCanvas` dargestellte SVG-Dokument gespeichert wird, wird immer die aktuelle, unter Umständen veränderte Ansicht gespeichert. Dies ist in diesem Fall aber nicht erwünscht, denn die mit ISIMAP bearbeitete SVG-Karte soll nach einer Veröffentlichung im Internet in einem Browser in ihrem ursprünglichen Design angezeigt werden. Für die Speicherung der SVG-Datei soll also die ursprünglich geladene Karte, erweitert um die angefügten Event-Handler und den JavaScript-Code, verwendet werden. Aus diesem Grund werden in

ISIMAP zwei Objekte des Typs `SVGDocument` verwendet. Die Variable `svgDocument` wird für die Anzeige in der `JSVGCanvas` verwendet. Das `SVGDocument` für die Speicherung ist in der Variable `svgDocumentToSave` abgelegt. Bei der Erzeugung einer Interaktion müssen beide Objekte berücksichtigt werden. Die Event-Handler müssen beiden angefügt werden.

Die Methode `changeInstruction(String id,String attribute,String value)` der Klasse `SVGAttributeHandler` ist in Listing 4.5 abgebildet. Diese ändert den Wert (`value`) eines Attributes (`attribute`) von einem SVG-Element (`id`). Wie oben gesehen, müssen das `svgDocumentToSave` und das `svgDocument` manipuliert werden. Deshalb sind die Anweisungen zur Manipulation auch in einem `try chatch`-Block zusammengefasst. Entweder werden die Event-Handler beiden oder keinem der beiden `SVGDocument`e angehängt.

Die Manipulation eines SVG-Elementes des `svgDocumentToSave` findet im Codefragment des Listings 4.5 in den Zeilen 8 und 9 statt. Durch die anschliessenden Anweisungen werden die Änderungen am `svgDocument` vorgenommen. Hier ist das Ganze ein bisschen komplizierter. Die im `svgDocument` registrierten DOM-Listener werden vom Canvas-Update-Thread aufgerufen. Deshalb muss bei der Veränderung der in der `JSVGCanvas` angezeigten SVG-Graphik der Weg über den `UpdateThread` genommen werden. Dabei wird der auszuführende JavaScript-Code in eine `Runnable`-Klasse gepackt und mit der Methode `invokeLater()` dem `UpdateThread` übergeben.

Auswahl von SVG-Elementen

Im Abschnitt 4.2.1 wurde gezeigt, wie die Benutzenden durch das Anklicken eines Elementes in der SVG-Graphik die entsprechende `id` in einen Wizard eintragen können. Schauen wir uns diesen Vorgang nun auf der technischen Ebene an (Listing 4.6). Für diesen Vorgang sind drei Teile der Klasse `ISIMapFrame` verantwortlich. In dieser wird ja auch das angezeigte `SVGDocument` referenziert.

Damit auf eine Aktion der Benutzenden reagiert werden kann, müssen zuerst die entsprechenden `Listener`, diese führen die Reaktion aus, angegeben werden. Dies geschieht in der Methode `registerListeners()`. Dazu muss zuerst das *root*-Element des `SVGDocument`es in ein `EventTarget` umgewandelt werden. Nur so können registrierte `Listener` informiert werden. Hier ist nur eine Aktion von Bedeutung, das Klicken mit der Maus auf ein Element der SVG-Graphik. Der `Listener` dieser Aktion, in diesem Fall eine `MouseClickAction`, wird in Zeile 18 am `EventTarget` registriert. Somit ist sichergestellt, dass bei jedem Anklicken eines Elementes in der SVG-Graphik die `MouseClickAction` aufgerufen wird und die gewünschte Reaktion erfolgt.

`Listener` können erst an ein `SVGDocument` angefügt werden, nachdem die DOM-Baumstruktur vollständig aufgebaut und allfällige Skripts initialisiert worden sind. Dies ist nach Abschluss der Phase 3 des Prozesses zur Übergabe eines `SVGDocument`es an eine `JSVGCanvas` (siehe Abschnitt Laden, Anzeigen und Speichern von SVG-Dokumenten) der Fall. Zur Sicherstellung dieser Vorgabe dienen die aufgeführten Anweisungen im Konstruktor der Klasse. Der `SVGLoadEventDispacherAdapter` wird informiert, sobald die Startskripts ausgeführt sind. Erst jetzt wird die Methode `registerListener()` aufgerufen.

Die `MouseClickAction` bestimmt die auszuführenden Anweisungen, wenn sie nach dem Anklicken eines Elementes vom `EventTarget` benachrichtigt wird. Mit Hilfe der DOM-

```
import org.apache.batik.swing.*;
import org.apache.batik.swing.svg.*;
import org.w3c.dom.*;
import org.w3c.dom.events.*;
import org.w3c.dom.svg.*;

public ISIMapFrame() {
   svgCanvasDynamic.addSVGLoadEventDispatcherListener(
      new SVGLoadEventDispatcherAdapter() {
         public void svgLoadEventDispatchStarted(
            SVGLoadEventDispatcherEvent e) {
               registerListeners();
   }});
}

private void registerListeners() {
   Element mdoc = svgCanvasDynamic.getSVGDocument().getRootElement();
   EventTarget doctarg = (EventTarget) mdoc;
   doctarg.addEventListener("click", new MouseClickAction(), false);
}

public class MouseClickAction implements EventListener {
   public void handleEvent(Event evt) {
      SVGElement targ = (SVGElement) evt.getTarget();
      wizardPanel.elementClicked(targ);
   }
}
```

Listing 4.6: Auswahl von SVG-Elementen

Funktion `getTarget()` wird zuerst das angeklickte `SVGElement` bestimmt. Dieses wird anschliessend an den Wizard, ein in der Variable `wizardPanel` referenziertes Objekt der Klasse `AbstractJSC` weitergegeben. Dieser bestimmt selber, wie das `SVGElement` verarbeitet werden soll. Je nach Wizard sind dies unterschiedliche Aktionen.

Erzeugung des JavaScript-Codes

Für alle Aktionen im Zusammenhang mit dem JavaScript-Code ist die Klasse `JSDocument` verantwortlich. Wie bereits erklährt, ist der Code, der in dieser Klasse erzeugt wird, für die Steuerung der Interaktionen in der SVG-Karte verantwortlich. Er wird im selben Verzeichnis wie die SVG-Datei in der Datei *isimapscripts.js* gespeichert. Im SVG-Dokument wird eine Referenz auf diese Datei eingefügt, sobald die erste Interaktion ausgewählt wurde.

Für die Speicherung werden Klassen aus dem Package `java.io` verwendet. Gespeichert wird nur der Code, der für die Ausführung der erstellten Interaktionen benötigt wird. Die Datei *isimapscripts.js* ist also keine JavaScript-Bibliothek, aus der nur ein Teil der Skripts bei der Betrachtung durch die Kartennutzenden aufgerufen wird. Alle erstellten Skripts werden wirklich benötigt. Somit kann die Grösse der Datei möglichst klein gehalten werden, was für eine kurze Übertragungszeit bei einer Veröffentlichung der interaktiven SVG-Karte im

```
private StringBuffer generateJSScripts(int target) {
   StringBuffer jsScripts = new StringBuffer();

   if (zoomIt > 0) {
      jsScripts = jsScripts.append("Skript 1");
   }
   if (zoomRect > 0) {
      if (target == JSDocument.TARGET_UPDATE_MANAGER) {
         jsScripts = jsScripts.append("Skript 2_um");
      } else {
         jsScripts = jsScripts.append("Skript 2_file");
      }
   }

   return jsScripts;
}
```

Listing 4.7: Zusammenstellung des JavaScript-Codes

Internet wünschenswert ist.

Dieser Mechanismus wird durch die Verwendung von Musterskripts (siehe Abschnitt 4.3.2) in der Klasse `JSDocument` ermöglicht. Jedes Skript wird in einem eigenen `String` gespeichert. Zu jedem Skript gib es ausserdem einen Zähler, welcher angibt, wie oft ein Skript durch die Auswahl von Interaktionen durch die Benutzenden verwendet wird. Logischerweise wird das Skript nur einmal gespeichert. Die genaue Anzahl ist aber notwendig, damit die Instanz von `JSDocument` auch nach einem undo-Schritt noch weiss, ob es das Skript überhaupt noch speichern soll.

Der komplette JavaScript-Code der benötigten Musterskripts kann mit den drei Methoden `generateJSGlobalVars(int target)`, `generateInitMapScript(int target)` und `generateJSScripts(int target)` zusammengestellt werden. Ein Fragment der letzten Methode ist in Listing 4.7 zu sehen. Die drei Zeichenketten `"Skript 1"`, `"Skript 2_um"` und `"Skript 2_file"` sind in diesem Beispiel für eine bessere Übersicht als Platzhalter für die eigentlichen JavaScript-Anweisungen eingesetzt worden. Die Aufteilung in drei Methoden ist einzig und allein aus organisatorischen Gründen erfolgt. Vor jedem Speicherungsvorgang werden sie ausgeführt, um den kompletten Code zu erzeugen. Ausserdem wird nach jeder Erzeugung einer Interaktion der Code in die Script-Engine von Batik eingespiesen, damit die Interaktionen in der Vorschau funktionieren. Da der Code für die beiden Verwendungen leicht variiert, wird das Argument `int target` der Methode übergeben. Mit diesem kann angegeben werden, für welche Aktion der Code eingesetzt wird. Die eigentliche Zusammenstellung der Skripts ist eine einfache Abfolge von `if`-Anweisungen, in welcher die benötigten Skripts aneinandergefügt werden.

4.3.2 Code der JavaScript-Interaktionen

Ein ausgesprochen ärgerlicher Mangel bei vielen interaktiven SVG-Karten im Internet ist ihre Browserabhängigkeit. Viele Applikationen funktionieren nur auf dem Betriebssystem Win-

dows in Kombination mit dem *Internet Explorer* einwandfrei. Zwar verwenden auch heute noch mehr als die Hälfte der Internet-Besucher und -Besucherinnen diese Umgebung. Im Mai 2005 besass der *Internet Explorer* einen Marktanteil von 65 % und zu über 90 % werden Betriebssysteme der Windows-Familie eingesetzt [43]. Aber dies ist noch lange kein Grund, Benutzende, die einen anderen Browser oder ein anderes Betriebssystem verwenden, auszuschliessen. Der Trend zur Verwendung von anderen Browsern [43] macht ein Umdenken unumgänglich. Ein Anspruch an die JavaScript-Interaktionen ist deshalb, browserunabhängig zu sein.

Die Lösung des Problems ist eigentlich ganz einfach. Wie es scheint, ist sie aber einigen, um nicht zu sagen vielen Autoren und Autorinnen von interaktiven SVG-Karten unbekannt. Der Ursprung der Browserabhängigkeit liegt in den meisten Fällen in der Einbettung des JavaScript-Codes in eine HTML-Datei. In dieser Konstellation wird die Script-Engine des Browsers zur Interpretation der Skripts benutzt und der Zugriff auf Plugin-Funktionen ist aus technischen Gründen nur in der oben genannten Kombination möglich. Browserunabhängigkeit kann durch die Verwendung der internen Script-Engine des *Adobe SVG Viewer* durch die direkte Einbettung des JavaScript-Codes im SVG-Dokument erlangt werden. In Listing 3.1 wurde gezeigt, wie das mit dem Einsatz eines `<script>`-Tags bewerkstelligt wird.

In einigen Projekten ist eine Koppelung von HTML-Elementen, z. B. Auswahllisten oder Formularfeldern, mit der SVG-Graphik wünschenswert. Wie erläutert sollte dies jedoch unterlassen werden. Die HTML-Elemente können auch mit SVG und JavaScript erzeugt werden. Dies erfordert zwar in vielen Fällen einen höheren Aufwand in der Realisierung, ermöglicht jedoch die Browserunabhängigkeit. Nach dieser Klarstellung können wir uns nun dem eigentlichen Code der JavaScript-Interaktionen zuwenden.

Für die Interaktionen wurde eine Reihe von Musterskripts erstellt, welche mit ISIMAP einer Karte angefügt werden können. Die Musterskripts einer Interaktion bestehen aus einer oder mehreren JavaScript-Funktionen. Um eine einfache und saubere Zuweisung an SVG-Elemente zu garantieren, wurde eine abgekapselte und eigenständige Implementierung der Musterskripts angestrebt. Die Musterskripts sollen sich gegenseitig so wenig wie möglich beeinflussen. Dies gelang zu einem grossen Teil. Durch die Komplexität in der Steuerung einer Karte mussten allerdings gewisse Abstriche gemacht werden. Ein Beispiel dafür ist die Funktionsweise der Übersichtskarte. Das Rechteck in der Übersichtskarte symbolisiert den angezeigten Kartenausschnitt und wird in der Interaktion für die Verschiebung des Kartenbildes eingesetzt. Andererseits beeinflussen auch die Zoom-Interaktionen das Rechteck. Mit jedem erfolgten Zoom-Vorgang ändert sich dessen Grösse. Die beiden Interaktionen beeinflussen sich also gegenseitig. Dieses Verhalten spiegelt sich auch in den JavaScript-Funktionen wieder. Die Beziehungen zwischen den Musterskripts machte eine besonders ausgeklügelte Programmierung der drei Methoden für die Zusammenstellung der Skripts in der Klasse `JSDocument` notwendig. Gewisse Anweisungen in einer JavaScript-Funktion werden abhängig davon, ob eine bestimmte andere Interaktion der Karte angefügt ist, im Code eingefügt oder weggelassen.

Der erstellte JavaScript-Code ist aus drei Teilen aufgebaut. Im ersten Bereich werden die globalen Variablen definiert, z. B. der aktuelle Zoom-Faktor der Karte. Anschliessend ist die Funktion `initMap(evt)` im JavaScript-Code zu finden. Diese ist insofern speziell, als dass sie unabhängig von den angefügten Interaktionen immer erstellt wird. Sie wird durch den Event-

Handler `onload` des `<svg>`-Tags gestartet. In ihr sind Anweisungen zur Initialisierung der restlichen JavaScript-Funktionen zu finden. Je nach Umfang der Interaktionen beinhaltet die `initMap(evt)`-Funktion mehr oder weniger Anweisungen. Für die Zusammenstellung der einzelnen Code-Zeilen dieser Funktion wurde in der Klasse `JSDocument` extra die Methode `generateInitMapScript(int target)` erstellt. Diese entscheidet, welche Code-Fragmente benötigt werden. Der Rest des Codes besteht aus den einzelnen JavaScript-Funktionen zur Umsetzung der Interaktionen.

Ein wichtiger Punkt der Anforderungsspezifikation ist die Unterstützung von SVG-Dateien, die mit Adobe *Illustrator* exportiert wurden. Normalerweise wird, wie wir im Abschnitt 2.3 gesehen haben, der SVG-Code vor der Verknüpfung mit den Event-Handler von Hand angepasst und optimiert. Dieser Schritt sollte bei der automatischen Erzeugung der Interaktionen wegfallen, um die Kartenautoren und -autorinnen auch von dieser Last zu befreien. Die Anpassung der JavaScript-Funktionen an den unveränderten SVG-Code ist allerdings eine nicht zu unterschätzende Herausforderung. Insbesondere die mangelhaften Optionen für den SVG-Export aus Adobe *Illustrator* resultierten in einer zum Teil aufwendigen Programmierung des JavaScript-Codes. Im JavaScript-Code sind deshalb einige auf den ersten Blick spezielle, aber notwendige Konstrukte anzutreffen. An dieser Stelle werden deshalb exemplarisch zwei Beispiele zur Illustration dieser Anpassungen vorgestellt.

Deklaration des Kartenbereiches

Ein Problem beruht auf der Funktionsweise der Interaktionen zum Zoomen und Verschieben des Kartenbildes. Zum besseren Verständnis wird die grundlegende Funktionsweise dieser Interaktionen an dieser Stelle kurz erläutert. Über das `viewBox`-Attribut des `<svg>`-Tags kann der angezeigte Bereich eines SVG-Dokumentes gesteuert werden. Da nun aber nicht das ganze SVG-Dokument verändert dargestellt werden soll, sondern nur das Kartenbild, müssen die Elemente des Kartenbildes von einem eigenen, im eigentlichen SVG-Dokument eingebetteten `<svg>`-Tag umgeben sein. Durch die Änderung des Wertes des `viewBox`-Attributes des eingeschlossenen `<svg>`-Tags kann ein Teilbereich des gesamten SVG-Dokumentes vergrössert, verkleinert oder verschoben werden.

Problematisch ist nun, dass in Adobe *Illustrator* keine eingebetteten `<svg>`-Tags erstellt werden können. Ein SVG-Dokument besteht aus genau einem `<svg>`-Tag, in welchem alle gezeichneten Elemente zu liegen kommen. Eine Trennung zwischen dem Kartenbild und den Randangaben ist nur durch die Verwendung von Ebenen resp. Gruppen möglich. Der eingebettete `<svg>`-Tag muss also automatisch erstellt werden, am besten ohne dass die Benutzenden etwas davon mitbekommen.

Zur Lösung dieses Problems wurde eine JavaScript-Funktion programmiert, welche im SVG-Dokument einen eingebetteten `<svg>`-Tag erstellt und die Elemente des Kartenbildes in diesen Tag verschiebt. Die Funktion nennt sich `svgAroundMap()` und wird in der Funktion `initMap(evt)`, also nach dem Laden der SVG-Datei, aufgerufen. Der relevante Code ist in Listing 4.8 zu sehen.

ISIMAP kann nicht selber erkennen, welche Teile des SVG-Dokumentes die Kartenelemente darstellen. Die Benutzenden müssen deshalb den Kartenbereich definieren, bevor sie eine der erwähnten Interaktionen erzeugen können . Im entsprechenden Wizard können sie eine

```
var svgNS='http://www.w3.org/2000/svg';
var mapGroupID='Karte';
var mapBorderID='Rahmen';

function svgAroundMap() {
   var mapBorder = svgdoc.getElementById(mapBorderID);
   var bb = mapBorder.getBBox();
   var mapX = bb.x;
   var mapY = bb.y;
   var mapWidth = bb.width;
   var mapHeight = bb.height;
   var mapVBWidth = mapX*1 + mapWidth*1;
   var mapVBHeight = mapY*1 + mapHeight*1;
   var g = svgdoc.getElementById(mapGroupID);
   map = svgdoc.createElementNS(svgNS, 'svg');
   map.setAttributeNS(null, 'id', 'map');
   map.setAttributeNS(null, 'x', '+mapX);
   map.setAttributeNS(null, 'y', '+mapY);
   map.setAttributeNS(null, 'width', '+mapWidth);
   map.setAttributeNS(null, 'height', '+mapHeight);
   map.setAttributeNS(null, 'viewBox', mapX+' '+mapY+' '+mapVBWidth+' '+
      mapVBHeight);
   var gParent = g.getParentNode();
   gParent.insertBefore(map, g);
   map.appendChild(g.cloneNode(true));
   gParent.removeChild(g);
}
```

Listing 4.8: Erzeugung und Einbettung der Kartenelemente

Gruppe, welche die Kartenelemente enthält, sowie einen rechteckigen Rahmen, welcher die Grösse des Kartenausschnittes definiert, angeben. Die `ids` von diesen werden in den Variablen `mapGroupID` und `mapBorderD` abgelegt. In der Funktion `svgAroundMap()` selber wird zuerst die Grösse des Kartenbildes für die Angabe der Position und Grösse des eingebetteten `<svg>`-Tags sowie der Wert für das Attribut `viewBox` berechnet (Zeilen 8 bis 13). Dazu wird das einschliessende Rechteck (engl. *bounding box*) um den Kartenrahmen gelegt (Zeile 7). In einem zweiten Schritt wird das eingebettete `<svg>`-Tag erstellt und die Attribute werden angegeben (Zeilen 15 bis 21). Das Tag wird im SVG-Dokument vor der Gruppe mit dem Kartenbild eingefügt und die Elemente des Kartenbildes in das `<svg>`-Tag verschoben (Zeilen 22 bis 24). Zum Schluss wird die ursprüngliche Gruppe mit dem Kartenbild gelöscht (Zeile 25), damit diese Elemente nicht zweimal erscheinen.

Attribute von Gruppen mit mehreren untergeordneten Elementen ändern

Eine weitere Herausforderung ergab sich im Zusammenhang mit Interaktionen zum Hervorheben von mehreren Elementen, z. B. wenn beim Überfahren eines Legendeneintrages die entsprechende Ebene in der Karte anders eingefärbt werden soll. In manuell überarbeiteten SVG-Dokumenten wird dazu das `fill`-Attribut in der Gruppe, die alle Elemente einer Klas-

```
function simTraverseOn(node) {

   //Änderung der Attribute

   if (node.childNodes != null) {
      for (var i=0; i < node.childNodes.length; i++) {
         simTraverseOn(node.childNodes.item(i));
      }
   }
}
```

Listing 4.9: Rekursives Abarbeiten der untergeordneten Elemente

se enthält, angegeben. Eine JavaScript-Funktion muss nur das Attribut dieses einen SVG-Elementes ändern. In Adobe *Illustrator* kann aber die Farbe nur von einzelnen geometrischen Elementen und nicht von Ebenen angegeben werden. Jedes Element der Gruppe verfügt also über das `fill`-Attribut. Wird nun das `fill`-Attibut in der Gruppe geändert, geschieht nichts, da die Attribute der untergeordneten Elemente eine höhere Priorität geniessen. Die JavaScript-Funktion muss also alle der Gruppe untergeordneten Elemente berücksichtigen.

Die erforderliche Funktionalität ist mit einer Rekursion verwirklicht worden. In Listing 4.9 ist ein Ausschnitt der Funktion `simTraverseOn(node)` ersichtlich. Als Argument `node` kann der oberste Knoten, in diesem Fall das Gruppenelement der zu verändernden Ebene, übergeben werden. Zuerst wird das Attribut auf seinen neuen Wert gesetzt, anschliessend folgt die rekursive Abarbeitung der untergeordneten Elemente. Es wird geprüft, ob überhaupt untergeordnete Elemente vorhanden sind. Ist dies der Fall, werden diese sequentiell abgearbeitet. Für jedes der Elemente wird wiederum die Funktion `simTraverseOn(node)` aufgerufen. Das Attribut wird geändert und die allfälligen untergeordneten Elemente werden ermittelt und bearbeitet. Dieser Algorithmus garantiert, dass alle zu einer Gruppe gehörenden Elemente verändert werden.

5 Beurteilung

Im letzten Kapitel haben wir gesehen, wie die Realisierung von ISIMAP angegangen wurde. Doch die Implementierung eines Computerprogrammes ist nur das eine. Entscheidend ist, ob die angestrebten Erleichterungen für die Benutzenden auch in geeigneter Form erreicht worden sind. Ein Programm kann über eine noch so umfangreiche Funktionalität verfügen, wenn die Benutzenden mit der Umsetzung nicht zurecht kommen, werden sie es nicht einsetzen. Die ganze Arbeit wäre vergebens gewesen. Zur Beurteilung der Realisierung von ISIMAP ist deshalb eine Evaluation durchgeführt worden.

Die Evaluation besteht aus zwei Teilen. Der Abschnitt 5.1 umfasst die formale Überprüfung der Implementierung der Anforderungsspezifikation. Im zweiten Teil wird mit einer empirischen Untersuchung eine Bewertung von ISIMAP vorgenommen. Dazu ist ISIMAP potentiellen Benutzern und Benutzerinnen vorgestellt worden. Durch das Lösen von Übungsaufgaben haben sie das Programm besser kennenlernen können. In einer Benutzerbefragung sind quantitative und qualitative Daten zur Einschätzung der Bedeutung eines Werkzeuges zur automatischen Erzeugung von JavaScript-Interaktionen für SVG-Karten sowie Daten zur Bewertung der Benutzerfreundlichkeit gesammelt worden. Erläuterungen zum Vorgehen der Evaluation sind im Abschnitt 5.2 zu finden. Die Auswertung der Angaben zur Beurteilung des Programms ist im Abschnitt 5.3 zusammengefasst.

5.1 Umsetzung der Anforderungspezifikationen

Zur Beurteilung von ISIMAP ist es angebracht, die Umsetzung der Anforderungsspezifikationen zu betrachten. So können eventuell Schwachpunkte von ISIMAP erklärt werden und Verbesserungs- und Erweiterungsmöglichkeiten für eine allfällige Weiterentwicklung des Programms vorgeschlagen werden.

In der Tabelle 5.1 ist zu sehen, dass die Punkte der Kategorien *must*, *should* und sogar *could* der Anforderungen an das Werkzeug beinahe vollständig implementiert werden konnten. Die

	Anforderungen	
Priorität	erfüllt	nicht erfüllt
must	1, 2, (3,) 4, 5, 6, 7, 8, (9,) (10,) 11	
should	1, (2,) 3, 4, 5, 6	
could	2, 3, 4	1
would		1, 2, 3, 4

Tabelle 5.1: Umsetzung der Anforderungen an ISIMAP
vgl. Tabelle 4.1

Umsetzung der drei wichtigen Prioritätenklassen der gewünschten Interaktionen (Tabelle 5.2) weist grössere Lücken auf, doch auch hier ist ein Grossteil der Anforderungen realisiert worden.

In beiden Tabellen ist klar ersichtlich, dass hauptsächlich Anforderungen mit niedriger Priorität nicht realisiert worden sind. Hier kam die Stärke des MoSCoW-Systems zur Einteilung von Anforderungen zum Tragen. Durch den zeitlichen Rahmen der Implementierung mussten Abstriche gemacht werden. Dank der Einteilung in Prioritäten sind aber fast nur unwichtige Punkte der Anforderungsspezifikationen weggefallen. Dies ist durchaus zulässig.

Im Nachhinein zeigt sich, dass einige Punkte für eine Einteilung in die beiden Kategorien „erfüllt“ oder „nicht erfüllt“ zu vage formuliert worden sind. In der Tabelle 5.1 sind sie in Klammern gesetzt. Vier Punkte können nicht klar in eine der beiden Kategorien eingeordnet werden. Erstens ist dies die Anforderung, dass das Programm intuitiv und benutzerfreundlich sein soll (Punkt *must* 9). Zur Überprüfung dieses Kriteriums wurde eine Benutzerbefragung eingesetzt (siehe Abschnitt 5.3). Wir werden sehen, dass die Frage nach der Umsetzung dieser Anforderung nicht klar mit Ja oder Nein beantwortet werden kann. Zweitens ist die Forderung nach einem visuellen Feedback über den Programmstatus nicht vollständig erfüllt (Punkt *must* 10). Ein visuelles Feedback wird in ISIMAP zwar angezeigt, es wäre jedoch noch stark ausbaufähig. Drittens ist auch die Anforderung, dass das Programm möglichst viele JavaScript-Interaktionen anbieten soll, in Klammern gesetzt (Punkt *should* 2). Die grundlegenden Interaktionen der Web-Kartographie sind mit ISIMAP erstellbar. Allerdings können die Anzahl und die Variationen der möglichen Interaktionen ebenfalls noch stark ausgebaut werden.

Die meisten Probleme bei der Umsetzung bereitete die Unterstützung von Adobe *Illustrator* (Punkt *must* 1). Die Priorität dieser Anforderung ist sehr hoch eingestuft und musste unbedingt berücksichtigt werden. Die Verwendung von Batik erlaubt es den Benutzenden von ISIMAP zwar alle SVG-Dateien zu laden, unabhängig davon, wie diese erzeugt wurden. Da die SVG-Exportfunktion von Adobe *Illustrator* allerdings Schwächen aufweist, mussten als Konsequenz einige andere Anforderungen mit hoher Priorität zumindest teilweise fallen gelassen werden. Ein Beispiel dafür ist der Punkt 3 der *must*-Anforderungen an ISIMAP, das Öffnen von bearbeiteten SVG-Dateien mit diesem Programm. Das Öffnen der interaktiven Karte ist an und für sich kein Problem. Doch wenn die mit Adobe *Illustrator* überarbeitete Karte wieder gespeichert wird, fällt die Referenz zum JavaScript-Dokument aus dem SVG-Code. Die mit ISIMAP angefügten Attribute der Event-Handler bleiben zwar bestehen, da aber die JavaScript-Funktionen bei der Ansicht der SVG-Karte in einem Browser nicht geladen werden, sind alle Interaktionen inaktiv.

Eine weitere Schwachstelle der SVG-Exportfunktion von Adobe *Illustrator* tritt bei der Verwendung von Textelementen zum Vorschein. Die `ids` von diesen werden nicht in das SVG-Dokument geschrieben. Entsprechend können sie auch nicht durch JavaScript manipuliert werden. Dieses Verhalten verunmöglichte die Implementierung mehrerer Interaktionen: dynamischer Massstab, Einblenden von Informationen, Anzeige von Tooltips und Labels, Koordinatenanzeige sowie das Zoomen durch die Eingabe eines Prozentwertes. Für einen Teil dieser Interaktionen sind zwar versuchsweise Musterskripts erstellt worden, diese sind jedoch nicht in ISIMAP integriert.

	Anforderungen	
Priorität	erfüllt	nicht erfüllt
must	1, 2, 3, 5, 6, 7, 8, 10	4, 9
should	1, 3	2
could	3, 4	1, 2, 5, 6, 7
would		1,2, 3, 4, 5, 6, 7, 8, 9, 10

Tabelle 5.2: Umsetzung der gewünschten mit ISIMAP erstellbaren Interaktionen vgl. Tabellen 4.2 und 4.3

5.2 Evaluation

Das Bedürfnis nach Benutzerfreundlichkeit nimmt seit Jahren zu und sollte integrativer Bestandteil von moderner Software sein [Nielsen, 1993]. Sie muss wie die syntaktische und inhaltliche Richtigkeit des Programm-Codes überprüft werden. Für die Evaluation von Benutzerschnittstellen werden normalerweise empirische Methoden eingesetzt [Mack und Nielsen, 1994]. Derzeit gibt es allerdings keine allgemein anerkannte Methodik zur Prüfung von Software, aber immerhin eine Reihe von einsetzbaren Verfahren [4]. Geeignete Resultate liefern die Aufmerksamkeitsanalyse, der Einsatz von Fokusgruppen, die Experteninspektion oder die Verwendung von Fragebögen. Für die Beurteilung der Benutzerfreundlichkeit von ISIMAP wurde mit einem Fragebogen gearbeitet. Die Daten fallen so in numerischer Form an und können einfach ausgewertet werden. Zusätzlich wurden den Testpersonen Fragen für eine persönliche Einschätzung in mündlicher Form gestellt.

Für die Evaluation von Benutzerfreundlichkeit kann auf eine Reihe von standardisierten Fragebogen zurückgegriffen werden. Eine Übersicht der gebräuchlichen Fragebögen ist auf der Webseite von Usability Net [39] zu finden. Viele der Fragebögen sind nur in englisch verfügbar und kommen deshalb nicht für eine Verwendung in dieser Arbeit in Frage. Für die Bewertung von ISIMAP ist der deutschsprachige Fragebogen zur ISO-Norm 9241/10 eingesetzt worden. Dieses Verfahren ist ein einfaches Mittel zur Bewertung von Software. Es ist kein Expertenverfahren und kann somit ohne fremde Unterstützung durchgeführt werden. [Burmester et al., 1997] Der Fragebogen ist in der Praxis abgesichert, zuverlässig, kostenlos verfügbar und praktisch einsetzbar hinsichtlich Aufwand und Verständlichkeit [4]. Nach Bräutigam [4] sollte im Anschluss an die Beurteilung durch diesen Fragebogen den Probanden und Probandinnen die Möglichkeit geboten werden, ihre Meinung in ihren eigenen Worten darzulegen.

Der Fragebogen (siehe Anhang A) besteht aus 35 Fragen und ist somit in relativ kurzer Zeit auszufüllen. Die Fragen sind allgemein und leicht verständlich. Sie können somit auch ohne vorbereitende Schulung beantwortet werden [4]. Die Fragen sind in die sieben Kategorien Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Fehlertoleranz, Individualisierbarkeit und Lernförderlichkeit eingeteilt. Für die Beantwortung steht den Befragten jeweils eine siebenstufige Skala zur Verfügung, in welcher sie ihre Zustimmung oder Ablehnung ausdrücken können. Für die Ermittlung von ISIMAP-spezifischen Daten wurde der Fragebogen um einige Fragen erweitert. Diese beziehen sich auf einen möglichen Einsatz von ISIMAP und auf den Wissensstand der Befragten in Bezug

auf interaktive SVG-Karten.

Zur Durchführung der Untersuchung wurden Teilnehmer und Teilnehmerinnen ausgewählt, die auf Grund ihrer beruflichen Tätigkeit Interesse an ISIMAP haben könnten oder die auf ihr umfangreiches Vorwissen in der Thematik interaktive SVG-Kartographie für eine fachliche Beurteilung zurückgreifen können. Die Probanden und Probandinnen sollten über ein gutes kartographisches und über ein grundlegendes Wissen über SVG verfügen. Ansonsten würden sie die Einsatzmöglichkeiten von ISIMAP nicht vollständig verstehen, was eine unvoreingenommene Beurteilung verunmöglichen würde. Unter den Befragten sind zum Teil auch Personen, die bereits bei der Ermittlung der Anforderungen mitgewirkt haben. Diese können die Vorstellungen an das Werkzeug am besten mit der Umsetzung vergleichen. Neben der Definition des Profils der Untersuchungsteilnehmer und -teilnehmerinnen stellt sich noch die Frage nach der Anzahl der Testpersonen. Je komplexer und umfangreicher die Funktionalität eines Programms ist, umso mehr Testpersonen benötigt man, um statistisch repräsentative Ergebnisse in einer Benutzerbefragung zu erhalten. Aus rein praktischen Erwägungen reichen unter gewissen Umständen schon sechs bis acht Personen. [Lorenzen-Schmidt, 2003] Obwohl der Funktionsumfang von ISIMAP im Vergleich zu anderen Applikationen relativ gering ist, sind 13 Personen für die Evaluation befragt worden. So konnte eine breit abgestützte Einschätzung vorgenommen werden.

Nicht nur die Rekrutierung, sondern auch ein klar definierter Ablauf der Untersuchung ist entscheidend für die Qualität der ermittelten Daten. Eine wichtige Frage ist, wie die Testpersonen das Programm kennen lernen sollen. Nach Karat [Karat, 1994] bieten sich dazu zwei Möglichkeiten an. Erstens die selbständige Exploration, bei welcher die Benutzenden die Funktionalität des Programmes in Eigenregie „entdecken“. Die zweite Variante ist das Lösen von vordefinierten Aufgaben, welche den Benutzenden die Möglichkeiten eines Programms näher bringen. Karat kommt zum Schluss, dass beide Techniken erfolgreich eingesetzt werden können. Wenn die typischen Aufgaben eines Systems bekannt sind, ist es jedoch angemessen, vordefinierte Szenarien zu verwenden. Für diese Evaluation wurde der zweite Ansatz gewählt, da aus den Zielen der Arbeit und der Anforderungsspezifikation an ISIMAP die Anwendungsmöglichkeiten klar hervorgehen. Die Teilnehmenden mussten einer statischen, aus Adobe *Illustrator* exportierten SVG-Karte drei vorgegebene Interaktionen anfügen. Dabei handelt es sich um je eine Interaktion zum Zoomen, zum Verschieben des Kartenbildes und zur Hervorhebung von Kartenelementen.

Der vollständige Ablauf der Benutzerbefragung umfasste vier Schritte. Zuerst wurde den Testpersonen eine interaktive, mit ISIMAP erstellte SVG-Karte präsentiert, um ihnen zu zeigen, was mit dem Programm möglich ist. Anschliessend wurden die Beispielaufgaben gestellt, und die Teilnehmenden versuchten diese selbständig zu lösen. Es war ihnen jedoch freigestellt, ob sie das Tutorial zu Hilfe nehmen wollten. Während der Ausführung der Übungsaufgaben wurde ihnen vom Übungsleiter „über die Schulter geblickt “ und schriftlich festgehalten, bei welchen Schritten Schwierigkeiten entstanden oder Fragen auftauchten. Aus diesen Problempunkten konnten Verbesserungs- und Erweiterungsvorschläge angeleitet werden. Ausserdem konnten durch das Stoppen der benötigten Zeit für die Erzeugung von Interaktionen Werte für einen Vergleich mit der manuellen Programmierung ermittelt werden. Durch das Lösen von Beispielaufgaben wurde sichergestellt, dass sich die Probanden und Probandinnen bei der anschliessenden Befragung mit ISIMAP auskannten. Ein grundsätzlicher Schwachpunkt von

Fragebögen besteht darin, dass die Beantwortung beziehungsweise die Beurteilung der Fragen stark vom Erinnerungsvermögen, der Selbstwahrnehmung und der Aufmerksamkeit der Probanden und Probandinnen abhängt und sowohl für unwillkürliche Fehler und Verzerrungen als auch für absichtliche Verfälschungen anfälliger ist als verhaltensbasierte Benutzertests [Hamborg et al., 2003]. Es bietet sich also an, den Fragebogen sofort nach dem Lösen der Übungsaufgaben direkt am Arbeitsplatz auszufüllen. So ist die Erinnerung noch frisch und einzelne Arbeitsschritte der Übungsaufgaben können noch einmal ausprobiert werden. Die mündlichen Interviews wurden ebenfalls direkt anschliessend durchgeführt.

Die Auswertung des Fragebogens liefert erste Hinweise auf ergonomische Schwachstellen von ISIMAP. Ausgangspunkt für die numerische Bewertung einer Software sind die Antworten der Untersuchungspersonen. Die Einträge in der siebenstufigen Skala wurden in Zahlen umgewandelt, Zustimmung entspricht dem Wert +3, Ablehnung dem Wert -3. Die fünf Antworten einer Kategorie wurden für jeden Benutzenden zusammengezählt, was Gesamtbewertungen für die einzelnen Prüfprinzipien ergab. Anschliessend werden die Mittelwerte der Summen aller Fragebögen berechnet. Wenn der resultierende Mittelwert über dem Kriteriumswert liegt, also über dem Wert 5, so besteht kein dringender Verbesserungsbedarf der Software hinsichtlich dieses Prüfkriteriums. (nach [Burmester et al., 1997])

5.3 Diskussion der Resultate

In diesem Abschnitt wird besprochen, wie die Resultate der Benutzerbefragung ausgefallen sind. Neben der Präsentation der Daten werden Überlegungen zu den einzelnen Aspekten von ISIMAP dargelegt. Auf negativ beurteilte Punkte wird besonders ausführlich eingegangen.

5.3.1 Profil der Befragten

Zur Einschätzung der Beurteilung ist es wichtig zu wissen, wie gut sich die Befragten mit der Erstellung von interaktiven SVG-Karten auskennen. Im Abschnitt 5.2 wurde erläutert, dass in der Benutzerbefragung Probanden und Probandinnen mit mindestens grundlegenden Kenntnissen in SVG zu Einsatz kommen sollten. Abbildung 5.1 zeigt, dass dies der Fall war. Nur eine Testpersonen verfügt über keine Kenntnisse in diesem Bereich. JavaScript ist den meisten Personen ebenfalls bekannt, aber nicht mehr ganz so vielen.

Abbildung 5.2 deutet darauf hin, dass nicht alle Befragten mit Kenntnissen in SVG, diese auch einsetzten, um Karten zu produzieren. Ein Drittel der Befragten gab an, nie SVG-Karten zu erzeugen. Die anderen beiden Drittel verteilen sich auf Personen die selten resp. oft SVG-Karten erstellen. Interaktionen werden von den Befragten relativ oft in die produzierten SVG-Karten eingebaut.

5.3.2 Benutzerfreundlichkeit

Die Resultate zur Benutzerfreundlichkeit von ISIMAP sind in Abbildung 5.3 dargestellt. Bei den Kriterien Aufgabenangemessenheit, Steuerbarkeit, Erwartungskonformität und Lernförderlichkeit besteht kein Handlungsbedarf. In der mündlichen Befragung wurden zwei Eigen-

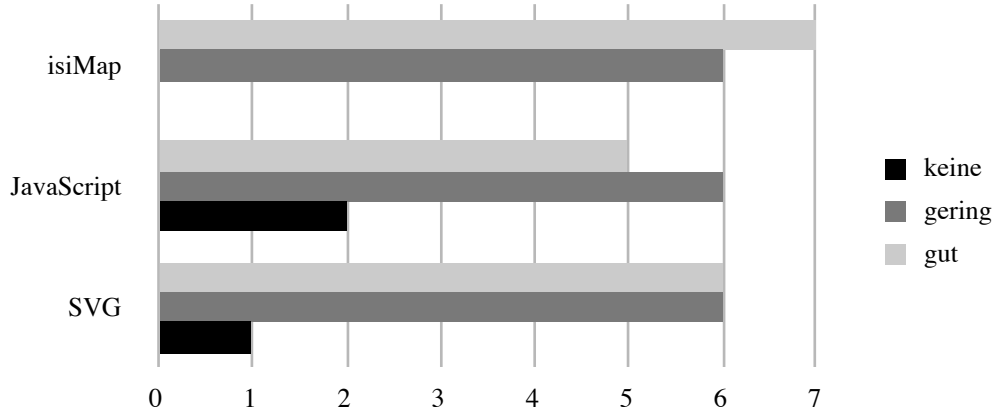

Abbildung 5.1: Kenntnisse der Befragten

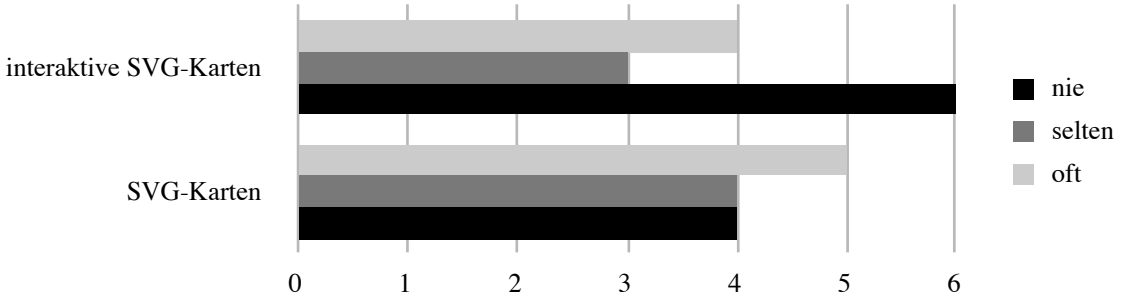

Abbildung 5.2: Häufigkeit der Erstellung von SVG-Karten der Befragten

schaften von ISIMAP besonders hervorgehoben. Zum einen sei das Layout und die Anordnung der Menübefehle sehr übersichtlich. Man erkenne schnell, was mit ISIMAP möglich ist. Zum anderen wurde gelobt, wie schnell die Handhabung des Programms erlernt werden kann. Bereits nach dem Lösen der drei Übungsaufgaben beherrschte die Hälfte der Probanden und Probandinnen das Programm gut (Abbildung 5.1). Die Lernförderlichkeit sei insbesondere an der Konzentration auf die wichtigste Funktionalität zurückzuführen.

Das Kriterium Fehlertoleranz liegt knapp unter der Grenze von fünf Punkten. Vorauszuschicken ist, dass bei vielen Testpersonen keine Fehlermeldungen aufgetreten sind, da sie die Übungsaufgaben korrekt gelöst haben. Diese Benutzer haben deshalb bei den meisten Fragen den mittleren Wert (0) angegeben, da sie dieses Kriterium nicht beurteilen konnten. Dies ist eigentlich erfreulich, zieht aber den Resultatwert nach unten. Vier Benutzer gaben an, dass die Fehlermeldung zu spät erscheinen würden. So zeigt ISIMAP zum Beispiel die Fehlermeldung erst an, wenn der *finish*-Button eines Wizards gedrückt wird und nicht bereits wenn ein anderes Feld innerhalb des Wizards angewählt wird. Als gewichtigsten Grund für den tiefen Wert ist jedoch das Fehlen von konkreten, hilfreichen Hinweisen zur Fehlerbehebung angegeben worden.

Bei der Selbstbeschreibungsfähigkeit ist ISIMAP ebenfalls verbesserungswürdig. Dies zeigte sich bereits beim Lösen der Übungsaufgaben. Einzig und alleine einer der 13 Untersuchungsteilnehmer war in der Lage, die Aufgaben ohne Nachschlagen im Tutorial zu lösen. Ein

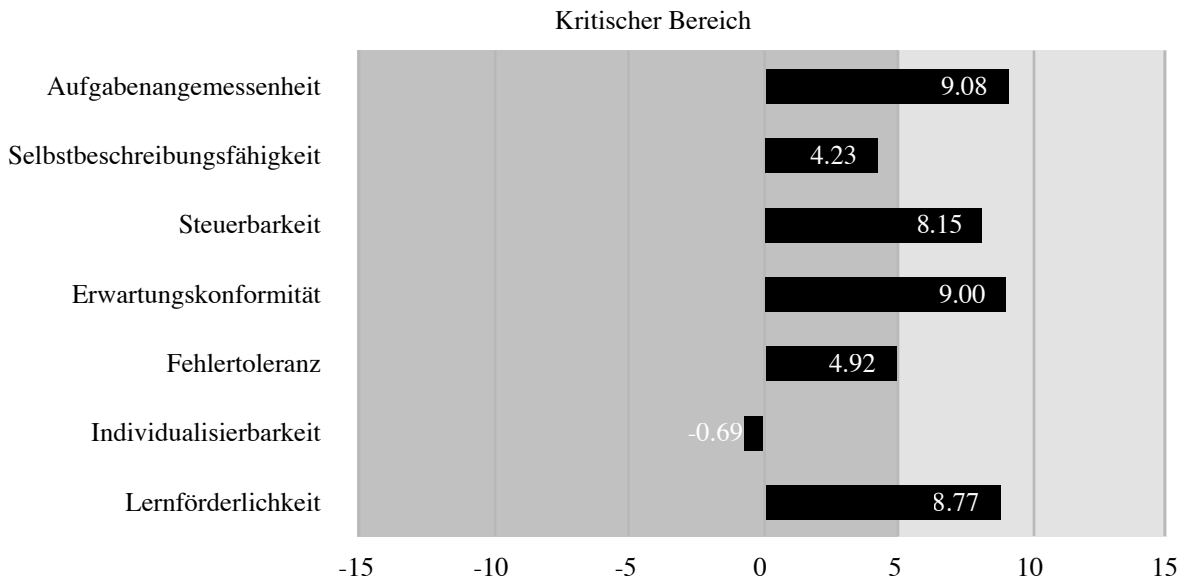

Abbildung 5.3: Benutzerfreundlichkeit von ISIMAP

kritischer Punkt ist sicher die in ISIMAP verwendete Sprache. Die Anweisungen und Beschriftungen der Eingabefelder sind englisch. Dies führte bei den Testpersonen, die ausnahmslos Deutsch als Muttersprache haben, zum Teil zu Verständnisschwierigkeiten. Ausserdem wurde bemängelt, dass die Texte nicht immer klar genug formuliert seien, um zu verstehen, was eingegeben werden soll. Teilweise kann man diese Aussage auf das geringe Wissen über SVG einiger Übungsteilnehmer zurückführen. Durch die Verwendung von verständlicheren Formulierungen könnte dieser Mangel aber vermindert oder behoben werden. Zusammenfassend kann man sagen, dass die Texte in ISIMAP überarbeitet werden müssen.

Sehr schlecht abgeschnitten hat ISIMAP beim Kriterium Individualisierbarkeit. Da es sich bei diesem Programm jedoch um einen Prototypen handelt, bei dem das Schwergewicht auf der Funktionalität liegt, ist dieser Punkt vernachlässigbar. Ebenso sahen dies auch einige der Befragten. Sie fanden eine Individualisierbarkeit sei für ein hilfreiches Programm nicht zwingend notwendig, entscheidend seien die Umsetzung der benötigten Funktionalität sowie eine geringe Fehleranfälligkeit.

Aus dieser Bewertung kann geschlossen werden, dass die Benutzerfreundlichkeit zwar nicht perfekt ist, jedoch ausreicht für eine gelegentliche Anwendung des Programms. Für den Entscheid zum Einsatz sind andere Faktoren bedeutender. In erster Linie muss die implementierte Funktionalität den Benutzer bei seiner Arbeit hilfreich unterstützen. ISIMAP ist konzipiert für die Erzeugung von JavaScript-Interaktionen für SVG-Karten. Wenden wir uns deshalb der Beurteilung der zur Verfügung stehenden Interaktionen zu.

5.3.3 Erzeugung der Interaktionen

Mit der Auswahl der erstellbaren Interaktionen waren alle Befragten zufrieden. Die vorherrschende Meinung war, dass alle grundlegenden Interaktionen erstellbar seien. ISIMAP eigne

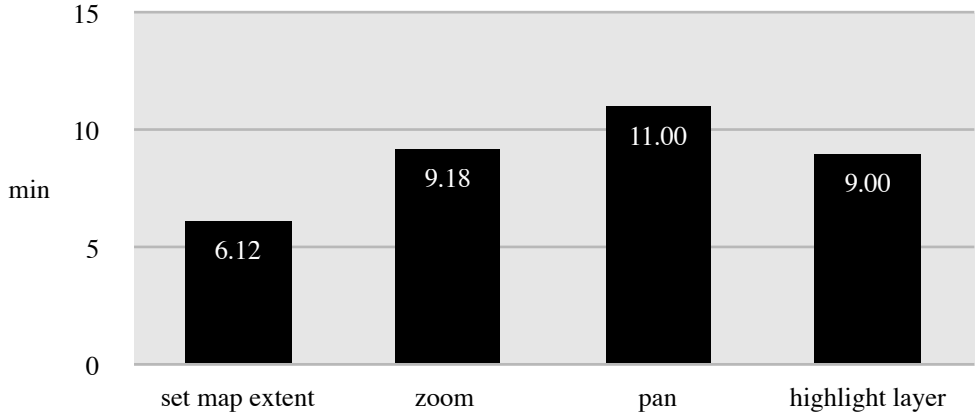

Abbildung 5.4: Durchschnittlich benötigte Zeit zur Erzeugung von Interaktionen

sich ausgezeichnet, um auf eine einfache Art und Weise Karten, die ins Internet gestellt werden sollen, mit den wichtigsten Interaktionen zu versehen. In den mündlichen Bemerkungen wurden drei fehlende Interaktionen mehrmals erwähnt: Informationsanzeige beim Überfahren eines Kartenelementes, ein dynamischer Massstab und die Möglichkeit zum Verschieben des Kartenbildes mit gedrückter Maustaste. Das Weglassen der ersten beiden Interaktionen liegt in der mangelhaften SVG-Export-Funktion von Adobe *Illustrator* begründet (siehe Abschnitt 4.3.1). Die dritte Interaktion wurde bei der Aufnahme der Anforderungsspezifikation nicht erwähnt und bei der Implementierung deshalb nicht berücksichtigt. Für den Einsatz in komplexen und speziellen Applikationen ist die Funktionalität von ISIMAP allerdings zu bescheiden. Die erfahrenen Kartographen und Kartographinnen meinten, eventuell liesse sich durch den Einsatz von ISIMAP eine geeignete Ausgangslage für die Implementierung weiterer Interaktionen erzeugen. Da jedoch niemand den erstellten JavaScript-Code begutachtete, wollte sich bei dieser Aussage niemand definitiv festlegen.

Aufschlussreich ist die Betrachtung der benötigten Zeiten zur Erzeugung einer Interaktion durch die Testpersonen (Abbildung 5.4). Der Eintrag *set map extent* ist keine eigentliche Interaktion, sondern die Eingabe des Kartenbereiches. Dieser muss vorgängig zur *zoom*- und zur *pan*-Interaktion angegeben werden. Einige der Personen gaben die benötigten Angaben einer Interaktion fehlerhaft ein und mussten die Übung noch einmal lösen. Die gestoppten Zeiten beinhalten die Spanne vom Beginn des ersten Versuches zum Anfügen einer Interaktion bis zum Abschluss der letzten und erfolgreichen Eingabe der Parameter. Wenn man bedenkt, dass alle Untersuchungsteilnehmer zum ersten Mal mit dem Programm arbeiteten, kann man sicher behaupten, dass die Zeiten bei einer zweiten Anwendung kürzer werden sollten. Im Schnitt ist für die Erzeugung nicht länger als elf Minuten benötigt worden. Die benötigten Zeiten sind im Vergleich zur manuellen Programmierung als sehr schnell zu betrachten. Auch ein geübter Spezialist benötigt für die Programmierung einer einfachen JavaScript-Interaktion, wie sie mit ISIMAP erstellt werden können, ungefähr gleich lange. Bei einem Laien kann dies gut und gerne einige Stunden in Anspruch nehmen. Bemerkenswert ist auch, dass die Hälfte der Befragten angaben, nicht in der Lage zu sein, solche Interaktionen alleine zu programmieren. Mit der Hilfe von ISIMAP war die Erstellung von JavaScript-Interaktionen auch für sie möglich.

Abbildung 5.5: Allgemeine Beurteilung von ISIMAP

5.3.4 Zusammenfassung

Sehr erfreulich ist die allgemeine Einschätzung von ISIMAP. Die Frage nach der gesamthaften Beurteilung von ISIMAP konnte ebenfalls mit einer siebenstufigen Skala von unbrauchbar (-3) bis hervorragend (+3) beantwortet werden. Als durchschnittlicher Wert wurde 2 angegeben (Abbildung 5.5). ISIMAP wird also allgemein betrachtet als sehr gut eingestuft. Acht der 13 Untersuchungsteilnehmer könnten sich auch vorstellen, das Werkzeug in Zukunft bei ihrer Arbeit zu Hilfe zu nehmen. Zwei Personen gaben an, dass das Programm zwar gelungen sei, sie aber keinen Verwendungszweck bei ihrer Arbeit dafür hätten. Die beiden Personen mit den grössten Erfahrungen in der Erstellung von interaktiven SVG-Karten werden auch keinen Gebrauch von ISIMAP machen. Ihre Projekte seien in der Regel zu komplex und für die Erstellung von grundlegenden Interaktionen hätten sie eigene Code-Fragmente, welche sie per Copy & Paste in den JavaScript-Code einsetzen und anpassen würden.

Die Ergebnisse der Evaluation zeigen die positiven und negativen Eigenschaften von ISIMAP auf. Für einen Einsatz zur Erzeugung von grundlegenden JavaScript-Interaktionen für SVG-Karten eignet sich das Programm sehr gut. In der Benutzerbefragung hat ISIMAP einen überzeugenden Eindruck auf die Untersuchungsteilnehmer hinterlassen. Dies liegt unter anderem sicher in der konsequenten Umsetzung der zu Beginn der Realisierung ermittelten Spezifikationen. Zwar ist die Benutzerfreundlichkeit noch verbesserungswürdig, die erstellbaren Interaktionen sind scheinbar jedoch gut gewählt und der Umfang der Funktionalität von ISIMAP ist für einen erfolgreichen Einsatz ausreichend. Im momentanen Entwicklungsstand kann ISIMAP sicher nicht kommerziell vertrieben werden, dies war aber auch nie beabsichtigt. Es kann aber Kartographen einen einfachen Einstieg in die interaktive Web-Kartographie bieten und sie bei der Herstellung von interaktiven SVG-Karten effizient unterstützen.

6 Schlussfolgerungen und Ausblick

6.1 Schlussfolgerungen

Ausgehend von den Schwierigkeiten bei der Erstellung von JavaScript-Interaktionen für SVG-Karten ist in dieser Arbeit versucht worden, diesen Vorgang mit Hilfe eines geeigneten Werkzeuges zu vereinfachen und zu beschleunigen. Dazu ist ISIMAP erstellt worden, ein Computerprogramm, das die Produktivität dieses Vorgangs steigern soll und programmiertechnisch ungeübte Kartographen und Kartographinnen ermuntern soll, JavaScript-Interaktionen in ihre Web-Karten einzubauen. In diesem Kapitel wird diskutiert, inwiefern die angepeilten Ziele erreicht worden sind und welche Schussfolgerungen aus den Resultaten gezogen werden können.

6.1.1 Konzentration auf die notwendige Funktionalität

In Zusammenarbeit mit Personen aus dem kartographischen Bereich sind Anforderungen an ein Werkzeug für die automatische Erzeugung von JavaScript-Interaktionen für SVG-Karten ermittelt worden. Dabei hat sich bestätigt, dass ein **grosses Bedürfnis** nach einem solchen Hilfsmittel besteht. Die Bereitschaft für die Mitarbeit bei der Anforderungsermittlung und bei der Evaluation des implementierten Programms war dementsprechend gross. Dank diesem Einsatz konnte eine umfassende Anforderungsspezifikation für das Werkzeug erstellt werden.

Die zentralen **Eigenschaften eines geeigneten Werkzeuges** müssen, neben der einfachen und intuitiven Bedienung, die Unterstützung von Adobe *Illustrator*-Karten, die Erzeugung der grundlegenden Interaktionen und die Einstellung von Parametern der Interaktionen durch die Benutzenden sein. Dabei sollen die Anwendenden möglichst keinen Kontakt mit dem eigentlichen JavaScript-Code haben. Die ermittelten Anforderungen können auf verschiedene Art und Weise umgesetzt werden. Der in dieser Arbeit gewählte Ansatz, ein Computerprogramm mit einer graphischen Oberfläche, scheint in eine erfolgsversprechende Richtung zu weisen. Dies ergab die abschliessende Evaluation.

Die Zufriedenheit der Anwendenden von ISIMAP kann nur durch eine sinnvolle Auswahl von erstellbaren Interaktionen erreicht werden. Die Begutachtung von interaktiven SVG-Karten im Internet sowie Befragungen von Web-Kartographen und -Kartographinnen ermöglichte die Zusammenstellung der **grundlegenden Interaktionen**. Besonders oft eingesetzt werden Funktionen zur Ausgleichung von bildschirmspezifischen Defiziten. In diese Gruppe gehören Interaktionen zur Vergrösserung und zur Verkleinerung des Kartenbildes sowie zur Navigation im angezeigten Kartenausschnitt. Des weiteren kommen Interaktionen zur verbesserten graphischen Sichtbarkeit von Kartenelementen durch Hervorhebungen und zur Anzeige von ergänzenden Informationen eine grosse Bedeutung zu. Die Berücksichtigung der erwähnten

Interaktionen bei der Umsetzung ist eine wesentliche Voraussetzung für einen praktischen Einsatz des Werkzeuges.

6.1.2 Benutzungsgerechte Umsetzung

Auf der Grundlage einer Anforderungsspezifikation konnte das geforderte Werkzeug benutzergerecht implementiert werden. Aus der Realisierung können drei wesentliche Folgerungen gezogen werden. Erstens ermöglicht die Einteilung der Anforderungen in **Prioritätenklassen** die Konzentration auf die relevanten Punkte eines erfolgreichen Projektes. Durch ein schrittweises Vorgehen konnte effizient gearbeitet und der Zeitrahmen somit leichter eingehalten werden. Für die Bildung von Prioritätenstufen bewährt sich die MoSCoW-Methode. Dies zeigt sich bei der Überprüfung der Umsetzung der Anforderungsspezifikation. Fast alle Punkte der drei wichtigsten Prioritätenklassen konnten realisiert werden.

Zweitens kann gesagt werden, dass sich die Programmiersprache **Java** für die Realisierung der geforderten Funktionalität eignet, insbesondere durch die Portabilität und die umfangreiche Klassenbibliothek. Für die Manipulation von SVG-Dokumenten verfügt die Java-Bibliothek Batik über umfangreiche Möglichkeiten und kann hilfreich eingesetzt werden.

Als dritter Punkt seien die Schwierigkeiten bei der Programmierung von abgekapselten und **unabhängigen Musterskripts** erwähnt. Die Beziehungen zwischen verschiedenen Interaktionen sind in vielen Fällen nicht einfach zu entflechten. Es besteht ein umgekehrt proportionaler Zusammenhang zwischen dem Grad der Eigenständigkeit der Musterskripts und der Komplexität des Algorithmus für die Zusammenstellung der benötigten Musterskripts. Ausserdem müssen wegen der ungenügenden SVG-Exportfunktion in Adobe *Illustrator* zum Teil umständliche Routinen in die Musterskripts eingebaut werden.

ISIMAP ermöglicht also die automatische Erzeugung von JavaScript-Interaktionen für SVG-Karten. Es stellten sich die Fragen, ob sich das Programm erfolgreich einsetzen lässt und ob es eine Erleichterung für die Kartenautoren und -autorinnen mit sich bringt. Diesen Fragen wurde in einer **Evaluation** nachgegangen worden. Die Durchführung einer Benutzerbefragung unter der Verwendung des Fragebogens zur ISO-Norm 9241/10 zusammen mit mündlichen Interviews mit Personen des kartographischen Fachbereiches ermöglichten eine fundierte Einschätzung des realisierten Werkzeuges.

Die Beurteilung der **Benutzerfreundlichkeit** von ISIMAP fällt nicht ganz so positiv aus. Insbesondere die Aspekte der Individualisierbarkeit, der Selbstbeschreibungsfähigkeit und der Fehlertoleranz schneiden ungenügend ab. Die in ISIMAP verwendeten, englischen Begriffe und Anweisungen sind zum Teil schwer verständlich. Mit Hilfe des gelungenen, deutschen Tutorials können aber alle Eingabedialoge mühelos beantwortet werden. Kritisiert wurden auch die Fehlermeldungen. Diese werden zu spät angezeigt und bieten keine konkreten Angaben zur Fehlerbehebung. Immerhin werden aber fehlerhafte Eingaben der Benutzenden abgefangen und angezeigt. Auf der anderen Seite können aber auch positive Eigenschaften der Benutzerfreundlichkeit von ISIMAP festgehalten werden. Durch die übersichtliche Anordnung des Layouts und der Menübefehle ist schnell erkennbar, was mit dem Programm möglich ist. Ausserdem haben die Benutzertests gezeigt, dass die Handhabung der Funktionalität in sehr kurzer Zeit erlernt werden kann.

6.1.3 Vorteilhafter Einsatz von ISIMAP

ISIMAP wird generell als sehr gut beurteilt. Insbesondere die Auswahl der erstellbaren Interaktionen wurde als umfassend genug für die Generierung der Grundfunktionalität von interaktiven Karten bewertet. Diese Interaktionen steigern die Qualität von Karten im Internet ungemein und werden bereits heute in vielen Projekten eingesetzt. Die komplizierte und zeitaufwändige Programmierung hat bisher jedoch viele Kartenautoren und -autorinnen von der Verwendung von Interaktionen abgeschreckt. ISIMAP bringt eine **Vereinfachung bei der Herstellung von Interaktionen** mit sich. Diese können nun mit einigen Mausklicks erstellt werden. Es verfügt dank den Funktionen zum Vergrössern, Verkleinern und Verschieben des angezeigten SVG-Dokumentes sowie einer *undo*-Funktion über Hilfsmittel, die den Anwendenden die Erstellung der Interaktionen erleichtern. Es hat sich gezeigt, dass mit Hilfe von ISIMAP auch Programmierunkundige ihre Karte ohne grosses Vorwissen für die Veröffentlichung im Internet optimieren können.

Die automatische Generierung des JavaScript-Codes bringt eine weitere Verbesserung mit sich. Da alle Musterskripts ausgiebig getestet sind, ist davon auszugehen, dass sich die **Fehleranfälligkeit vermindert**. Damit fällt die mühsame und nervenaufreibende Suche nach Fehlern in selber erstellten Interaktionen weg.

Für Anwendende mit grundlegenden Kenntnissen in der Programmierung von JavaScript-Interaktionen bringt ISIMAP eine gewaltige **Zeiteinsparung** mit sich. In Tests wurde als durchschnittliche Zeit für die Erzeugung einer Interaktion mit ISIMAP ein Wert von ungefähr zehn Minuten ermittelt. Bei der manuellen Implementierung kann dieser Vorgang für ungeübte Programmierer Stunden, wenn nicht sogar Tage, dauern. ISIMAP ist also ein effektives und effizientes Werkzeug zur Erzeugung von JavaScript-Interaktionen für SVG-Karten.

Vielen Evaluationsteilnehmenden gefällt das Programm ausgezeichnet. Sie forderten eine Kopie von ISIMAP an, um es bei Gelegenheit einsetzen zu können. Dies obwohl viele von ihnen noch nie SVG-Karten, geschweige denn interaktive, erstellt haben. Das vorgestellte Werkzeug weckte resp. **steigerte das Interesse an interaktiven SVG-Karten** bei mehreren Personen und trug erfolgreich zum Abbau von psychologischen Hemmschwellen bei. Die programmierunkundigen Probanden sehen jetzt eine Möglichkeit, Interaktionen zu erstellen, ohne vorgängig eine Skriptsprache mühsam erlernen zu müssen.

6.1.4 Zusammenfassung

Zusammenfassend kann gesagt werden, dass die gesteckten Ziele der Arbeit zu einem grossen Teil erreicht worden sind. Die Implementierung eines geeigneten Werkzeuges zur automatischen Erzeugung von JavaScript-Interaktionen für SVG-Karten ist erfolgreich realisiert worden. Eine Vereinfachung und eine Steigerung der Produktivität gegenüber der manuellen Programmierung von Interaktionen kann klar festgehalten werden und die Verwendung von Interaktionen in SVG-Karten kann durch die Bekanntmachung von ISIMAP angeregt werden.

Die vorliegende Arbeit bietet einen möglichen Lösungsansatz für eine der aktuellen Schwierigkeiten der Web-Kartographie, der relativ komplexen Implementierung der grundlegenden Funktionalität von interaktiven Karten im Internet. Mit ISIMAP ist es erstmals möglich, solche Interaktionen ohne grosse Vorkenntnisse zu generieren. Die Forderung von Räber und Jenny

[Räber und Jenny, 2003] nach einer Möglichkeit, interaktive Karten zu erstellen, ohne den Quellcode editieren oder Routinen programmieren zu müssen, ist somit erfüllt. Da nun ein handliches Werkzeug vorhanden ist, ist anzunehmen, dass in Zukunft vermehrt Interaktivität in SVG-Karten eingebaut wird.

Es bietet sich also an, den präsentierten Ansatz durch allfällige Verbesserungen und geeignete Erweiterungen oder zumindest die Idee der automatischen Erzeugung von JavaScript-Interaktionen für SVG-Karten weiterzuverfolgen.

6.2 Ausblick

6.2.1 Weiterentwicklung von ISIMAP

Die Entwicklung von ISIMAP ist im Rahmen dieser Arbeit abgeschlossen. Das Programm ist allerdings ein Prototyp und kein ausgereiftes, kommerziell einsetzbares Produkt. Es bietet mehrere Ansatzpunkte für Verbesserungen und für Erweiterungen. Viele der nachfolgenden Vorschläge sind aus der Evaluation abgeleitet und können als Grundlage für weiterführende Arbeiten dienen.

ISIMAP sollte noch in mehreren Bereichen überarbeitet werden. Es sind noch einige **kleine Fehler im Quellecode** vorhanden, welche ausgemerzt werden sollten. Ausserdem ist die Anforderungsspezifikation nicht vollständig implementiert. Insbesondere eine Funktion, welche es erlaubt, beliebige angefügte Interaktionen wieder zu entfernen, brächte einen weiteren Komfort für die Benutzenden von ISIMAP.

In der Evaluation hat sich gezeigt, dass die ermittelte Anforderungsspezifikation nicht alle relevanten Eigenschaften enthält. **Weitere Funktionen** sind für ein optimales Werkzeug notwendig. Mehrere Versuchspersonen sprachen Erweiterungsmöglichkeiten an. So wäre das Aufrufen von bereits erstellten Interaktionen und die nachträgliche Veränderung der verwendeten Parameter sehr hilfreich. Wünschenswert ist auch eine Möglichkeit zur Erstellung von graphischen Hilfselementen, wie zum Beispiel dem Rechteck des angezeigten Kartenbereiches in der Übersichtskarte. So müssten diese nicht bereits in Adobe *Illustrator* gezeichnet werden.

Einen enormen Mehrwert brächte auch eine Erweiterung zur **Eingabe von eigenen Interaktionen**. So könnten die Auswahl der Interaktionen auch von Personen ergänzt werden, welche die Erstellung von JavaScript-Interaktionen gut beherrschen und eigene Interaktionen für andere Web-Kartographen und -Kartographinnen bereitstellen wollen. Eine solche Erweiterung der aktuellen Version von ISIMAP ist nur schwer möglich, ein umfassendes Redesign der Architektur müsste durchgeführt werden. Es müssen ja nicht nur die JavaScript-Funktionen erstellt werden können, sondern die Benutzenden von ISIMAP müssten auch die variablen Parameter einstellen können. Bei der Eingabe von eigenen Interaktionen müsste also gleichzeitig eine Möglichkeit zur Definition des JavaScript-Codes, der Parameter, der Beschreibungen der Eingabefelder, der Beschränkungen von Eingabewerten usw. vorhanden sein. In der vorliegenden Version werden diese Angaben mit Java-Methoden umgesetzt und können nur durch eine Änderung des Quellecodes angepasst werden. Eine weitere Voraussetzung für die Eingabe von eigenen Musterskripts wäre auch die Verwendung von eigenständigen Musterskripts

ohne Bezug zu anderen Musterskripts. Dies ist in dieser Arbeit noch nicht ganz gelungen.

Ein weiteres zu überdenkendes Konzept ist die **Angabe der SVG-Elemente** in Wizards durch die Benutzenden. Die geschieht wie gesehen über die `id` der Elemente. Um übergeordnete Elemente anzusprechen, werden Auswahllisten mit einer Abbildung der DOM-Struktur angezeigt. Die Bedeutung dieser Angaben wird von SVG-Laien nicht immer auf den ersten Blick verstanden. Hier sollte vielleicht mit anderen (graphischen) Konzepten gearbeitet werden. In diesem Bereich ist auch ein Mangel in der Implementierung vorhanden. SVG-Elemente ohne `id` können nicht ausgewählt werden. Hier sollte die Möglichkeit zur Angabe einer `id` beim Anklicken eines solchen SVG-Elementes geboten werden.

Ein in der Evaluation teilweise kritisierter Punkt ist die **Benutzerfreundlichkeit** von ISIMAP. Im Zentrum stehen zwei Problembereiche. Zum einen müssen Fehlermeldungen gezielter angezeigt werden und Hinweise zur Fehlerbehebung liefern. Zum anderen sind die verwendeten Begriffe und Anweisungen zu überarbeiten. Zu überlegen ist dabei, ob die verwendete Sprache von englisch auf deutsch gewechselt werden soll. Bei einem Einsatz in einem internationalen Umfeld ist dies aber sicher nicht sinnvoll. Als Alternative ist die Entwicklung mehrerer Sprachversionen von ISIMAP zu überlegen. Hilfreich für die Benutzenden wäre sicher auch die Integration einer umfangreichen Hilfefunktion. Neben den Beschreibungen der einzelnen Funktionen von ISIMAP könnte diese auch eine bildliche Darstellung der einzelnen auswählbaren Interaktionen beinhalten. Als Ausgangslage dazu könnte dabei das Tutorial dienen.

Ein Stärke von ISIMAP ist die Portabilität. Diese Eigenschaft beruht auf dem Einsatz von Java als Programmiersprache. Eine andere Möglichkeit dafür wäre die Implementierung einer **Online-Version**. Bereits mit kleinen technischen Veränderungen könnte man ISIMAP in ein Applet umwandeln. Hier stellt sich allerdings die Frage, ob die Grösse von Batik (3.8 MB) nicht zu einer für die Benutzenden unzumutbaren Übertragungszeit führen würde. Ein anderer Ansatz wäre eine reine JavaScript-Lösung. Peto [29] bietet auf einer Webseite einen SVG Editor an, der nach diesem Prinzip aufgebaut ist. Ähnlich könnte auch ein Werkzeug für die automatische Erzeugung von JavaScript-Interaktionen für SVG-Karten realisiert werden.

Für die **optimale Einbindung in den kartographischen Herstellungsprozess**, wäre die Implementierung der in ISIMAP verfügbaren Funktionalität in anderen, oft eingesetzte Systeme denkbar. Eine Möglichkeit wäre unter anderem die Erstellung einer Erweiterung, ein Plugin, für Adobe *Illustrator*. Adobe *Illustrator* wird im kartographischen Bereich oft eingesetzt, um Karten zu zeichnen und verfügt über eine SVG-Exportfunktion. Durch die Integration der Funktionalität in dieses Programm stünde den Kartenautoren und -autorinnen ein Hilfsmittel zu Verfügung, das über alle für die Erstellung von interaktiven SVG-Karten notwendigen Möglichkeiten verfügt.

6.2.2 Vereinfachungen in der Web-Kartographie

Es ist wohl unbestritten, dass auch in Zukunft die Veröffentlichung von Karten im Internet zunehmen wird. Mit SVG und JavaScript stehen den Kartenautoren und -autorinnen effektive Techniken für die Gestaltung von benutzerfreundlichen, graphisch ansprechenden und interaktiven Karten zu Verfügung. Der Aufwand zur Erstellung solcher Karten ist jedoch mit Schwierigkeiten und einem hohen Aufwand verbunden. Es sollten deshalb **weitere Hilfsmittel** für die

Vereinfachung und für die Beschleunigung dieses Prozesses entwickelt werden. Dies könnte Werkzeuge für unerfahrene Kartenersteller und -erstellerinnen, wie zum Beispiel einen für die Kartographie zugeschnittenen SVG-Editor, oder für routinierte Entwickler und Entwicklerinnen von interaktiven SVG-Karten, wie zum Beispiel einen ausgereiften JavaScript-Debugger, umfassen. Nur so kann sichergestellt werden, dass diese Techniken eine breite Anwendung finden und dass die Qualität von Karten im Internet zunehmend besser wird.

Die angestrebte Überwindung der aktuellen Schwierigkeiten in der Web-Kartographie kann aber nur eintreten, sofern die Kartenautoren und -autorinnen solche Hilfsmittel überhaupt einsetzten. Eine Voraussetzung dafür ist, dass die kartographischen Fachleute über die angebotene Palette an Hilfsmittel informiert sind. Aus dieser Feststellung ergeben sich zwei Aufgaben. Erstens müssen die Entwickler von Software ihre Produkte bekannt machen. Neben einer benutzergerechten Implementierung trägt auch die gezielte **Vermarktung** zum Erfolg eines Werkzeuges bei. Dies gilt auch für ISIMAP. Für einen gelungenen Abschluss dieses Projektes besteht der nächste Schritt also in der Bekanntmachung des Programms im kartographischen Sektor. Zweitens wäre eine aktuelle **Zusammenstellung über die angebotenen Werkzeuge** hilfreich für die Kartenautoren und -autorinnen. Diese Auflistung könnte eine Beschreibung der Funktionalität und den Einsatzmöglichkeiten sowie eine Beurteilung der einzelnen Werkzeuge enthalten.

Bei aller Begeisterung im kartographischen Bereich für SVG sollte jedoch nicht vergessen werden, dass die Informatik ein sehr kurzlebiges Geschäft ist. Was heute noch als state-of-the-art gilt, ist schon morgen Schnee von gestern. **Neue Techniken** bringen immer auch neue Möglichkeiten und neue Chancen mit sich. Web-Kartographen und Kartographinnen sollten deshalb immer ein Auge auf weitere Entwicklungen werfen und versuchen die Eignung von anderen Techniken für die Web-Kartographie abzuschätzen. Vielleicht zeigt sich dabei, dass zur Lösung der aktuellen Schwierigkeiten in der Web-Kartographie ein ganz anderer Weg eingeschlagen werden muss.

7 Literaturverzeichnis

Literatur

[Baggen, 2003] Baggen, R. (2003). Normen und Zertifizierung. In Heinsen, S. und Vogt, P. (Eds.), *Usability praktisch umsetzen*. Hanser, München.

[Berger, 2003] Berger, A. (2003). Scalable Vector Graphics - Gestalltung einer interaktiven Karte. In Wilfert, I. (Ed.), *Scalable Vector Graphics zur Internetpräsentation von Karten*. Kartographische Bausteine, Institut für Kartographie, Dresden.

[Brunner, 2001] Brunner, K. (2001). Kartengraphik am Bildschirm - Einschränkungen und Probleme. *Kartographische Nachrichten*, Nr. 5, 233–240.

[Burmester et al., 1997] Burmester, M., Görner, C., Vossen, P. H., Zolleis, T. M., und Zouboulidis, V. (1997). Qualitatives Software Screening unter der Verwendung des Fragebogens ISONORM 9241/10 von Joachim Prümper und Michael Anft. In für Arbeitsschutz und Arbeitsmedizin, B. (Ed.), *Das SANUS Handbuch - Bildschirmarbeit EU-konform*. Wirtschaftsverlag NW, Dortmund.

[Cartwright, 2003] Cartwright, W. (2003). Maps on the Web. In Peterson, M. P. (Ed.), *Maps and the Internet*. Elsevier Science B. V., Oxford.

[Cecconi et al., 2000] Cecconi, A., Shenton, C., und Weibel, R. (2000). Verwendung von Java für die kartographische Visualisierung von statistischen Daten auf dem Internet. *Kartographische Nachrichten*, Nr. 4, 151–162.

[Dahinden et al., 2003] Dahinden, T., Neumann, A., und Winter, A. M. (2003). Internet-Kartengraphik mit SVG-Werkzeugen - Techniken und Anwendungen. In Asche, H. und Herrmann, C. (Eds.), *Web.Mapping 2*. Wichmann, Heidelberg.

[Dickmann, 1997] Dickmann, F. (1997). Kartographie und Internet. *Kartographische Nachrichten*, Nr. 3, 87–96.

[Dickmann, 1999] Dickmann, F. (1999). ”Web Mapping” and ”Online GIS” - die Entwicklung kartographischer und geographischer Informationssysteme im Internet. *Petermanns Geographische Mitteilungen*, Nr. 5+6, 465–473.

[Dickmann, 2000a] Dickmann, F. (2000a). Webmapping (Teil I): Kartenentwurf mit dem World Wide Web. *Geographische Rundschau*, Nr. 3, 42–47.

[Dickmann, 2001] Dickmann, F. (2001). Vektorgrafik vor dem Durchbruch - XML-basierte 2D-Vektorformate visualisieren Geodaten im WWW. *Kartographische Nachrichten*, Nr. 6, 286–291.

[Dickmann, 2002] Dickmann, F. (2002). Interaktionserweiterung von Web-Karten mit Hilfe von Skriptsprachen - Das Beispiel JavaScript. *Kartographische Nachrichten*, Nr. 1, 19–26.

[Dickmann, 2004] Dickmann, F. (2004). Mehr Schein als Sein? - Die Wahrnehmung kartengestützter Rauminformationen aus dem Internet. *Kartographische Nachrichten*, Nr. 2, 61–67.

[Eisenberg, 2002] Eisenberg, J. D. (2002). *SVG Essentials*. O'Reilly, Beijing.

[European Computer Manufacturer's Assocation, 1999] European Computer Manufacturer's Assocation (1999). *ECMAScript Language Specification*. ECMA, Genf.

[Fibinger, 2001] Fibinger, I. (2001). Scalable Vector Graphics (SVG) - Untersuchung des XML-Standards für zweidimensionale Vektorgraphiken und Erstellung eines SVG-Tutorials. Diplomarbeit, Fachhochschule Karlsruhe.

[Flanagan, 1996] Flanagan, D. (1996). *Java in a Nutshell*. O'Reilly, Köln.

[Gartner, 2003] Gartner, G. (2003). Geovisualisierung und Kartenpräsentation im Internet - Stand und Entwicklung. In Asche, H. und Herrmann, C. (Eds.), *Web.Mapping 2*. Wichmann, Heidelberg.

[Glinz, 2003] Glinz, M. (2003). *Software Engineering I*. Vorlesungsskript, Institut für Informatik, Universität Zürich, Zürich.

[Hake und Grünreich, 2002] Hake, G. und Grünreich, D. (2002). *Kartographie*. de Gruyter, 8. Auflage.

[Hamborg et al., 2003] Hamborg, K.-C., Gediga, G., und Hassenzahl, M. (2003). Fragebogen zur evaluation. In Heinsen, S. und Vogt, P. (Eds.), *Usability praktisch umsetzen*. Hanser, München.

[Hurni et al., 2001] Hurni, L., Neumann, A., und Winter, A. M. (2001). Aktuelle Webtechniken und deren Anwendung in der thematischen Kartographie und der Hochgebirgskartographie. *Beiträge zum 50. Deutschen Kartographentag*, 127–147.

[Joray, 2001] Joray, C. (2001). Migration zwischen der Schweiz und Europa von 1981 bis 1999. Diplomarbeit, Institut für Kartographie, ETH Zürich.

[Karat, 1994] Karat, C.-M. (1994). A Comarision of User Interface Evaluation Methods. In Nielsen, J. und Mack, R. L. (Eds.), *Usability Inspection Methods*. Usability Inspection Methods.

[Kellenberger, 2005] Kellenberger, K. (2005). Aspekte zur umsetzung des schweizer weltatlas aus einer gedruckten karte in eine multimediale version. Diplomarbeit, Geographisches Institut der Universität Zürich.

[Kraak, 2001] Kraak, M.-J. (2001). Settings and needs for web cartography. In Kraak, M.-J. und Brown, A. (Eds.), *Web Cartography*. Taylor & Francis, London and New York.

[Lorenzen-Schmidt, 2003] Lorenzen-Schmidt, O. (2003). Testpersonen rekrutieren. In Heinsen, S. und Vogt, P. (Eds.), *Usability praktisch umsetzen*. Hanser, München.

[Mack und Nielsen, 1994] Mack, R. L. und Nielsen, J. (1994). Executive Summary. In Nielsen, J. und Mack, R. L. (Eds.), *Usability Inspection Methods*. Usability Inspection Methods.

[Müller-Prove, 2003] Müller-Prove, M. (2003). Usability in Unternehmen. In Heinsen, S. und Vogt, P. (Eds.), *Usability praktisch umsetzen - Handbuch für Software, Web, Mobile Devices und andere interaktive Produkte*. Hanser, München.

[Neumann und Winter, 2001] Neumann, A. und Winter, A. M. (2001). Time for SVG - Towards High-Quality Interactive Web-Maps. In *Proceedings of the 20th Internationals Cartographic Congress*, Beijing. 9 Seiten.

[Neumann und Winter, 2003a] Neumann, A. und Winter, A. M. (2003a). Webkartographie mit Verktorgraphik - Formate, Standards und Beispiele mit SVG. In Asche, H. und Herrmann, C. (Eds.), *Web.Mapping 2*. Wichmann, Heidelberg.

[Neumann und Winter, 2003b] Neumann, A. und Winter, A. M. (2003b). Webmapping with Scalable Vector Graphics. In Peterson, M. P. (Ed.), *Maps and the Internet*. Elsevier Science B. V., Oxford.

[Nielsen, 1993] Nielsen, J. (1993). *Usability Engineering*. Academic Press, San Diego.

[Peterson, 1999] Peterson, M. (1999). Trends in Internet Map Use - A Second Look. In *Proceedings 19th ICA/ACI International Cartographic Conference*, Ottawa. 25-34.

[Peterson, 2003] Peterson, M. P. (2003). Maps and the Internet: An Introduction. In Peterson, M. P. (Ed.), *Maps and the Internet*. Elsevier Science B. V., Oxford.

[Plewe, 1997] Plewe, B. (1997). *GIS Online - Information Retrieval, Mapping and the Internet*. OnWorld Press, Santa Fe.

[Räber und Jenny, 2003] Räber, S. und Jenny, B. (2003). Karten im Netz - ein Plädoyer für mediengerechte Kartengraphik. In Asche, H. und Herrmann, C. (Eds.), *Web.Mapping 2*. Wichmann, Heidelberg.

[Shenton, 2000] Shenton, C. (2000). Kartografische Visualisierung multivariater, statistischer Daten in einer verteilten objekt-orientierten Umgebung. Diplomarbeit, Geographisches Institut der Universität Zürich.

[Sieber, 2002] Sieber, M. (2002). A Digital Atlas of the Sri Lankan Central Province. Diplomarbeit, Geographisches Institut der Universität Zürich.

[Überschär et al., 2003] Überschär, N., Held, G., Neumann, A., und Winter, A. M. (2003). SVG für die Webkartographie - Aktuelles und Zukünftiges. In *Proceedings of the Webmapping 2003 Forum*, Potsdam. 15 Seiten.

[Ullrich, 2003] Ullrich, T. (2003). Dynamische Online-Visualisierung thematischer Karten im Internet. Diplomarbeit, Fachhochschule Karlsruhe.

[van den Worm, 2001] van den Worm, J. (2001). Web map design in practice. In Kraak, M.-J. und Brown, A. (Eds.), *Web Cartography*. Taylor & Francis, London and New York.

[van Elzakker, 2001a] van Elzakker, C. P. J. M. (2001a). Use of maps on the Web. In Kraak, M.-J. und Brown, A. (Eds.), *Web Cartography*. Taylor & Francis, London and New York.

[van Elzakker, 2001b] van Elzakker, C. P. J. M. (2001b). Users of maps on the Web. In Kraak, M.-J. und Brown, A. (Eds.), *Web Cartography*. Taylor & Francis, London and New York.

[Vogt, 2003] Vogt, P. (2003). Usability im Entwicklungsprozess. In Heinsen, S. und Vogt, P. (Eds.), *Usability praktisch umsetzen - Handbuch für Software, Web, Mobile Devices und andere interaktive Produkte*. Hanser, München.

[Vogt und Heinsen, 2003] Vogt, P. und Heinsen, S. (2003). Einleitung: Usability - darum geht es. In Heinsen, S. und Vogt, P. (Eds.), *Usability praktisch umsetzen*. Hanser, München.

[Watt und Lilley, 2002] Watt, A. H. und Lilley, C. (2002). *SVG Unleashed*. Sams, Indianapolis.

[Wilfert, 2003] Wilfert, I. (Ed.) (2003). *Scalable Vector Graphics (SVG) zur Internetpräsentation von Karten*. Kartographische Bausteine, Institut für Kartographie, Dresden.

[Winkler, 2000] Winkler, D. (2000). Erstellung eines Java-Applets zur Visualisierung und Analyse von Thematischen Karten. Diplomarbeit, Fachhochschuhle München, Fachbereich Vermessungswesen und Kartographie.

[Winter, 2000] Winter, A. M. (2000). Interetkartographie mit SVG. Diplomarbeit, Fakultät der Human- und Sozialwissenschaften an der Universität Wien.

[Zuser et al., 2004] Zuser, W., Grechenig, T., und Köhle, M. (2004). *Software Engineering - mit UML und dem Unified Process*. Pearson Studium, München.

URLs

[1] Adobe Systems Incorporated. Adobe SVG Viewer Download Area. `http://www.adobe.com/svg/viewer/install/main.html`, Zugriff: December 2003.

[2] (The Mail Archive). batik-users. http://www.mail-archive.com/batik-users@xml.apache.org/, Zugriff: Juni 2004.

[3] Autodesk. MapGuide - Product Information. http://www.mapguide.com, Zugriff: April 2005.

[4] Lothar Bräutigam. Beurteilung der software-ergonomie angand des isonorm-fragebogens. http://www.sozialnetz-hessen.de/ca/pq/mdl/, Zugriff: Dezember 2004.

[5] carto:net. svg links. http://www.carto.net/papers/svg/links/, Zugriff: April 2004.

[6] Coley Consulting. MoSCoW Priorisation. http://www.coleyconsulting.co.uk/moscow.htm, Zugriff: Mai 2004.

[7] Eclipse Foundation. Welcome. http://www.eclipse.org/, Zugriff: Juni 2004.

[8] ESRI. ArcIMS. http://www.esri.com/software/arcgis/arcims/index.html, Zugriff: April 2005.

[9] The Apache Software Foundation. Batik SVG Toolkit. http://xml.apache.org/batik/, Zugriff: Mai 2004.

[10] Gaëtan Gaborit. Département du Sèvre-et-Maine. http://svgmap.free.fr/carte.htm, Zugriff: Mai 2004.

[11] Vincent J. Hardy and Thierry Kormann. Leveraging SVG on the Java platform with Batik. http://www.gca.org/papers/xmleurope2001/papers/html/s21-1.html, Zugriff: April 2004.

[12] Intermaps. intermaps makes maps smarter. www.intermaps.com/2004/de/news_frame.html, Zugriff: Mai 2005.

[13] Yvonne Isakowski. Dynamic Processes of the Gruben Glacier - Switzerland. http://www.carto.net/papers/svg/gruben_glacier/index.svgz, Zugriff: January 2003.

[14] Kathrin Kellenberger. Interaktiver Schweizer Weltatlas - Neuseeland. http://www.schweizerweltatlas.ch/prototyp/NewZealand_SVG/index.html, Zugriff: Mai 2005.

[15] Kitfox. SVG Salamander. https://svgsalamander.dev.java.net/, Zugriff: Juni 2004.

[16] Thierry Kormann. Developing SVG Applikations with Batik. http://www.svgopen.org/2002/papers/kormann__developing_svg_apps_with_batik/, Zugriff: April 2004.

[17] Learn SVG. Scripting the DOM. http://www.learnsvg.com/html/bitmap/chapter10/page10-1.htm, Zugriff: April 2004.

[18] MapQuest. Maps, Directions and More. http://www.mapquest.com, Zugriff: April 2005.

[19] MeteoSchweiz. Wetter und Klima in der Schweiz. http://www.meteoschweiz.ch/de/Daten/Messwerte/IndexMesswerte.shtml, Zugriff: Mai 2005.

[20] Tobias Metzger. Stadtplan Stuttgart. http://www.stuttgart2006.net/svg/karte.htm, Zugriff: November 2004.

[21] Microsoft Corporation. MSN Maps & More. http://www.mapblast.com, Zugriff: April 2005.

[22] Stefan Mintert. JavaScript-Informationen. http://www.mintert.com/javascript/, Zugriff: Mai 2005.

[23] Dept of Transport Engineering Nat. Tech. University of Athens. Athens Traffic Map. http://www.transport.ntua.gr/map/en/index.php, Zugriff: Mai 2005.

[24] Andreas Neumann. Tuerlersee -Interactive Topographic Map. http://www.carto.net/papers/svg/tuerlersee/, Zugriff: Mai 2004.

[25] Andreas Neumann. Comparing .SWF and .SVG file format specifications. www.carto.net/papers/svg/comparison_flash_svg/, Zugriff: April 2005.

[26] Port of Oakland. Replay of Bay Area Air Traffic. http://www.oaklandtracks.com/noise/svgtest.html, Zugriff: January 2005.

[27] University of Texas Libraries. Perry-Castañeda Library Map Collection. http://www.lib.utexas.edu/maps/, Zugriff: April 2005.

[28] Opera Software ASA. Opera Features: SVG. www.opera.com/features/svg/, Mai.

[29] Chris Peto. Svg editor. http://www.resource-solutions.de/svgeditor/main.svg, Zugriff: Februar 2005.

[30] Lonely Planet. Lonely Planet - Destinations. http://www.lonelyplanet.com/destinations/, Mai.

[31] Marco Sieber. Interactive Atlas of the Sri Lankan Central Province. http://www.geoconcept.ch/atlas/atlas.html, Zugriff: Juli 2002.

[32] National Statistics. NS 2001 Area Classification for Local Authorities. http://www.statistics.gov.uk/about/methodology_by_theme/area_classification/svg/index.html, Zugriff: April 2005.

[33] Sun Microsystems. JavaBean Spec. http://java.sun.com/products/javabeans/docs/spec.html, Zugriff: Mai 2004.

[34] Sun Microsystems. The java(tm) language: An overview. http://java.sun.com/docs/overviews/java/java-overview-1.html, Zugriff: April 2004.

[35] Sun Microsystems. The Java Tutorial. http://java.sun.com/docs/books/tutorial/index.html, Zugriff: Juni 2004.

[36] The Mozilla Organisation. Building Mozilla with SVG Support. www.mozilla.org/projects/svg/build.html, Zugriff: Mai 2005.

[37] Unbekannt. AnotherSimpleShapefile Converter. http://www.mycgiserver.com/~amri/converter.cocoon.xml, Zugriff: Juni 2004.

[38] U.S Census Bureau. TIGER - Topologically Integrated Geographic Encoding and Referencing system. http://www.census.gov/geo/www/tiger/index.html, Zugriff: April 2005.

[39] Usability Net. subjective Assessment (testing & post-release). http://www.usabilitynet.org/tools/subjective.htm, Zugriff: Januar 2005.

[40] W3C. Document Object Model (DOM) Level 2 Core Specification. http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/, Zugriff: Mai 2004.

[41] W3C. Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3.org/TR/SVG11/, Zugriff: April 2004.

[42] W3C. Document Object Model (DOM). http://www.w3.org/DOM/, Zugriff: January 2005.

[43] W3Schools. Browser Statistics. http://www.w3schools.com/browsers/browsers_stats.asp, Zugriff: Juni 2005.

[44] Juliana Williams. Yosemite National Park Hiking Map. http://www.carto.net/williams/yosemite/, Zugriff: Mai 2005.

[45] Juliana Williams and Andreas Neumann. Manipulating SVG Documents Using ECMAScript (Javascript) and the DOM. http://www.carto.net/papers/svg/manipulating_svg_with_dom_ecmascript/index.shtml, Zugriff: Mai 2005.

A Fragebogen zur ISONORM 9241/10

Mit Hilfe des standardisierten Fragebogens zur ISONORM 9241/10 von Joachim Prümper und Michael Anft wurde die Benutzerfreundlichkeit von ISIMAP ermittelt. Die grau eingefärbten Fragen sind zur Beurteilung weiterer Aspekte nachträglich eingefügt worden. Die Vorlage des Fragebogens ist in [Burmester et al., 1997] zu finden oder kann auf der Webseite [4] bezogen werden.

Beurteilung von isiMap

Anweisung

Das Ziel dieser Beurteilung ist es, Schwachstellen bei Softwaresystemen (in diesem Fall isiMap) aufzudecken und konkrete Verbesserungsvorschläge zu entwickeln.

Um dies zu bewerkstelligen, ist Ihr Urteil als Kenner des Softwaresystems von entscheidender Bedeutung! Grundlage Ihrer Bewertung sind Ihre individuellen Erfahrungen mit dem Software-Programm, das Sie beurteilen möchten.

Dabei geht es nicht um eine Beurteilung Ihrer Person, sondern um Ihre persönliche Bewertung der Software mit der Sie arbeiten.

Am besten bearbeiten Sie den Beurteilungsbogen, während Sie das zu bewertende Softwaresystem vor sich am Bildschirm haben. Dadurch haben Sie die Möglichkeit, bei der Beantwortung der einzelnen Fragen die eine oder andere Sache noch einmal zu überprüfen.

Noch ein Hinweis zur Beantwortung des Beurteilungsbogens:

Die einzelnen Normen werden über Beschreibungen konkretisiert. Diese Beschreibungen weisen immer folgende Form auf.

Beispiel Nr.1:

Die Software ...	- - -	- -	-	- / +	+	++	+++	*Die Software ...*
ist schlecht.	☐	☐	☐	☐	☐	☒	☐	ist gut.

Im ersten Beispiel wird danach gefragt, wie gut, bzw. wie schlecht die Software ist.
Der Benutzer beurteilt in diesem Fall die Software zwar als gut, sieht jedoch noch Verbesserungsmöglichkeiten.

Beispiel Nr.2:

Die Software ...	- - -	- -	-	- / +	+	++	+++	*Die Software ...*
ist langsam.	☐	☒	☐	☐	☐	☐	☐	ist schnell.

Im zweiten Beispiel beurteilt der Benutzer die Software als ziemlich langsam.

Füllen Sie bitte den Beurteilungsbogen äusserst sorgfältig aus und lassen Sie keine der Fragen aus!

Die Auswertung der Daten erfolgt anonym.

Aufgabenangemessenheit

Unterstützt die Software die Erledigung Ihrer Arbeitsaufgaben, ohne Sie als Benutzer unnötig zu belasten?

Die Software ...	- - -	- -	-	- / +	+	++	+++	*Die Software ...*
ist kompliziert zu bedienen.	☐	☐	☐	☐	☐	☐	☐	ist unkompliziert zu bedienen.
bietet nicht alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.	☐	☐	☐	☐	☐	☐	☐	bietet alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.
bietet keine sinnvolle Auswahl an Interaktionen an.	☐	☐	☐	☐	☐	☐	☐	bietet eine sinnvolle Auswahl an Interaktionen an.
bietet schlechte Möglichkeiten, sich häufig wiederholende Bearbeitungsvorgänge zu automatisieren.	☐	☐	☐	☐	☐	☐	☐	bietet gute Möglichkeiten, sich häufig wiederholende Bearbeitungsvorgänge zu automatisieren.
erfordert überflüssige Eingaben.	☐	☐	☐	☐	☐	☐	☐	erfordert keine überflüssigen Eingaben.
ist schlecht auf die Anforderungen der Arbeit zugeschnitten.	☐	☐	☐	☐	☐	☐	☐	ist gut auf die Anforderungen der Arbeit zugeschnitten.
werde ich in Zukunft nicht bei meiner Arbeit einsetzen.	☐	☐	☐	☐	☐	☐	☐	werde ich in Zukunft regelmässig bei meiner Arbeit einsetzen.

Bemerkungen:

Selbstbeschreibungsfähigkeit

Gibt Ihnen die Software genügend Erläuterungen und ist sie in ausreichendem Masse verständlich?

Die Software ...	- - -	- -	-	- / +	+	++	+++	*Die Software ...*
bietet einen schlechten Überblick über ihr Funktionsangebot.	☐	☐	☐	☐	☐	☐	☐	bietet einen guten Überblick über ihr Funktionsangebot.
verwendet schlecht verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs.	☐	☐	☐	☐	☐	☐	☐	verwendet gut verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs.
liefert in unzureichendem Masse Informationen darüber, welche Eingaben zulässig oder nötig sind.	☐	☐	☐	☐	☐	☐	☐	liefert in zureichendem Masse Informationen darüber, welche Eingaben zulässig oder nötig sind.
bietet auf Verlangen keine situationsspezifischen Erklärungen, die konkret weiterhelfen.	☐	☐	☐	☐	☐	☐	☐	bietet auf Verlangen situationsspezifische Erklärungen, die konkret weiterhelfen.
bietet von sich aus keine situationsspezifischen Erklärungen, die konkret weiterhelfen.	☐	☐	☐	☐	☐	☐	☐	bietet von sich aus situationsspezifische Erklärungen, die konkret weiterhelfen.
reagiert auf Eingaben zu langsam.	☐	☐	☐	☐	☐	☐	☐	reagiert auf Eingaben in einer angemessenen Zeit.
erstellt Interaktionen, welche Mängel aufweisen.	☐	☐	☐	☐	☐	☐	☐	erstellt Interaktionen, an welchen nichts zu bemängeln ist.

Bemerkungen:

Steuerbarkeit

Können Sie als Benutzer die Art und Weise, wie Sie mit der Software arbeiten, beeinflussen?

Die Software ...	- - -	- -	-	- / +	+	++	+++	*Die Software ...*
bietet keine Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen.	☐	☐	☐	☐	☐	☐	☐	bietet die Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen.
Erzwingt eine unnötig starre Einhaltung von Bearbeitungsschritten.	☐	☐	☐	☐	☐	☐	☐	erzwingt keine unnötig starre Einhaltung von Bearbeitungsschritten.
Ermöglicht keinen leichten Wechsel zwischen einzelnen Menüs oder Masken.	☐	☐	☐	☐	☐	☐	☐	ermöglicht einen leichten Wechsel zwischen einzelnen Menüs oder Masken.
ist so gestaltet, dass der Benutzer nicht beeinflussen kann, wie und welche Informationen am Bildschirm dargeboten werden.	☐	☐	☐	☐	☐	☐	☐	ist so gestaltet, dass der Benutzer beeinflussen kann, wie und welche Informationen am Bildschirm dargeboten werden.
erzwingt unnötige Unterbrechungen der Arbeit.	☐	☐	☐	☐	☐	☐	☐	erzwingt keine unnötigen Unterbrechungen der Arbeit.

Bemerkungen:

Erwartungskonformität

Kommt die Software durch eine einheitliche und verständliche Gestaltung Ihren Erwartungen und Gewohnheiten entgegen?

Die Software ...	- - -	- -	-	- / +	+	++	+++	*Die Software ...*
erschwert die Orientierung, durch eine uneinheitliche Gestaltung.	☐	☐	☐	☐	☐	☐	☐	erleichtert die Orientierung, durch eine einheitliche Gestaltung.
lässt einen im Unklaren darüber, ob eine Eingabe erfolgreich war oder nicht.	☐	☐	☐	☐	☐	☐	☐	lässt einen nicht im Unklaren darüber, ob eine Eingabe erfolgreich war oder nicht.
informiert in unzureichendem Masse über das, was sie gerade macht.	☐	☐	☐	☐	☐	☐	☐	informiert in ausreichendem Masse über das, was sie gerade macht.
reagiert mit schwer vorhersehbaren Bearbeitungszeiten.	☐	☐	☐	☐	☐	☐	☐	reagiert mit gut vorhersehbaren Bearbeitungszeiten.
lässt sich nicht durchgehend nach einem einheitlichen Prinzip bedienen.	☐	☐	☐	☐	☐	☐	☐	lässt sich durchgehend nach einem einheitlichen Prinzip bedienen.

Bemerkungen:

Fehlertoleranz

Bietet Ihnen die Software die Möglichkeit, trotz fehlerhafter Eingaben das beabsichtigte Arbeitsergebnis ohne oder mit geringem Korrekturaufwand zu erreichen?

Die Software ...	- - -	- -	-	- / +	+	++	+++	*Die Software ...*
ist so gestaltet, dass kleine Fehler schwerwiegende Folgen haben können.	☐	☐	☐	☐	☐	☐	☐	ist so gestaltet, dass kleine Fehler keine schwerwiegenden Folgen haben können.
informiert zu spät über fehlerhafte Eingaben.	☐	☐	☐	☐	☐	☐	☐	informiert sofort über fehlerhafte Eingaben.
liefert schlecht verständliche Fehlermeldungen.	☐	☐	☐	☐	☐	☐	☐	liefert gut verständliche Fehlermeldungen.
erfordert bei Fehlern im grossen und ganzen einen hohen Korrekturaufwand.	☐	☐	☐	☐	☐	☐	☐	erfordert bei Fehlern im grossen und ganzen einen geringen Korrekturaufwand.
gibt keine konkreten Hinweise zur Fehlerbehebung.	☐	☐	☐	☐	☐	☐	☐	gibt konkrete Hinweise zur Fehlerbehebung.
erstellt Interaktionen, welche in isiMap nicht funktionieren	☐	☐	☐	☐	☐	☐	☐	erstellt Interaktionen, welche in isiMap funktionieren
erstellt Interaktionen, welche in Web-Browsern nicht funktionieren	☐	☐	☐	☐	☐	☐	☐	erstellt Interaktionen, welche in Web-Browsern funktionieren

Bemerkungen:

Individualisierbarkeit

Können Sie als Benutzer die Software ohne grossen Aufwand auf Ihre individuellen Bedürfnisse und Anforderungen anpassen?

Die Software ...	- - -	- -	-	- / +	+	++	+++	*Die Software ...*
lässt sich von dem Benutzer schwer erweitern, wenn für ihn neue Aufgaben entstehen.	☐	☐	☐	☐	☐	☐	☐	lässt sich von dem Benutzer leicht erweitern, wenn für ihn neue Aufgaben entstehen.
lässt sich von dem Benutzer schlecht an seine persönliche, individuelle Art der Arbeitserledigung anpassen.	☐	☐	☐	☐	☐	☐	☐	lässt sich von dem Benutzer gut an seine persönliche, individuelle Art der Arbeitserledigung anpassen.
eignet sich für Anfänger und Experten nicht gleichermassen, weil der Benutzer sie nur schwer an seinen Kenntnisstand anpassen kann.	☐	☐	☐	☐	☐	☐	☐	eignet sich für Anfänger und Experten gleichermassen, weil der Benutzer sie leicht an seinen Kenntnisstand anpassen kann.
lässt sich - im Rahmen ihres Leistungsumfangs - von dem Benutzer schlecht für unterschiedliche Aufgaben passend einrichten.	☐	☐	☐	☐	☐	☐	☐	lässt sich - im Rahmen ihres Leistungsumfangs - von dem Benutzer gut für unterschiedliche Aufgaben passend einrichten.
ist so gestaltet, dass der Benutzer die Bildschirmdarstellung schlecht an seine individuellen Bedürfnisse anpassen kann.	☐	☐	☐	☐	☐	☐	☐	ist so gestaltet,dass der Benutzer die Bildschirmdarstellung gut an seine individuellen Bedürfnisse anpassen kann.

Bemerkungen:

Lernförderlichkeit

Ist die Software so gestaltet, dass Sie sich ohne grossen Aufwand in sie einarbeiten konnten und bietet sie auch dann Unterstützung, wenn Sie neue Funktionen lernen möchten?

Die Software ...	- - -	- -	-	- / +	+	++	+++	*Die Software ...*
erfordert viel Zeit zum Erlernen.	☐	☐	☐	☐	☐	☐	☐	erfordert wenig Zeit zum Erlernen.
ermutigt nicht dazu, auch neue Funktionen auszuprobieren.	☐	☐	☐	☐	☐	☐	☐	ermutigt dazu, auch neue Funktionen auszuprobieren.
erfordert, dass man sich viele Details merken muss.	☐	☐	☐	☐	☐	☐	☐	erfordert nicht, dass man sich viele Details merken muss.
ist so gestaltet, dass sich einmal Gelerntes schlecht einprägt.	☐	☐	☐	☐	☐	☐	☐	ist so gestaltet, dass sich einmal Gelerntes gut einprägt.
ist schlecht ohne fremde Hilfe oder Handbuch erlernbar.	☐	☐	☐	☐	☐	☐	☐	ist gut ohne fremde Hilfe oder Handbuch erlernbar.

Bemerkungen:

Zum Schluss

Zum Schluss bitten wir Sie, noch folgende Fragen zu beantworten:

Seit wievielen Monaten arbeiten Sie schon mit Computern?	.....	Monate
Wieviele Stunden arbeiten Sie pro Woche durchschnittlich mit Computern?	.....	Stunden
Wie gut beherrschen Sie die beurteilte Software?	sehr schlecht o o o o o o o sehr gut	
Insgesamt beurteile ich die Software als...	un-brauchbar o o o o o o o hervor-regend	
Wie oft erstellen Sie SVG-Karten?	nie o o o o o o o oft	
Wie gross ist Ihre Erfahrung mit SVG?	sehr schlecht o o o o o o o sehr gut	
Wie oft erstellen Sie interaktive SVG-Karten?	nie o o o o o o o oft	
Wie gross ist Ihre Erfahrung mit JavaScript oder ECMA-Script?	sehr schlecht o o o o o o o sehr gut	
Mit wievielen Programmen arbeiten Sie derzeit?	.....	Programme
Davon:	.....	PC-Programme
	.....	Grossrechner-programme

Was ist Ihr Beruf?	.....	
Wie alt sind Sie?	.....	Jahre
Ihr Geschlecht?	.....	m/w
Ich stehe für ein mündliches Interview zu Verfügung.	☐ Ja	☐ Nein
Name (der Name wird nur für allfällige Rückfragen benötigt):	.....	