



Universität Würzburg
Lehrstuhl für Informatik II



Diplomarbeit

Web-basierter GIS-Service zur Überwachung von Infektionen

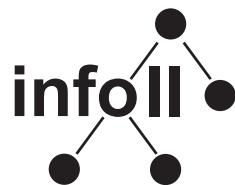
Autor: Markus Reinhardt

Betreuer: Prof. Dr. Jürgen Albert, Prof. Dr. med. Ulrich Vogel, Prof. Dr. Dag Harmsen, Dr. med. Johannes Elias,
Dipl.-Inf. Jörg Rothgänger

Würzburg, den 23. Dezember 2005



Universität Würzburg
Lehrstuhl für Informatik II



Diplomarbeit

Web-basierter GIS-Service zur Überwachung von Infektionen

Diplomand: Markus Reinhardt

Matrikelnummer: 1277246

Anschrift: Friesstraße 18
97074 Würzburg

Betreuer: Prof. Dr. Jürgen Albert
Prof. Dr. med. Ulrich Vogel
Prof. Dr. Dag Harmsen
Dr. med. Johannes Elias
Dipl.-Inf. Jörg Rothgänger

Beginn: 23. Juni 2005
Abgabe: 23. Dezember 2005

Inhaltsverzeichnis

1. Einleitung	1
1.1. Einführung	1
1.2. Aufgabenstellung	2
1.3. Aufbau der vorliegenden Arbeit	3
2. Grundlagen	5
2.1. Epidemiologie	5
2.1.1. Epidemiologische Kennzahlen und Definitionen	5
2.1.2. Epidemiologie in Deutschland	6
2.1.3. Meningokokken	7
2.1.4. Arbeit des Nationalen Referenzzentrums für Meningokokken	8
2.2. Geografische Informationssysteme	10
2.2.1. Räumliches Datenmodell	11
2.2.2. Koordinaten und Kartenprojektionen	12
2.2.3. Topologie	14
2.2.4. Einsatz von Geoinformationen und Statistik in der Epidemiologie	14
3. Definition der Anforderungen	17
3.1. Zielbestimmungen	17
3.1.1. Musskriterien	17
3.1.2. Wunschkriterien	18
3.1.3. Abgrenzungskriterien	18
3.2. Produkt-Einsatz	18
3.2.1. Anwendungsbereiche	18
3.2.2. Zielgruppen	19
3.2.3. Betriebsbedingungen	19
3.3. Produktfunktionen	19
3.4. Produkt-Daten	21
3.4.1. Eingabedaten	21
3.4.2. Ausgabedaten	22
3.5. Produkt-Leistungen	22
3.6. Qualitätsanforderungen	22
3.7. Technische Produktumgebung	23
3.7.1. Hardware-Umgebung	23
3.7.2. Software-Umgebung	24
3.8. Entwicklungsumgebung	25
3.8.1. Programmiersprache	25
3.8.2. Entwicklungswerkzeuge	25

Inhaltsverzeichnis

4. Untersuchung der Voraussetzungen	27
4.1. Web-basierte Systeme zur Überwachung von Infektionen	27
4.2. Anforderungen an ein Web-basiertes epidemiologisches GIS	31
4.2.1. Reichhaltige Internetanwendung als Benutzerschnittstelle	31
4.2.2. Geografisches Informationssystem	33
4.2.3. Aktualisierung der Daten	34
4.2.4. Automatische Identifizierung von Clustern	34
4.3. Anwendungsrelevante freie Softwarepakete	35
4.3.1. Web-basierte Anwendungen	35
4.3.2. Datenbanksysteme	39
4.3.3. Geografische Informationssysteme	40
4.3.4. Statistiksoftware	42
5. Konzepte	43
5.1. Datenmodell, Präsentation und Programmsteuerung	43
5.1.1. Globale und benutzerspezifische Daten	44
5.1.2. Präsentation der Informationen	46
5.1.3. Wohl definierte Zustandsänderungen	47
5.2. Leichte Erweiterbarkeit	48
5.2.1. Dynamische Karten- und Ebenenparameter	48
5.2.2. Änderbarkeit der Kartendefinition des MapServers	49
5.2.3. Aufteilung der Systemkomponenten in Pakete	50
5.2.4. Erweiterungen an externen Schnittstellen	51
5.3. Kapselung des MapServers und Caching	52
5.3.1. Datenschutz durch indirekten Zugriff	52
5.3.2. Optimierte Antwortzeiten durch Caching	53
6. Implementierungsdetails	55
6.1. Serverseitige Anwendung	55
6.1.1. Initialisierung der Karte	55
6.1.2. Export der Karte in unterschiedliche Dateiformate	58
6.1.3. Zustandsänderung durch Aktionen in Struts	59
6.2. Definition von Karten für den MapServer	61
6.2.1. Inzidenzdarstellung	62
6.2.2. Darstellung von Clustern	64
6.2.3. Streuung von Punktinformationen	66
6.3. Clientanwendung mit OpenLaszlo	66
6.3.1. Dynamischer Aufbau der Benutzeroberfläche	66
6.3.2. Kommunikation zwischen Laszlo und Struts	69
6.4. Serverseitige Clusteridentifizierung mittels SaTScan	70
7. Zusammenfassung und Ausblick	73
7.1. Zusammenfassung	73
7.2. Erweiterungsmöglichkeiten	75
7.2.1. Integration des Statistikpaketes R	75
7.2.2. Animationen über die Dimension Zeit	77
7.2.3. Ausbau der Benutzeroberfläche	77
Danksagung	79

A. Softwaresystem EpiDeGIS	81
A.1. Installationsanleitung unter Linux	81
A.1.1. Grundlagen	81
A.1.2. Installation der benötigten Softwarekomponenten	82
A.1.3. Installation und Konfiguration der EpiDeGIS-Webanwendung .	84
A.2. Vorstellung des Kartenbetrachters	86
A.2.1. Gesamtübersicht	86
A.2.2. Ebenen	87
A.3. Informationen für Entwickler	91
A.3.1. Projektverwaltung mit Maven	91
A.3.2. HTML-Dokumentation des Projektes EpiDeGIS mit Quellcode und JavaDoc	92
B. Geografische Daten und Zensusdaten	93
C. Inhalt des Datenträgers zu dieser Diplomarbeit	95
D. Datei-Listings	97
E. Abkürzungen	101
Selbstständigkeitserklärung	103

1. Einleitung

1.1. Einführung

Das Jahr 2005 steht ganz im Zeichen der Entwicklung eines neuen Abschnitts des Informationszeitalters. Mit “Google Earth” und “NASA World Wind” entstehen völlig neuartige Formen von Informationssystemen [82]. Sie bieten dem Anwender Geoinformatik zum Anfassen.

Die Geoinformatik beschäftigt sich als Teilgebiet der Informatik schon lange mit der Aufgabe, geografische Daten zu visualisieren. So genannte Geoinformationssysteme (GIS) bieten dabei vielfältige Möglichkeiten zur Bewertung von Informationen mit räumlichem Bezug.

“Google Earth” und “NASA World Wind” sind in der Entwicklung von GIS jedoch die ersten Software-Produkte, die Geoinformatik zum Anfassen bieten. Anfassen ist dabei durchaus wörtlich zu nehmen. Die am ehesten als “Planetenbrowser” bezeichneten Programme erlauben es, die Erdkugel in einer dreidimensionalen Visualisierung zu drehen und zu zoomen, um darauf Informationen zu lokalisieren (siehe Abbildung 1.1). Die Firma Google strebt mit dieser Software den Aufbau von neuen Märkten in der Informationstechnik an. Diese nutzt das ureigene menschliche Verständnis für räumliche Zusammenhänge.

Die vorliegende Arbeit befasst sich jedoch nicht mit den Märkten, die GI-Systeme bieten. Sie betrachtet die Möglichkeiten, wie einem Anwender mittels Karten Informationen aus dem Bereich der Epidemiologie präsentiert werden können. Der Zugang zu Informationen, welche die öffentliche Gesundheit betreffen, ist für Vertreter des öf-



Abbildung 1.1.: Darstellung der Weltkugel als Informationsquelle in “Google Earth”

1. Einleitung



Abbildung 1.2.: Ausbreitung der Geflügelpest/Vogelgrippe (aviäre Influenza Subtyp A/H5N1), Stand: Oktober 2005 [6]

fentlichen Gesundheitsdienstes und (politische) Entscheidungsträger, aber auch für die Öffentlichkeit von großem Wert.

Die Wichtigkeit solcher Informationen zeigt sich beispielsweise an der Ausbreitung der "Vogelgrippe". Dieser Begriff bezeichnet im aktuellen Zusammenhang die durch den Subtyp A/H5N1 verursachte Geflügelpest (aviäre Influenza). Diese hat sich - ausgehend von China im Jahr 1997 - bis heute in großen Teilen Asiens verbreitet [13]. Abbildung 1.2 zeigt die Verbreitung anhand einer Karte. Durch die Berichterstattung der Medien zu diesem Thema wird deutlich, dass unter der Bevölkerung ein großer Informationsbedarf an epidemiologischen Daten herrscht.

Bis dato gibt es kaum ein System, das anhand von Karten einer großen Benutzergruppe eingängig epidemiologische Daten vermitteln kann. Dabei könnte so eine Lösung durch Aktualität und eine entsprechende Visualisierung intuitiv über die geografische Verteilung von Erkrankungsfällen aufklären.

Neben der reinen Visualisierung könnte das System auch eine automatisierte statistische Analyse der Falldaten durchführen. Anhand dieser ließen sich potenzielle Erkrankungsausbrüche frühzeitig erkennen.

1.2. Aufgabenstellung

Web-basierter, einer breiten Benutzergruppe zugänglicher, epidemiologischer GIS-Service

Die Aufgabe der vorliegenden Diplomarbeit ist es, einen Web-basierten GIS-Service zur Überwachung von Infektionen zu implementieren. Dieser soll es dem Anwender in einer über das Internet ausführbaren Applikation erlauben, Karten zu generieren, welche die geografische Verteilung von Erkrankungsfällen eines Erregers aufzeigen.

Anhand dieser Karten sind Epidemiologen, Vertreter des öffentlichen Gesundheitsdienstes, (politische) Entscheidungsträger und die Öffentlichkeit in der Lage, Informationen über die Verteilung von Krankheitsfällen zu erlangen. Die Web-basierte Anwendung soll hierbei so ausgelegt sein, dass sie möglichst wenig Anforderungen an die Konfigu-

1.3. Aufbau der vorliegenden Arbeit

ration des clientseitigen Rechners stellt. Nur so ist eine weit reichende Zugänglichkeit zu den Informationen gesichert.

Ausbruchsanalyse

Zusätzlich hat die Arbeit zum Ziel, eine automatisierte Risikoanalyse anhand der im System gespeicherten Erkrankungsinformationen durchzuführen. Ein in das Software-system integriertes Statistikwerkzeug führt hierzu regelmäßige Tests durch, deren Ergebnisse Hinweise auf Kartenregionen mit einer überdurchschnittlichen Häufung von Erkrankungsfällen geben. Anhand entsprechender Darstellungen werden die Risikoregionen visualisiert.

Durch diesen neuartigen Ansatz nimmt das System die Stellung eines Web-basierten Frühwarnsystems ein, das mögliche Erkrankungsausbrüche vorhersagen kann.

Übertragbarkeit auf andere Krankheitserreger und Länder

Ein wichtiges Kriterium dieser Arbeit ist es, den Entwurf der Software auf Erweiterbarkeit auszulegen. Das zu entwickelnde System soll sich auf verschiedene Krankheitserreger anwenden und auch in anderen Ländern einsetzen lassen können.

Kooperationen

Diese Arbeit ist in Kooperation mit dem Nationalen Referenzzentrum für Meningokokken [25] des Instituts für Hygiene und Mikrobiologie der Universität Würzburg und der Firma Ridom GmbH [33] aus Würzburg entstanden.

Das Nationale Referenzzentrum für Meningokokken ermittelt seit mehreren Jahren detaillierte Informationen zu in Deutschland aufgetretenen Meningokokken-Erkrankungen. Das zu entwickelnde System soll diese Daten beispielhaft integrieren.

Die Firma Ridom ist an einer Machbarkeitsstudie zur Entwicklung eines Web-basierten epidemiologischen GIS interessiert.

Projektname

Das für diese Diplomarbeit entwickelte Softwaresystem trägt den Arbeitstitel **EpiDeGIS**.

1.3. Aufbau der vorliegenden Arbeit

Diese Diplomarbeit gliedert sich in folgende Kapitel:

Kapitel 2 vermittelt alle Grundlagen aus den Bereichen Epidemiologie und Geoinformationssysteme, die für das Verständnis dieser Arbeit erforderlich sind.

1. Einleitung

Kapitel 3 gibt das anhand der Lastenhefte des Nationalen Referenzzentrums für Meningokokken und der Firma Ridom GmbH erstellte Pflichtenheft zur Softwareentwicklung wieder.

Kapitel 4 untersucht die Voraussetzungen, die an einen Web-basierten GIS-Service zur Überwachung von Infektionen gestellt werden und beschreibt anwendungsrelevante Open-Source-Software, die in diesem Projekt Verwendung findet.

Kapitel 5 zeigt Konzepte, die bei dem Entwurf des Softwaresystems EpiDeGIS angewendet wurden.

Kapitel 6 veranschaulicht anhand von ausgesuchten Implementierungsbeispielen die Umsetzung der Entwurfskonzepte.

Kapitel 7 fasst alle Ergebnisse zusammen.

2. Grundlagen

Das folgende Kapitel vermittelt die Grundlagen, die für das Verständnis dieser Arbeit notwendig sind. Es gibt einen Einblick in die Epidemiologie und die Geografischen Informationssysteme (GIS). Die Inhalte der einzelnen Abschnitte beschränken sich dabei auf die Teilbereiche dieser Disziplinen, die in der vorliegenden Diplomarbeit verwendet werden.

2.1. Epidemiologie

Definition Die Epidemiologie ist die Lehre der Verbreitung und Ursachen von gesundheitsrelevanten Zuständen und Ereignissen in Bevölkerungsgruppen. Das epidemiologische Wissen wird im Allgemeinen angewandt, um Gesundheitsprobleme der Bevölkerung zu beobachten und so gut wie möglich zu kontrollieren. [67]

Diese Wissenschaft erfasst und bewertet Erkrankungshäufungen. Epidemiologen berücksichtigen hierzu verschiedene Faktoren. Neben dem Alter und Geschlecht eines Erkrankten spielen auch das soziale Umfeld und Umwelteinflüsse eine Rolle. Häufungen von Krankheitsfällen haben immer eine räumliche und zeitliche Komponente. Diese Informationen sind eine wichtige Bewertungshilfe.

Die Epidemiologie entwickelt mathematische Methoden und Modelle um die erfassten Daten zueinander in Beziehung zu bringen. Ein einfaches Mittel sind Kennzahlen. Diese helfen einem Beobachter dabei, aufgetretene Fälle objektiver zu beurteilen.

2.1.1. Epidemiologische Kennzahlen und Definitionen

Prävalenz

Diese Größe gibt die Anzahl der Erkrankungsfälle in einer Population an. Sie berechnet sich mit folgender Formel:

$$\frac{\text{Anzahl der Erkrankten}}{\text{Anzahl der Mitglieder einer Population}}$$

Inzidenz

Dieser Wert bezeichnet die Anzahl der Neuerkrankungen einer Population während eines bestimmten Zeitraumes. Die Jahresinzidenz bezieht sich dabei üblicherweise auf

2. Grundlagen

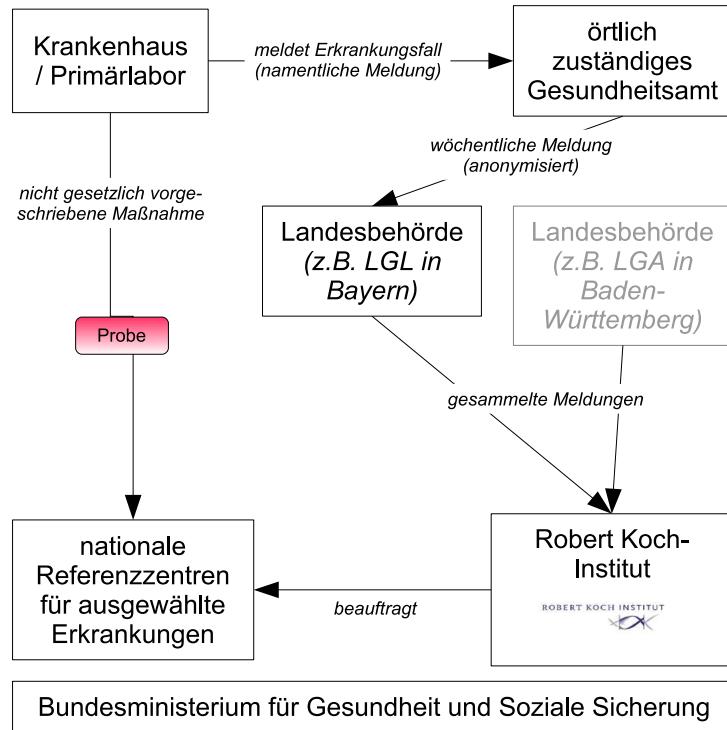


Abbildung 2.1.: Erfassung von Infektionserkrankungen in Deutschland.

100.000 Einwohner. Anhand der Inzidenz lässt sich die Rate der Neuerkrankungen in einem Gebiet unabhängig von der Bevölkerungsdichte betrachten. In Ballungszentren gibt es immer mehr Fälle als in einem kleinen Dorf. Diese Kennzahl schafft die Vergleichbarkeit zwischen einer Großstadt und einer 1000-Seelen-Gemeinde. Bei Krankheiten mit einer langen Erkrankungsdauer ist die Inzidenz niedriger als die Prävalenz.

Cluster

Dieser Begriff bezeichnet in der Epidemiologie eine zeitlich und örtlich verstärkt auftretende Häufungen von Erkrankungen des gleichen Typs.

2.1.2. Epidemiologie in Deutschland

Das Infektionsschutzgesetz (IfSG) [50] regelt in Deutschland die Zusammenarbeit von Behörden des Bundes, der Länder und der Kommunen, Ärzten, Tierärzten, Krankenhäusern, wissenschaftlichen Einrichtungen sowie sonstigen Beteiligten in allen Bereichen der Epidemiologie. Abschnitt 3 dieses Gesetzes beschreibt das Meldewesen. Abbildung 2.1 gibt einen Überblick.

Feststellende Ärzte, Krankenhäuser und andere Personen¹ müssen meldepflichtige Krankheiten² bzw. Nachweise von Krankheitserregern³ an das zuständige Gesundheits-

¹siehe auch IfSG § 8: zur Meldung verpflichtete Personen

²siehe auch IfSG § 6: meldepflichtige Krankheiten

³siehe auch IfSG § 7: meldepflichtige Nachweise von Krankheitserregern

amt übermitteln. Dieses erhält auch personenbezogene Patientendaten⁴. Das Amt muss die gesammelten Daten in anonymisierter Form⁵ an die zuständige Landesbehörde melden⁶. Die für Bayern zuständige Behörde ist beispielsweise das Bayerisches Landesamt für Gesundheit und Lebensmittelsicherheit (LGL).

Die zentrale Einrichtung der Bundesregierung auf dem Gebiet der Krankheitsüberwachung und -prävention ist das Robert Koch-Institut (RKI). Die Länder übermitteln dieser Behörde laufend die aktuellen Erkrankungsdaten (in der vorliegenden nichtnamenlichen Form). Nach dem IfSG hat das RKI folgende Aufgaben⁷:

Entwicklung von “Konzeptionen zur Vorbeugung übertragbarer Krankheiten sowie zur frühzeitigen Erkennung und Verhinderung der Weiterverbreitung von Infektionen. Dies schließt die Entwicklung und Durchführung epidemiologischer und laborgestützter Analysen sowie Forschung zu Ursache, Diagnostik und Prävention übertragbarer Krankheiten ein.”

Das Robert Koch-Institut

- “erstellt [...] Richtlinien, Empfehlungen, Merkblätter und sonstige Informationen zur Vorbeugung, Erkennung und Verhinderung der Weiterverbreitung übertragbarer Krankheiten”,
- “hat [...] Kriterien (Falldefinitionen) für die Übermittlung eines Erkrankungs- oder Todesfalls und eines Nachweises von Krankheitserregern zu erstellen”,
- “fasst [...] [übermittelte] Meldungen zusammen, um sie infektionsepidemiologisch auszuwerten”,
- veröffentlicht regelmäßig “die Zusammenfassungen und die Ergebnisse der infektionsepidemiologischen Auswertungen [...].”

Das RKI benennt seit 1995 bei besonders wichtigen übertragbaren Krankheitserregern wie *Mycobacterium tuberculosis*, Meningokokken oder Salmonellen Nationale Referenzzentren (NRZ) zu deren Überwachung [22]. Die Berufung erfolgt durch das Bundesministerium für Gesundheit und Soziale Sicherung (BMGS) für jeweils drei Jahre.

Diese Zentren haben unter anderem Beratungsaufgaben, erstellen Gutachten und führen Laboruntersuchungen von Erkrankungsfällen durch. Sie übernehmen Aufgaben in der Qualitätssicherung und kooperieren mit dem RKI und internationalen Organisationen. Das RKI empfiehlt Krankenhäusern und Primärlaboratorien diese Dienstleistungen in Anspruch zu nehmen und entnommene Proben von Patienten an das entsprechende Referenzzentrum zu schicken.

2.1.3. Meningokokken

Meningokokken⁸ sind gefährliche übertragbare Krankheitserreger des Menschen. Es handelt sich dabei um gramnegative Bakterien, die in 12 verschiedene Serogruppen

⁴siehe auch IfSG § 9: namentliche Meldung

⁵siehe auch IfSG § 10: nichtnamenliche Meldung

⁶siehe auch IfSG § 11: Übermittlungen durch das Gesundheitsamt und die zuständige Landesbehörde

⁷siehe auch IfSG § 4: Aufgaben des Robert Koch-Institutes

⁸wissenschaftlicher Name: *Neisseria meningitidis*

2. Grundlagen

unterteilt werden. Die Mikrobiologie teilt Variationen von Bakterien oder Viren anhand von Merkmalen auf deren Oberfläche in so genannte Serogruppen, Serotypen oder Serovare ein, wenn sich diese anhand spezifischer Antikörper unterscheiden lassen. In Deutschland kommen von diesen seit Jahren fast ausschließlich die Serogruppe B und C vor. Altersabhängig besiedeln die Bakterien bei bis zu über 30% der Gesunden die Schleimhäute im Nasen-Rachenraum [45]. Die Besiedelung löst jedoch in den wenigsten Fällen eine Erkrankung aus.

Die Jahresinzidenz lag 2003 in Deutschland bei ca. 1,1 Fällen pro 100.000 Einwohner [79]. Im Jahr 2004 sind insgesamt 603 Fälle von Meningokokkenerkrankungen gemeldet worden. Zu den verursachten Krankheiten zählen Hirnhautentzündung und Blutvergiftung (Sepsis). 5-10% der Erkrankungsfälle enden tödlich.

Säuglinge und Kleinkinder bis zwei Jahren sind am häufigsten von der Krankheit betroffen. Gegen die häufige Serogruppe B gibt es bis heute noch keinen generell verfügbaren Impfstoff. Die Krankheit kann in Einzelfällen von den ersten Anzeichen bis zum Tod innerhalb eines Tages verlaufen. Diese Tatsachen erfordern bei Meningokokken trotz der geringen Inzidenz eine sorgfältige Überwachung.

2.1.4. Arbeit des Nationalen Referenzzentrums für Meningokokken

Das Nationale Referenzzentrum für Meningokokken (NRZM) überwacht im Auftrag des Robert Koch-Instituts Meningokokkenerkrankungen in Deutschland. Es erfüllt die unter 2.1.2 angegebenen Aufgaben. Dazu gehört ein großes Leistungsangebot in der molekularen Epidemiologie. Zwei Bereiche aus der Arbeit des Referenzzentrums werden im Folgenden näher beschrieben.

Typisierung von Meningokokken-Erkrankungsfällen

Das NRZM führt mehrere verschiedene Typisierungsarten bei eingesandten Meningokkenstämmen durch:

- *serologische Typisierung*
Bestimmung der Serogruppe (definiert durch die Zuckerreste der Polysaccharidkapsel).
- *molekularbiologische Typisierung*
 - mittels PorA- und FetA-Sequenzierung
Typisierungsverfahren, das zunehmend für die serologische Bestimmung von einem Serotyp und Serosubtyp eingesetzt wird. Diese Werte erlauben eine viel feinere Typisierung als die Serogruppe allein.
 - mittels Multi-Locus-Sequenz-Typisierung (MLST) ausgewählter Isolate

Die Mitarbeiter des Referenzzentrums bestimmen mittels der PorA- und FetA-Sequenzierung einen genauen Typ für jede eingesandte Probe. Die Nomenklatur benennt diesen mit der Zeichenkette *{Serogruppe}:P1.{PorA}:F{FetA}*. Der so genannte

Feintyp identifiziert einen bestimmten Meningokokkenerreger eindeutig. Die Feintypisierung hilft unter anderem bei folgenden Aufgaben:

- *Erfassung der epidemiologischen Dynamik auf nationaler und internationaler Ebene*

Meningokokken verändern sich nur marginal bei einer Übertragung von Mensch zu Mensch. Wenn sich Person A an Person B ansteckt, tragen beide den selben Feintyp in sich. Hierdurch ist es möglich, dass einzelne Feintypen zu einem Zeitpunkt örtlich verstärkt auftreten oder komplett verschwinden.

- *Analyse von epidemiisch verknüpften Fällen (Ausbruchsanalyse)*

Person A und B werden aufgrund eines Verdachts auf Meningokokken-Meningitis zur stationären Behandlung in ein Kreiskrankenhaus eingewiesen. Dort entnommene Proben werden an das NRZM geschickt, um anhand des Feintyps feststellen zu lassen, ob die beiden Fälle in einem Zusammenhang stehen.

- *Ermittlung von Daten für die Entwicklung von Impfstoffen*

Die Hersteller von Impfstoffen benötigen genaue Informationen über die Oberflächenproteine eines Erregers. Die Feintypisierung liefert dieses Wissen.

Computergestützte Clusteridentifizierung am NRZM

Die Epidemiologen des Referenzzentrums nutzen die Feintypisierungsergebnisse zu weiteren statistischen Analysen. Bei der computergestützten Clusteridentifizierung handelt es sich um ein Verfahren, das mittels einer von Kulldorff et al. [62] entwickelten Software durchgeführt wird. Das Programm SaTScan [66] kann anhand der spatio-temporalen⁹ Informationen von Ereignissen erkennen, ob diese in einem Zusammenhang stehen (siehe 2.2.4).

Bezogen auf die Feintypen erkennt die Clusteridentifizierung, im Gegensatz zu der im vorherigen Abschnitt genannten Ausbruchsanalyse, selbstständig zusammengehörige Fälle eines Feintyps. Das NRZM entdeckt damit frühzeitig potenzielle Ausbrüche von Meningokokken-Erkrankungen. Anhand dieser Information sind die Landesbehörden und das Robert Koch-Institut in der Lage, z.B. Impfempfehlungen für die betroffenen Landkreise auszusprechen.

Informationsfluss und Arbeitsabläufe

Der Informationsfluss und die Arbeitsabläufe im NRZM sind in Abbildung 2.2 schematisch dargestellt.

1. Ein Krankenhaus oder Primärlabor schickt eine Probe an das Referenzzentrum.
2. Ein Mitarbeiter erfasst die Probe mit Eingangsdatum und weiteren Patientendaten (u. a. Alter, Initialen, Wohnort) in einer Access-Datenbank.
3. Das Labor führt die oben genannten Typisierungsmethoden durch. Der Datenbankeintrag einer Probe wird um den Feintyp ergänzt.

⁹raum-zeitlichen

2. Grundlagen

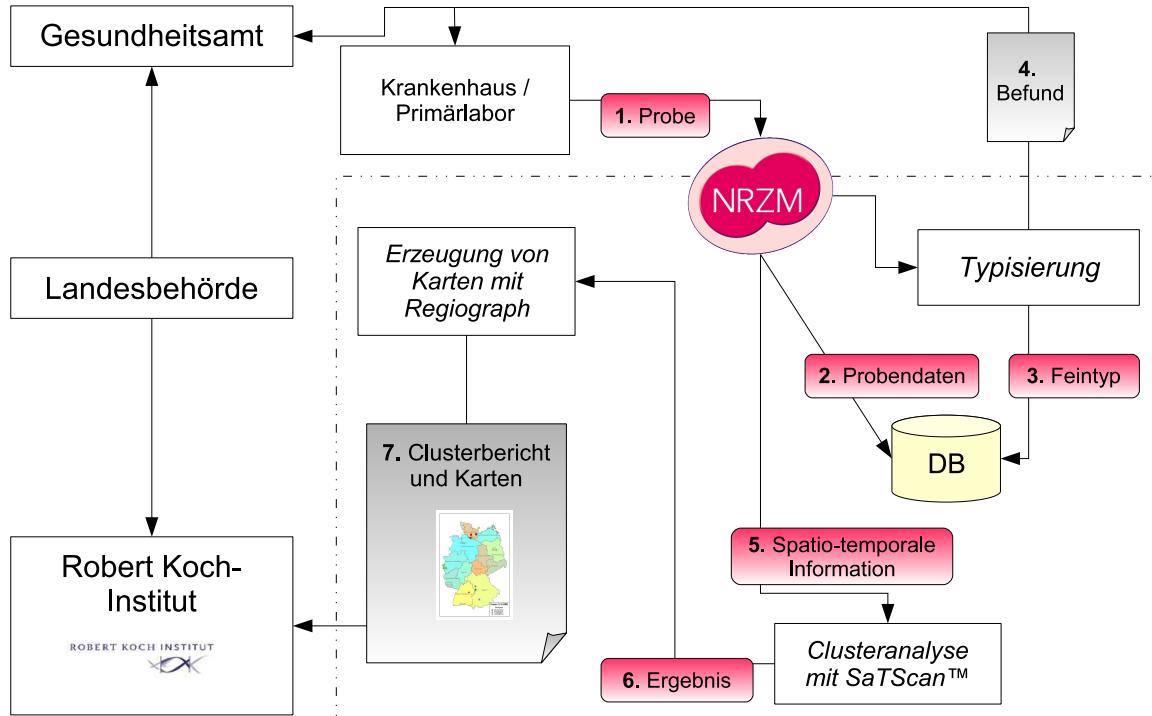


Abbildung 2.2.: Bisheriger Informationsfluss bei der Typisierung von eingesandten Proben im NRZM

4. Das Referenzzentrum schickt einen Befund mit allen Informationen zu dem gefundenen Meningokokken-Stamm an den Auftraggeber und das zuständige Gesundheitsamt.
5. Die Epidemiologen führen spatio-temporale Clusteridentifizierungen auf Basis der gefundenen Feintypen durch und
6. erzeugen mit Regiograph¹⁰ Clusterkarten.
7. Das RKI erhält diese Karten zusammen mit einem Clusterbericht und stimmt evtl. zu treffende Maßnahmen mit den Landesbehörden und dem Gesundheitsamt ab.

2.2. Geografische Informationssysteme

Definition Ein Geografisches Informationssystem (GIS) ist ein Computersystem (Hard- und Software). Es hilft dem Benutzer dabei, Daten mit räumlichem Bezug zu verwalten und zueinander in Beziehung zu setzen. GIS kann als eine Art von einem räumlichen Entscheidungsfindungssystem beschrieben werden. (Heywood et al. [57])

Das erste landesweite Geoinformationssystem hat die Firma ESRI¹¹ 1973 [46] unter dem Titel "The Maryland Automated Geographic System" erstellt. Seitdem gewinnt GIS

¹⁰Desktop GIS-Anwendung von der Firma GfK MACON

¹¹Environmental Systems Research Institute

in vielen Behörden, Firmen und auch bei Privatpersonen immer mehr an Bedeutung.

So hat zum Beispiel die Gemeinde Oerlenbach bei Bad Kissingen im Jahr 2005 eine ge-splittete Abwassergebühr eingeführt. Mitarbeiter teilen mittels Orthofotos¹² und einer GIS-Software alle Grundstücke der Gemeinde in die verschiedenen Versiegelungsarten (Dachflächen, Betonflächen, Hofpflaster, etc.) auf. Die Abwassergebühr errechnet sich u. a. nach dem Versiegelungsgrad und somit nach der Menge von Regenwasser, das in die Kanalisation gelangt. Ohne die Hilfe eines Geoinformationssystems wäre diese per Gerichtsurteil vorgeschriebene Maßnahme mit erheblich mehr Aufwand verbunden gewesen.

2.2.1. Räumliches Datenmodell

Der Benutzer eines Geoinformationssystems möchte bestimmte Bereiche der Realität modellieren. Das Modell soll so einfach wie möglich und so komplex wie nötig sein. GI-Systeme wandeln mit verschiedenen Methoden die vorhandenen Daten in digitale Karten um.

Grundlegende Komponenten zur Abbildung der Realität

GI-Systeme bilden die Realität mittels einfacher geometrischer Elemente ab:

- *Punkte*

Punkte werden, abhängig von dem benötigten Detailgrad der Karte, für unterschiedliche Informationen eingesetzt. Ein Stadtplan verwendet sie beispielsweise, um die Position einer Touristeninformation zu markieren. Auf Landkarten können andererseits ganze Städte nur anhand ihrer Mittelpunktkoordinaten platziert sein.

- *Linien*

Linien repräsentieren eine geordnete Menge von Punkten. Flüsse oder Straßen sind in GI-Systemen oft in dieser Form gespeichert.

- *Flächen*

Flächen können zum Beispiel Landesgrenzen darstellen. GI-Systeme speichern sie als Polygone. Wenn mehrere Einzelflächen eine Einheit bilden (z.B. ein Staat und seine Inseln), werden diese als Gruppe von Polygonen gespeichert.

- *Rasterdaten*

Rasterdaten sind spalten- und zeilenweise in einzelne Bildpunkte aufgeteilte Vektordaten. Im Vergleich zu den obigen punktgenauen geometrischen Daten gehen Informationen verloren (siehe Abbildung 2.3). Eine nützliche Einsatzmöglichkeit von Rasterdaten in Geoinformationssystemen sind Luft- oder Satellitenbilder.

¹²verzerrungsfreie und maßstabsgerechte Luftbildaufnahmen

2. Grundlagen

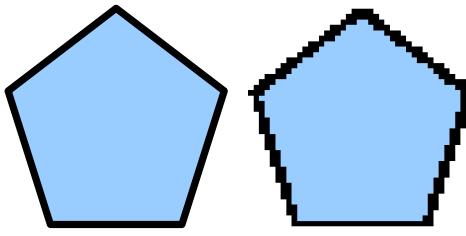


Abbildung 2.3.: Vergleich eines Polygons in Vektor- und Rasterdarstellung

2.2.2. Koordinaten und Kartenprojektionen

Jeder Punkt auf der Erdoberfläche und somit jede Koordinate in einem GIS ist exakt durch die geografische Länge und Breite definiert. Die Erde ist in 360 Längenkreise und 180 Breitenkreise aufgeteilt. Die Länge definiert sich dabei als Winkelmaß bis je 180° östlich (E) oder westlich (W) des Nullmeridians, der durch die Sternwarte von Greenwich verläuft; die Breite bis je 90° nördlich (N) oder südlich (S) des Äquators. Eine bestimmte Position auf dem Globus lässt sich am einfachsten mit Hilfe von geografischen Koordinaten beschreiben.

GI-Systeme stehen vor der Aufgabe, die gekrümmte Erdoberfläche auf eine flache Karte zu übertragen (Kartenprojektion). Dieser Vorgang lässt sich nach Heywood et al. am einfachsten folgendermaßen veranschaulichen:

Ein Globus ist am Nordpol an der Decke eines Raumes aufgehängt und von innen beleuchtet. Das Licht projiziert ein Bild der Erdoberfläche auf die Wände. Das Experiment wird mit drei verschiedenen Räumen durchgeführt. Einer ist quadratisch (flache Wände), einer zylinderförmig (runde Wände) und der dritte kegelförmig (vergleichbar mit einem Tipi). Die Wände lassen sich theoretisch auf eine Fläche ausbreiten. Die resultierenden Karten haben, abhängig vom der Form des Raumes, verschiedene Eigenschaften (siehe Abbildung 2.4).

Vergleich der grundlegenden Kartenprojektionen

- Zylinderprojektion
 - kontinuierliche Abbildung der Erde
 - die dem Zylinder am nächsten liegenden Punkte werden unverzerrt abgebildet
 - gut geeignet, um Karten von kleinen Gebieten zu erzeugen
 - Flächen bleiben weit gehend erhalten
- Azimutalprojektion (Ebene)
 - nur ein Teil der Erdoberfläche (maximal die Hälfte) ist sichtbar
 - Verzerrungen an allen vier Seiten
 - Abstände bleiben weit gehend erhalten
- Kegelprojektion

- Flächen stimmen nicht mehr überein
- die Abstände im unteren Bereich des Bildes sind stark verzerrt
- Größenverhältnisse bleiben weitgehend erhalten

Viele Kartografen und Mathematiker haben sich dem Thema Kartenprojektion angenommen und die oben genannten Projektionsarten gemäß bestimmter Anforderungen umgesetzt.

Mercator-Projektion

Diese Projektion wurde von Gerhard Mercator 1569 vorgestellt. Sie ist winkeltreu und wird häufig zur Navigation in der Schifffahrt eingesetzt. Die Mercator-Projektion gehört zu den Zylinderprojektionen. Der Zylinder berührt die Erde entlang des Äquators. Seefahrer haben durch die Winkeltreue den Vorteil, dass Kurse mit konstanter Richtung als gerade Linien auf der Karte erscheinen. Ein Nachteil dieser Projektion ist die starke Größenverzerrung in Richtung der Pole. Grönland ($2,2$ Mio. km^2) erscheint beispielsweise in Abbildung 2.5 größer als Südamerika ($17,8$ Mio. km^2).

Traversale Mercator-Projektion

Carl Friedrich Gauß und Johann Heinrich Louis Krüger haben diese Projektion entwickelt. Sie wird auch als Gauß-Krüger-Projektion bezeichnet und ist eine konforme, d.h. in kleinsten Bereichen winkeltreue, Abbildung.

Im Gegensatz zur Mercatorprojektion ist der die Erde umgebende Zylinder hier um 90° gedreht und berührt somit einen Längenkreis. Weiterhin bildet Gauß-Krüger die Erde nicht im Ganzen, sondern streifenweise ab.

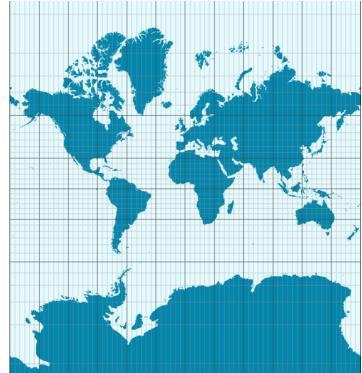


Abbildung 2.5.: Mercator-Projektion [60]

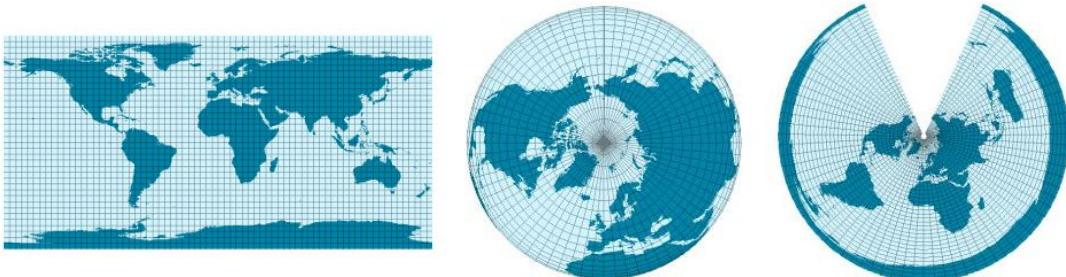


Abbildung 2.4.: Beispiele für das erzeugte Bild der Erdoberfläche bei einer Zylinder-, Azimutal- oder Kegelprojektion [60]

2. Grundlagen

Die Längenkreise, an denen der Zylinder die Erde berührt, werden Bezugsmeridiane genannt. Dort ist die Projektion verzerrungsfrei. Viele Koordinatensysteme verwenden die traversale Mercator-Projektion nach diesem Prinzip. Die zwei wichtigsten für Deutschland sind:

- *Gauß-Krüger-Koordinatensystem*

Das Gauß-Krüger-Koordinatensystem teilt die Erde in 3° breite Meridianstreifen auf. Jeder Streifen wird anhand seines Bezugsmeridians, der genau in der Mitte des Streifens liegt, von eins bis sieben durchnummeriert. Die sieben verwendeten Längenkreise sind 3°E , 6°E , 9°E , 12°E , 15°E , 18°E und 21°E [72]. Dieses Koordinatensystem bildet nur ein Teilgebiet der nördlichen Hemisphäre ab. Deutschland setzt Gauß-Krüger seit 1927 als amtliche Vermessungsmethode ein. Zur Zeit findet eine Umstellung auf das UTM-System statt.

- *UTM-Koordinatensystem*

UTM steht für den englischen Ausdruck “Universal Transverse Mercator”. Im Gegensatz zu dem Gauß-Krüger-System setzt das UTM-System 6° breite Meridianstreifen ein. Das ergibt bei 360 Längenkreisen 60 Bezugsmeridiane. Diese überspannen in der geografischen Breite ein Gebiet von 80°S bis 84°N . Bis auf die Polregionen wird die ganze Welt abgedeckt. Jeder Bezugsmeridian wird wieder nummeriert (Zone). Deutschland liegt in den Zonen 32 und 33.

Durch die Verwendung von Koordinatensystemen lassen sich Punkte auf der Erdoberfläche in einem rechtwinkligem Gitter markieren. Dies erleichtert, im Gegensatz zu den geografischen Koordinaten, das Eintragen und Ausmessen von Positionen.

2.2.3. Topologie

Topologie beschreibt die Beziehung von geometrischen Informationen zueinander. Diese ist - unabhängig vom Koordinatensystem - immer gleich. Folgende Beispiele sind topologische Fakten:

- Würzburg liegt in Unterfranken
- Unterfranken liegt in Bayern
- Hessen grenzt an Unterfranken
- Der Main mündet in den Rhein (ist mit ihm verbunden)
- Berlin liegt nicht in Bayern

Topologie spielt in GIS eine große Rolle. Viele Fragen lassen sich nur mit Hilfe dieser Beziehungen beantworten. Routenplanung wäre zum Beispiel unmöglich, wenn das System nicht erkennen könnte, welche Straßen miteinander verbunden sind.

2.2.4. Einsatz von Geoinformationen und Statistik in der Epidemiologie

Für die Epidemiologie sind Ort und Zeit neben den Daten zu dem Krankheitserreger und der Person (Alter, Geschlecht, etc.) die wichtigsten Zusatzinformationen zur

Aufdeckung von Gesundheitsproblemen in der Bevölkerung.

Chung et al. untersuchten, inwieweit GI-Systeme in Verbindung mit statistischer Datenanalyse in Studien der Epidemiologie und des öffentlichen Gesundheitswesens Anwendung finden [51]. Die Hauptaufgabe der Geoinformationssysteme ist dabei die Visualisierung der Ergebnisse. Die statistische Analyse führen in den betrachteten Beispielen oft eigenständige, außerhalb des GI-Systems stehenden Werkzeuge durch.

Räumliche Analysen

Die räumliche Analyse nutzt zur Bewertung von Gesundheitsrisiken Standard-GIS-Operationen. Diese führen unter anderem auf der Topologie basierende Abfragen durch. Der Anwender muss das Ergebnis selbst bewerten.

Eine bei Chung et al. genannte Studie analysiert beispielsweise mittels einer auf Verkehrswege spezialisierten GIS-Software die Zeiten für Krankentransporte. Sie errechnet mittels GIS den schnellsten Weg von allen Städten zum jeweils nächsten Krankenhaus. Anhand dieser Zeiten schätzt eine separate Statistiksoftware u. a. die Wahrscheinlichkeit für das Versterben eines Patienten im Krankenhaus ab.

Visualisierung von räumlicher Unsicherheit

Insbesondere bei der Darstellung von Punktdaten in einem GIS gehen bestimmte Informationen verloren. Durch die Auflösung der Datenerhebung (Postleitzahlen) wird beispielsweise die Position eines Erkrankungsfalles genau auf eine Koordinate diskretisiert.

So genannte “kernel-density maps” erlauben eine bessere Darstellung dieser diskreten Daten, indem Punktinformationen in unregelmäßige Flächen umgewandelt werden. Deinen Form und Einfärbung wird durch die Nachbarschaften und Dichte der Punkte in einem Bereich beeinflusst (siehe auch Abbildung 7.1 auf Seite 75).

Diese Art der Darstellung zieht damit die Unsicherheit mit in Betracht, dass ein Erkrankungsfall beispielsweise in einer Stadt nicht genau an der geografischen Länge und Breite deren Zentrums aufgetreten ist. Norström et al. visualisieren mittels dieser “kernel-density maps” die Entwicklung eines Erkrankungsausbruchs unter norwegischen Rinderherden [71]. Die Karten von aufeinander folgenden Zeiträumen zeigen dabei sehr anschaulich, wie sich das Zentrum der Krankheit über administrative Grenzen hinweg “bewegt”.

Clusteridentifizierung mit der räumlichen “Scan Statistic” nach Kulldorff

Die Methode der räumlichen “Scan Statistic” von Kulldorff [62] kann feststellen, ob Häufungen von punktuellen Ereignissen durch Zufall verursacht sind oder nicht.

Dazu stellt sie die maximale Anzahl von Ereignissen fest, die in einem über die Karte bewegten räumlichen Fenster variabler, nach oben begrenzter, Größe liegen. Das in der Epidemiologie eingesetzte kreisförmige Fenster nimmt dabei nur Positionen ein, die

2. Grundlagen

vorher definiert wurden und von denen die Bevölkerungsdaten bekannt sind. Das sind beispielsweise die Zentren von Landkreisen.

Für die Fallverteilung im Fenster mit der maximalen Anzahl wird die Nullhypothese aufgestellt, dass sie einer bestimmten Wahrscheinlichkeitsverteilung folgt (Monte-Carlo-Methode). Der aus mehreren Wiederholungen dieses Versuchs hervorgehende P-Wert drückt die Wahrscheinlichkeit aus, dass die beobachtete Verteilung von Ereignissen reiner Zufall war.

Kann beispielsweise bei 999 Wiederholungen nur einmal die vorgefundene geografische Konstellation von Erkrankungsfällen mittels einer Poisson-Verteilung angenähert werden, beträgt der P-Wert 0,001. Die beobachteten Ereignisse sind also nur mit einer Wahrscheinlichkeit von einem Promille durch Zufall entstanden. Die Methode von Kulldorff hat hier einen sehr wahrscheinlichen räumlichen Cluster entdeckt.

Clusteridentifizierung mit der spatio-temporalen “Scan Statistic” nach Kulldorff

Die Epidemiologie nutzt zur Identifizierung von Clustern oft nicht nur die räumliche, sondern auch die zeitliche Dimension. Kulldorff hat auch für diese Aufgabe eine statistische Methode entwickelt [64].

Das Grundprinzip gleicht dem der räumlichen “Scan Statistic”. Allerdings handelt es sich bei dem bewegten Fenster nicht um einen zweidimensionalen Kreis, sondern um einen Zylinder. Die dritte Dimension drückt dabei aus, welche Zeit zwischen zwei Ereignissen liegt. Durch Kombination aller möglichen Kreisradien und Zylinderhöhen stellt die Methode fest, ob hier eine Konstellation aufgetreten ist, die mit bestimmter Wahrscheinlichkeit kein Zufall sein kann.

Ein erfolgreicher Fund bezeichnet dann einen spatio-temporalen Cluster. Dieser hat eine bestimmte Ausdehnung (Radius) und ein Start- und Enddatum.

3. Definition der Anforderungen

Das vorliegende Kapitel gibt das mit dem Nationalen Referenzzentrum für Meningokokken und der Firma Ridom GmbH abgestimmte Pflichtenheft wieder. Es behandelt alle Kriterien, die das Softwaresystem erfüllen muss, die Einsatzbereiche, Daten, Funktionen und weitere Anforderungen. Das System ist während der Entwicklung auf zwei getrennte Rechner aufgeteilt. Diese werden im Folgenden als **Daten-/Kartenserver** und **Webserver** unterschieden.

Der Daten- und Kartenserver speichert alle Daten und führt die GIS-spezifischen Aufgaben, wie z.B. die Kartenerzeugung aus. Der Webserver liefert die Website und damit vorrangig den Kartenbetrachter an den Benutzer.

3.1. Zielbestimmungen

3.1.1. Musskriterien

Datenbankschnittstelle zur Entgegennahme von Meldedaten

Die von den Laboratorien erzeugten Daten müssen auf den Daten-Server übertragen werden. Das Labor schickt die Typisierungsdaten in einem vereinbarten Format an den Server. Auf diesem läuft eine Anwendung, die als Schnittstelle zu dem Labor dient. Sie stimmt die ankommenden Daten auf das einheitliche interne Datenbankformat ab. Hierdurch werden die Daten frühzeitig in eine normierte Form gebracht, die es erlaubt, das Programm einfach an andere Laboratorien oder Krankheitserreger anzupassen.

Webserver mit JSP

Die Website, die dem Benutzer die Daten präsentiert, wird dynamisch mit JavaServer Pages (JSP) generiert. Das Interface sollte leicht zu erfassen sein.

Datenschutz auf dem Webserver

Aus Gründen des Datenschutzes muss auf dem Webserver eine Zugriffsregelung möglich sein. Von Seiten des RKI bedeutet Datenschutz, dass es nicht möglich sein darf, weniger als vier komplett typisierte Erkrankungen zu einem Landkreis zuordnen zu können. Die öffentlich zugänglichen Bereiche müssen so ausgelegt sein, dass diese Vorgabe immer erfüllt ist.

3. Definition der Anforderungen

Dynamische serverseitige Erzeugung von Karten

Auf dem Kartenserver läuft eine Anwendung, die auf Anfrage des Webservers dynamisch epidemiologische Karten in Form von Bilddateien generiert. Das soll für verschiedene Auflösungen (z.B. Landkreis, Bundesland etc.) geschehen. Die zugrunde liegenden geografischen Daten sind in der Datenbank gespeichert.

Statistische Analyse

Zur Clusterdetektion muss auf dem Datenserver eine statistische Analyse der gemeldeten Erkrankungsfälle durchgeführt werden. Das Labor muss grundlegende Parameter dieser Analyse anpassen können.

3.1.2. Wunschkriterien

Keine festgelegten Auflösungsebenen

Um das Produkt so universell wie möglich zu halten, wäre es wünschenswert, die unterschiedlichen Kartenauflösungen nicht an speziellen Informationen wie z.B. Kreisnummern oder Bundesländern zu verankern. Bei einem Einsatz in Kliniken sind z.B. kleinere Gebiete wie Gebäude und Abteilungen relevant. Daher ist es besser, mit Koordinaten und universellen Hierarchien zu arbeiten.

3.1.3. Abgrenzungskriterien

Bei der Software handelt es sich um ein System zur Entscheidungshilfe. Sie hat nicht zum Ziel, eine künstliche Intelligenz zu integrieren, die eigenmächtig die Signifikanz der erfassten Daten bewertet. Das Hauptziel liegt vielmehr in der Kombination aller nötigen Komponenten zu einem homogenen und benutzbaren Gesamtsystem.

3.2. Produkt-Einsatz

3.2.1. Anwendungsbereiche

Der Anwender möchte mit dem Produkt eine automatisierte Analyse der spatio-temporalen Daten durchführen. Die einzige Anforderung an die Daten ist daher, dass sie eine zeitliche und örtliche Komponente haben und sich verschiedene Ereignisse unterscheiden lassen. Ein Webserver soll die Ergebnisse der Analysen einer breit gefächerten Personengruppe zugänglich machen.

3.2.2. Zielgruppen

“Käufer”-Zielgruppen des Systems sind auf nationaler Ebene Referenzzentren (speziell das NRZM), Landes- bzw. Bundesgesundheitsbehörden (z.B. RKI), Krankenhäuser oder Militär. Auch supranationale Behörden wie das ECDC¹ gehören zu den potenziellen Zielgruppen. Das System ist daher gänzlich in Englisch zu realisieren.

Benutzer-Zielgruppen sind die interessierte Öffentlichkeit, (politische) Entscheidungsträger und die Vertreter des öffentlichen Gesundheitsdienstes (Gesundheitsämter und RKI).

3.2.3. Betriebsbedingungen

Das Produkt wird komplett serverseitig realisiert und daher nur über die Weboberfläche bedient. Diese soll rund um die Uhr erreichbar sein.

3.3. Produktfunktionen

/F10/ Datenbankmodell zur Speicherung epidemiologischer Daten

Es wird ein Datenbankmodell benötigt, welches so generisch wie möglich gestaltet ist. Ziel hiervon ist es, das Produkt mit möglichst geringen Anpassungen auch für die epidemiologischen Daten anderer Erkrankungen nutzen zu können.

/F11/ Entgegennahme von Typisierungsergebnisdaten

Die vom Benutzer erzeugten Typisierungsergebnisdaten müssen über eine Schnittstelle entgegen genommen werden. Hierbei ist darauf zu achten, dass ein proprietäres Datenbankformat wie beispielsweise Access auf die interne PostgreSQL Datenbank abgebildet wird. Das NRZM überträgt die Daten mittels einer CSV² Datei. Unter Linux lassen sich Access Datenquellen nicht direkt nutzen.

/F12/ Datenschutzaspekte

Neben dem sicheren Transfer der Daten zum Server muss auch gewährleistet sein, dass die Datenbank vor unberechtigtem Zugriff ausreichend geschützt ist.

/F20/ Universeller GIS-Service für epidemiologische Daten

Auf Grundlage der Typisierungsergebnisse wird ein GIS-Service entwickelt, welcher die epidemiologischen Daten (Person, Ort, Zeit, (Sub)Typen) auf Landkarten visualisiert. Hierbei ist es wichtig, dass Karten mit verschiedenen Detailgraden (5-stellige PLZ, Kreis, Bundesland, Land) angezeigt werden können.

/F21/ Webservice zur Präsentation des Kartenmaterials

¹European Centre for Disease Prevention and Control

²Character Separated Values bzw. Comma Separated Values

3. Definition der Anforderungen

Eine Flash-Anwendung präsentiert den Benutzern die erzeugten Landkarten. Sie muss klar gestaltet sein, damit die Informationen leicht zugänglich sind.

/F22/ Kartenarten

Es werden verschiedene Arten von Karten benötigt. Um die Inzidenz für einzelne Bereiche darzustellen, sollen diese eingefärbt werden. Für Punktereignisse (Einzelfälle) werden verschiedene Symbole (z.B. Kreis, Dreieck, Stern) in unterschiedlichen Farben verwendet.

Es wäre wünschenswert, Punktinformationen ähnlich wie bei der Arbeitsgemeinschaft Influenza [5] darzustellen.

/F30/ Durchführung der spatio-temporalen Analyse mit SaTScan

SaTScan wird als Werkzeug aus Java heraus aufgerufen. Es benötigt verschiedene Dateien als Eingabe und schreibt die Ergebnisse in Ausgabedateien. Das Dateiformat ist vorgegeben.

/F31/ Erstellung von Falldaten zur Übergabe an SaTScan

Über Abfragen aus der Datenbank werden mittels Java die von SaTScan benötigten Falldateien erstellt. Bei diesen handelt es sich um Textdateien, welche folgende Informationen über aufgetretene Fälle enthalten:

- *location id*
eine eindeutige Zahl oder zusammenhängende Zeichenkette, die einen Ort kennzeichnet
- *#cases*
die Anzahl der Fälle, welche zur angegebenen Zeit an diesem Ort aufgetreten sind
- *time (optional)*
eine Zeitangabe in Jahren, Monaten oder Tagen
- *covariates (optional)*
beliebige Menge an Zusatzwerten, welche die erfassten Fälle näher spezifizieren (z.B. männlich oder weiblich)

Bei den Meningokokken-Daten des NRZM muss für jeden Feintyp eine Falldatei erstellt werden. SaTScan bearbeitet jede dieser Dateien einzeln und erzeugt jeweils Ergebnisdateien.

/F32/ Integration der SaTScan Ergebnisse in die Datenbank

Die Ergebnisdateien enthalten verschiedene berechnete Werte. Sie müssen eingelesen und in die Datenbank übertragen werden. Ein mögliches Ausgabeformat von SaTScan sind durch Leerzeichen getrennte Werte, dieses ist am besten für die Weiterverarbeitung geeignet.

Listing D.1 auf Seite 97 zeigt an einem Beispiel das kommentierte Format, anhand dessen ersichtlich ist, welche Werte von SaTScan berechnet werden.

/F33/W Konfiguration der SaTScan Parameter über eine Webschnittstelle

In der Parameterdatei von SaTScan lassen sich alle Optionen der Analyse anpassen. Die wichtigsten Parameter sollten sich über ein Web-Interface verändern lassen.

/F40/ Ausgabe der Ergebnisse der spatio-temporalen Analyse

Die Ergebnisse der Analysen von SaTScan werden mit Hilfe der GIS-Komponente auf Landkarten angezeigt. Insbesondere muss es möglich sein, die Position und geografische Ausdehnung von Clustern zu erkennen und über eine Zeitspanne zu verfolgen.

/F41/W Animationen zur Darstellung von zeitlichen Verläufen

Eine mögliche Art der Präsentation von zeitlichen Entwicklungen der Fallzahlen und Clusterpositionen sind Animationen. Diese könnten direkt in Flash dargestellt werden.

/F50/ Errechnung und geografische Visualisierung von Inzidenzen

Inzidenzen haben eine besondere Bedeutung bei der epidemiologischen Beobachtung. Bei dieser Analyse sind größere geografische Regionen interessant (in Deutschland z.B. Landkreise und Bundesländer). Die Inzidenz soll dynamisch anhand der vorhandenen Fälle errechnet werden.

/F60/W Dynamische Durchführung anderer, von SaTScan unabhängigen, Abfragen

Neben dem Kartenmaterial sollten auch herkömmliche Statistiken in Form von Kreuztabellen auf dem Webserver präsentiert werden. Die folgenden Abfragen werden vom NRZM bisher durchgeführt:

- Feintypverteilung in Deutschland bzw. einzelnen Bundesländern
- Serogruppenverteilung nach Bundesländern bzw. Alter
- Häufigste Feintypen nach Alter
- Inzidenztabellen

/F70/ Alle Daten werden fortschreitend jährlich zusammengefasst

Die SaTScan Analyse wird einmal wöchentlich ausgeführt. Sie bezieht rückblickend vom aktuellen Datum alle Erkrankungsfälle der letzten 100 Tage mit ein. Zusätzlich erfolgt für jedes halbe Jahr eine "historische" Berechnung.

3.4. Produkt-Daten

3.4.1. Eingabedaten

/D10/ epidemiologische Daten

Die epidemiologischen Daten (Person, Ort, Zeit, (Sub)Typen) und weitere untergliederte Werte (Alter, Geschlecht, Koordinate, Erkrankungsbeginn, Erreger-Isolationsdatum) müssen in der Datenbank abgelegt und für statistische Analysen zugänglich sein.

3. Definition der Anforderungen

/D20/ Populationsdaten

Die Anwendung benötigt Populationsdaten, um die Inzidenzen in /F50/ zu berechnen und spatio-temporale Analysen durchführen zu können.

/D30/ räumliche Daten

PostgreSQL bietet die Möglichkeit, alle grundlegenden geografischen Datenformen (siehe Abschnitt 2.2.1) zu speichern.

3.4.2. Ausgabedaten

/D40/ verschiedene Karten

Das System gibt Karten in unterschiedlicher geografischer Auflösung (z.B. Kreis, Bundesland etc.) aus. Epidemiologische Daten sollen in verschiedener Weise visualisiert werden (siehe /F22/). Bei den Karten sind ggf. die Lizenzbestimmungen der Kartenhersteller zu berücksichtigen.

/D50/ errechnete Inzidenzen

/D60/ alle Ergebnisse aus der statistischen Analyse von SaTScan

3.5. Produkt-Leistungen

/L10/ wöchentliche Batch-Prozessierungen der spatio-temporalen Analysen

/L20/ Die Ausgabe von dynamischen Abfrageergebnissen bzw. Landkarten auf dem Webserver sollte so schnell wie möglich erfolgen.

3.6. Qualitätsanforderungen

Tabelle 3.1 definiert die Anforderungen an die Produktqualität in den Kategorien Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit.

Produktqualität	sehr gut	gut	normal	irrelevant
Funktionalität		X		
Zuverlässigkeit			X	
Benutzbarkeit		X		
Effizienz			X	
Änderbarkeit	X			
Übertragbarkeit	X			

Tabelle 3.1.: Geforderte Produktqualität nach Kategorien

3.7. Technische Produktumgebung

Dieser Abschnitt beschreibt die technischen Voraussetzungen zum Einsatz des Softwaresystems EpiDeGIS. Wie in diesem Kapitel eingangs erwähnt, wird hier zwischen dem Daten-/Kartenserver und Webserver unterschieden.

3.7.1. Hardware-Umgebung

Es ist schwierig, die Anforderungen an die Hardware für die jeweiligen Server in konkreten Zahlen zu bestimmen. Gerade im Serverbetrieb hat die Anzahl der simultanen Anfragen einen entscheidenden Einfluss auf die benötigte Leistung bzw. Speicherausstattung. Die geografischen Abfragen und die Bilderzeugung auf dem Kartenserver, erfordern grundsätzlich eine höhere Rechenleistung als die Auslieferung der Daten durch den Webserver.

Die nachfolgende Übersicht stellt eine mögliche Konfiguration dar:

Daten- und Kartenserver

- Minimales System
 - Prozessor der Intel Pentium/Celeron-Produktfamilie, der AMD Athlon/Duron-Produktfamilie oder kompatibel mit 2 GHz
 - 512 MB Arbeitsspeicher
- Empfohlenes System
 - Prozessor der Intel Pentium/Celeron-Produktfamilie, der AMD Athlon/Duron-Produktfamilie oder kompatibel mit 3 GHz
 - 1 GB Arbeitsspeicher

Webserver

- Minimales System
 - Prozessor der Intel Pentium/Celeron-Produktfamilie, der AMD Athlon/Duron-Produktfamilie oder kompatibel mit 800 MHz
 - 256 MB Arbeitsspeicher
- Empfohlenes System
 - Prozessor der Intel Pentium/Celeron-Produktfamilie, der AMD Athlon/Duron-Produktfamilie oder kompatibel mit 2 GHz
 - 512 MB Arbeitsspeicher

3. Definition der Anforderungen

3.7.2. Software-Umgebung

Die Software-Umgebung bezeichnet alle auf dem jeweiligen Server benötigten Programme. Der folgend Abschnitt listet diese auf.

Grundvoraussetzungen

Beide Server benötigen folgende Software:

- Betriebssystem Linux

Linux bezeichnet den freien Kern eines Betriebssystems. Es wurde ursprünglich von Linus Torvalds entwickelt und bildet heute die Grundlage für mehrere Distributionen. Die als Basissystem für EpiDeGIS eingesetzte Linux-Distribution muss als einzige Voraussetzung zulassen, dass sich alle benötigten Softwarekomponenten installieren lassen.

- Java Runtime Environment

Der Tomcat Webserver [4] und alle Werkzeugprogramme von EpiDeGIS sind in Java implementiert. Aus diesem Grund ist ein installiertes Java Runtime Environment, Version 5.0 [16] Grundvoraussetzung für die Serversysteme und auch den Client, der die Daten überträgt.

Daten- und Kartenserver

Für die Datenbank und Kartenerzeugung müssen folgende Programme installiert sein.

- GNU C Compiler (*gcc*) und GNU make (*gmake* oder *make*)
- GEOS geometrische Bibliothek [73]
- PROJ.4 Kartenprojektions-Bibliothek [56]
- PostgreSQL (Version 8.x) [31]
- PostGIS für PostgreSQL [78]
- UMN MapServer (Version 4.6x) [68]
- SaTScan (Die Entwickler verschicken eine Internetadresse zum Download der Linux-Version auf Anfrage.)

Webserver

EpiDeGIS verwendet den Apache Tomcat Servlet Container in der Version 5.x als Webserver.

3.8. Entwicklungsumgebung

3.8.1. Programmiersprache

Alle Softwarekomponenten für diese Diplomarbeit sind in Java realisiert. Diese von Sun entwickelte Programmiersprache ist mittlerweile in der Version 5.0 verfügbar. Die Apache Software Foundation [40] fördert mit vielen freien Projekten die Verwendung von Java bei der Softwareentwicklung. So stellt z.B. die durch das Jakarta Projekt [39] entwickelte Commons-Bibliothek [15] eine Vielzahl von Java-Standardlösungen bereit [81]. Sie werden in diesem Projekt unter anderem für den Datenbankzugriff und die Ein-/Ausgabe eingesetzt.

3.8.2. Entwicklungswerkzeuge

Integrierte Entwicklungsumgebung

Als integrierte Entwicklungsumgebung hat sich für Java das frei erhältliche Programm Eclipse [9] durchgesetzt. Dieses Plugin-basierte Softwareentwicklungs-Framework besteht auch den Vergleich mit kommerziellen Produkten. EpiDeGIS ist mit Hilfe von Eclipse entwickelt. Dank eines vorgeschalteten Projektverwaltungssystems können die Quellen jedoch auch mit allen anderen integrierten Entwicklungsumgebungen bearbeitet werden.

Projektverwaltung

Eine Projektverwaltung vereinfacht den Entwicklungsprozess im Hinblick auf die Organisation des Quellcodes. Die Software Maven [2] ist eine Projektverwaltung für Java-Entwickler. Sie ist ebenfalls aus dem Jakarta-Projekt hervorgegangen und erleichtert den Entwicklungsprozess in mehreren Bereichen [12]:

- Der Entwicklungsorgang (kompilieren, testen, verteilen) ist standardisiert. Maven verbirgt unnötige Details.
- Alle Software-Projekte sind einheitlich organisiert. Ein Entwickler findet sich in unbekannten Projekten schneller zurecht.
- Das System erzeugt ausführliche Projektinformationen (Changelog, Javadoc, Testergebnisse, Abhängigkeiten, Metriken, etc.).
- Maven bietet dem Entwickler eine Hilfestellung für das Einhalten von bewährten Praktiken
- Externe Abhängigkeiten (Bibliotheken) lassen sich gut organisieren.

Maven ist unabhängig von einer zusätzlich verwendeten Entwicklungsumgebung. Der Entwickler kann zum Beispiel die Projektdefinitionsdatei für Eclipse mit dem Befehl `maven eclipse` erzeugen lassen. Selbst umfangreiche Projekte mit vielen Abhängigkeiten lassen sich durch einen Befehl sofort in Eclipse öffnen und bearbeiten.

3. Definition der Anforderungen

Die Projektverwaltung unterstützt den Anwender auch bei der Versionsverwaltung. Dateien, die ein Projekt beschreiben, liegen im XML-Format vor. Der Entwickler kann dadurch die Änderungen im Projekt über verschiedenen Versionen hinweg leicht verfolgen.

4. Untersuchung der Voraussetzungen

Dieses Kapitel dokumentiert die der Softwareentwicklung vorausgegangenen Vorgänge. Es präsentiert Beispiele von bereits über das Internet zugänglichen Systemen zur Infektionsüberwachung.

Weiterhin definiert es die grundlegenden Anforderungen für das zu entwickelnde Web-basierte epidemiologische GIS mit integrierter Ausbruchsanalyse. Dieses stellt eine neuartige Kombination von Einzelkomponenten dar, deren Voraussetzungen näher betrachtet werden.

Der letzte Teil dieses Kapitels gibt einen Überblick über verfügbare Open Source Software, die sich zur Lösung der gegebenen Problemstellung eignet. Jedes der in Betracht gezogenen freien Produkte wird vorgestellt und hinsichtlich der Verwendbarkeit bewertet.

4.1. Web-basierte Systeme zur Überwachung von Infektionen

Die Epidemiologen haben dank des Internets die Möglichkeit, die Ergebnisse ihrer Arbeit einer größeren Benutzerzahl zur Verfügung zu stellen. Abbildung 4.1 veranschaulicht diese Möglichkeit am Beispiel des NRZM.

Ein Web-basiertes Informationssystem hat neben der einfachen Zugänglichkeit über ein Netzwerk weitere Vorteile:

- Der Benutzer kann die angezeigten Informationen nach eigenen Kriterien filtern.
- Das System kann sehr aktuell sein.
- Das Internetangebot übernimmt einen Teil der Öffentlichkeitsarbeit durch Bereitstellung von Informationen.

Im weltweiten Datennetz finden sich viele solcher epidemiologischer Informationsportale. Die oft auf Landesebene eingesetzten Webangebote sind von unterschiedlicher Komplexität. Manche bieten nur eine Zusammenfassung von Statistiken in Jahresberichten an, andere präsentieren Daten interaktiv in Form von Zahlen und Diagrammen. Fortgeschrittene Systeme nutzen auch Karten zur Präsentation der geografischen Verteilung von Erregern. Drei dieser Systeme werden im Folgenden vorgestellt.

4. Untersuchung der Voraussetzungen

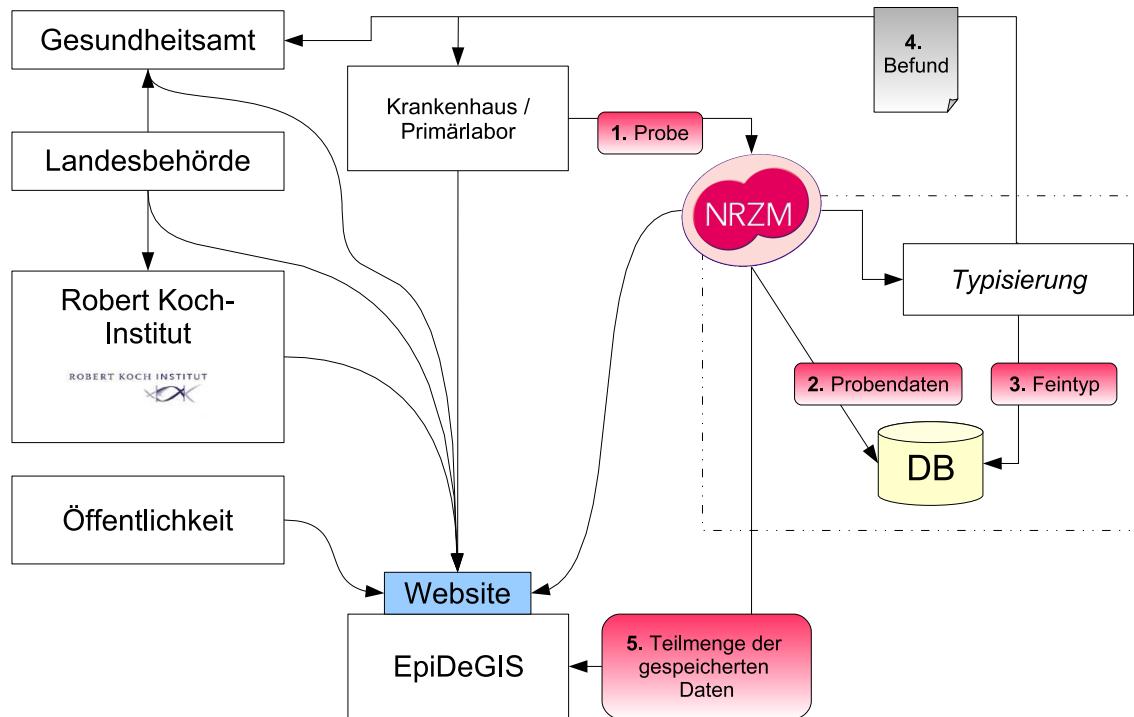


Abbildung 4.1.: Neuer Informationsfluss, ermöglicht durch den Einsatz eines Web-basierten Systems (am Beispiel des NRZM, vgl. Abbildung 2.2 auf Seite 10)

SurvStat vom Robert Koch-Institut [38] Dieses System erlaubt den Zugriff auf die vom RKI erfassten Krankheitsfälle und Erreger nachweise. Es ergänzt die bisher in Form des “Epidemiologischen Bulletins” und des “Infektionsepidemiologischen Jahrbuchs” vom Institut angebotenen Informationen. Der Anwender des Systems kann über die Benutzeroberfläche verschiedene Parameter einstellen (siehe Abbildung 4.2, linke Seite). SurvStat erlaubt das Filtern der Daten nach spezifischen Krankheiten bzw. Erregern, Zeiträumen, Orten, Personendaten und Falldefinitionen. Die Abfrageergebnisse lassen sich anhand eines unter diesen Kategorien verfügbaren Merkmals (Geschlecht, Altersgruppe, Quartal, Landkreis, etc.) gruppieren. Weiterhin kann der Anwender ein zusätzliches zweites Merkmal für eine kreuztabellarische Abfrage angeben.

In dem Kreisdiagramm auf der rechten Seite von Abbildung 4.2 sind die Meningokokkenfälle im Jahr 2005 gruppiert nach den Serogruppen B, C und Y dargestellt.

SurvStat ist ein sehr umfangreiches System. Es kann auf den vereinfachten Datenbestand des RKI zugreifen. Die Daten sind sehr aktuell (wöchentliche Aktualisierung). Ein Nachteil ist die clientseitige Umsetzung mittels JavaScript und HTML. Nahezu jeder Mausklick führt zu einer kompletten Aktualisierung der Website. Zudem zeigt eine durchgeführte Anfrage das Ergebnis in einem separaten Browser-Fenster an. Diese beiden Faktoren hemmen den Arbeitsfluss.

Arbeitsgemeinschaft Influenza [5] Die Arbeitsgemeinschaft Influenza (AGI) informiert die Bevölkerung während der Wintersaison über die Aktivität der auch unter dem Namen “Grippe” bekannten Influenza. Dieses Projekt ermöglicht es dem Anwender nicht, Filter einzustellen, um ausführliche Tabellen und Diagramme zu erzeugen.

4.1. Web-basierte Systeme zur Überwachung von Infektionen

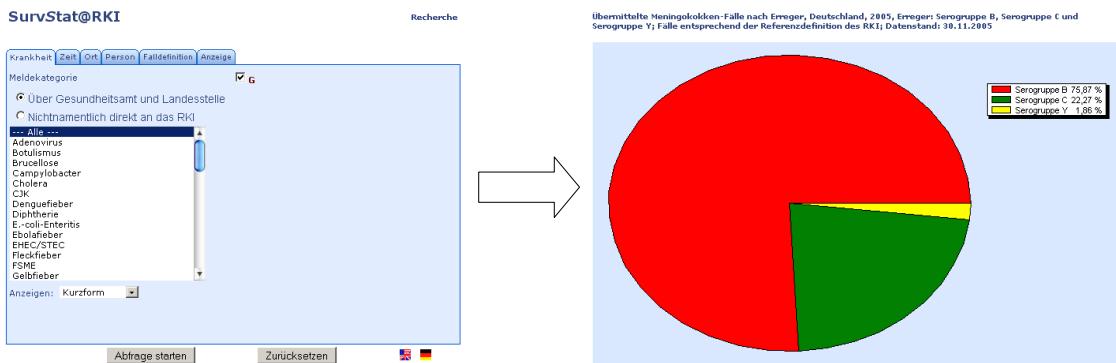


Abbildung 4.2.: Benutzeroberfläche von SurvStat. Das System erzeugt anhand der in der Maske eingestellten Parameter unter anderem Kreisdiagramme. (Bildschirmfoto, Stand: 05. Dezember 2005)

Zudem beschränkt es sich im Vergleich zu SurvStat auf lediglich einen Erreger. Der Grund dafür, dass es trotzdem hier als Beispiel angebracht wird, ist die gute Präsentation der aktuellen Erkrankungssituation.

Die AGI verwendet in ihrem Internetangebot übersichtliche Karten (Abbildung 4.3), um die Verteilung von Erkrankungsfällen in Deutschland intuitiv zu vermitteln. Sie veranschaulicht zudem die Dynamik der Erregeraktivität durch Animationen mit einem Bild pro Kalenderwoche. Der Benutzer kann neben dieser Darstellung noch eine vorgegebene Menge von Diagrammen anfordern. Diese zeigen die zeitliche Verteilung verschiedener Messwerte für das Bundesgebiet und die Länder. Das Angebot der AGI bietet keine interaktiv änderbaren Karten oder Diagramme.

Swedish Institute for Infectious Disease Control (SMI) [35] Das SMI ist eine schwedische Behörde, die mit der Epidemiologie von Infektionskrankheiten betraut ist. Das Institut bietet auf seiner Website ausführliche Statistiken zu meldepflichtigen Erkrankungsfällen in Schweden. Die Menge der unterschiedlichen erfassten Krankheitserreger ist mit der von SurvStat vergleichbar. Rolfhamre et al. [77] beschreiben den Web-basierten Dienst, welcher der Öffentlichkeit seit Mai 2003 Zugriff auf die Daten des SMI ermöglicht. Das System präsentiert die Statistik zu den erfassten meldepflichtigen Krankheiten in verschiedener Weise:

- Tabellen zeigen die Anzahl der Fälle und Inzidenz nach Kreisen als Monats- und Jahresstatistik und die Altersverteilung nach Gruppen, oder die Geschlechterverteilung als Jahresstatistik an.
- Grafen informieren über die gesamte Anzahl der Fälle pro Woche und deren Trend.
- Karten veranschaulichen die geografische Verteilung der Erkrankungszahlen und die Inzidenz anhand der Einfärbung von Kreisen (Abbildung 4.4).

Die schwedischen Wissenschaftler haben mit diesem Onlineangebot eine Möglichkeit geschaffen, Interessenten mit Hilfe von GIS online über die epidemiologische Situation in ihrem Land zu informieren. Das Angebot an Statistiken ist, die Anzahl der erfassten

4. Untersuchung der Voraussetzungen

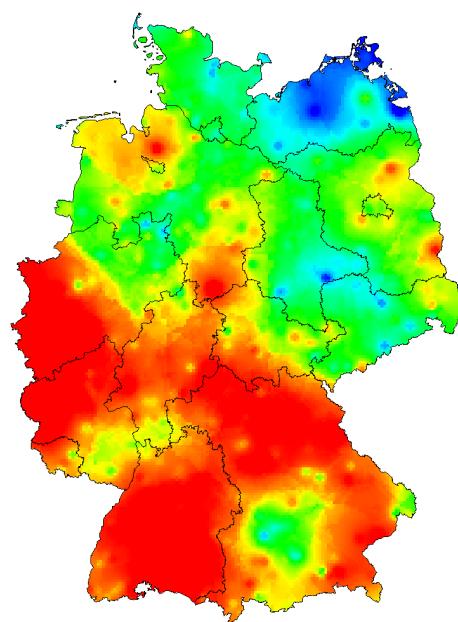


Abbildung 4.3.: Aktivität der Influenza in der 6. Kalenderwoche 2005 (AGI-Website, Stand: 05. Dezember 2005)

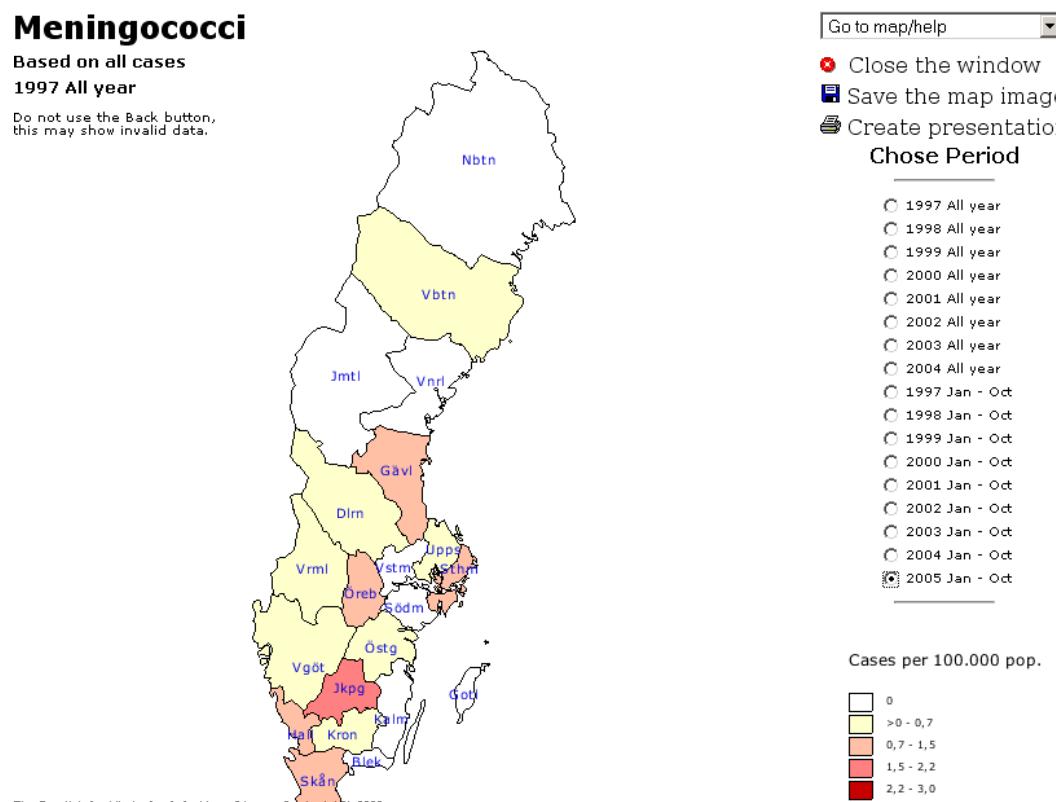


Abbildung 4.4.: Inzidenz der Meningokokken-Erkrankungsfälle in Schweden von Januar bis Oktober 2005 (Bildschirmfoto, Stand: 06. Dezember 2005)

Erkrankungen betreffend, sehr umfangreich. Im Vergleich zu SurvStat behandelt es allerdings alle verschiedenen Krankheitserreger gleich. Unter den Meningokokkenerkrankungen tauchen zum Beispiel keine Serogruppen auf. Die Interaktionsmöglichkeiten des Benutzers beschränken sich auf die Auswahlmöglichkeit bestimmter Jahre bei den angezeigten Karten. Alle anderen Inhalte sind statisch. Die Daten im System werden lediglich einmal im Monat aktualisiert.

4.2. Anforderungen an ein Web-basiertes epidemiologisches GIS

Die im vorigen Abschnitt genannten Beispiele für epidemiologische Informationsplattformen vermitteln bereits einige Anforderungen an das zu entwickelnde System. Gleichzeitig zeigen sie auch, dass öffentlich zugängliche geografische Informationssysteme in diesem Bereich noch am Anfang stehen. Dies erstaunt umso mehr, als Karten bereits für die ersten Menschen ein Mittel waren, Informationen auszutauschen (z.B. Strategien für eine Jagd) [49].

Ein öffentlich zugängliches Informationssystem muss dem Anwender intuitiv Daten und Zusammenhänge vermitteln. Die nachfolgend aufgeführten Einzelkomponenten tragen jeweils zur Erfüllung dieser Aufgabe bei.

4.2.1. Reichhaltige Internetanwendung als Benutzerschnittstelle

Die Beispielsysteme aus Abschnitt 4.1 haben alle den Nachteil, dass sie HTML-Dokumente verwenden, um dem Benutzer Informationen zu präsentieren. Jede Interaktion erfordert eine Anfrage des Browsers an den Server. Es kommt kein richtiger Arbeits- und Informationsfluss zu Stande.

Die Lösung dieses Problems heißt “Rich Internet Applications” (RIAs), wobei “Rich” reichhaltig bedeutet. Der Benutzer lädt RIAs über eine Website direkt aus dem Internet. Im Vergleich zu HTML-Seiten sind sie um Funktionen angereichert, die sonst nur von Desktop-Applikationen bekannt sind. IDC¹ [14] hat in einem White Paper [55] die Verwendbarkeit von RIAs für Firmen bewertet. Für den Einsatz als clientseitige Schnittstelle eines epidemiologischen Geoinformationssystems sprechen mehr Vor- als Nachteile:

Vorteile

- Der Benutzer kann die Client-Anwendung “überall” verwenden.
- RIAs bieten auf verschiedenen Hard- und Softwaresystemen immer die gleiche Funktionalität.
- Die Anwendung wird clientseitig ausgeführt. Einmal geladen, muss sie nur noch wenige Daten mit dem Server austauschen.

¹International Data Corporation

4. Untersuchung der Voraussetzungen

	<i>Version 5</i>	<i>Version 6</i>	<i>Version 7</i>
<i>USA</i>	97,3%	96,4%	91,9%
<i>Kanada</i>	98,7%	97,5%	93,8%
<i>Europa</i>	98,7%	97,4%	93,3%
<i>Asien</i>	96,6%	94,0%	87,7%

Tabelle 4.1.: Ergebnisse einer weltweiten Umfrage über die Verbreitung des Flash-Plugins [23] (Stand: September 2005)

- Die Benutzeroberfläche ist eingängig und erlaubt ein hohes Maß an Interaktivität. Sie „fühlt“ sich wie eine Desktop-Anwendung an.
- Die Medien Text, Bild, Audio und Video lassen sich in der Präsentation nahtlos verbinden.
- Im Gegensatz zu Desktop-Anwendungen sind Updates für den Benutzer sofort verfügbar.
- Der Anwender setzt seinen Rechner durch die Ausführung von RIAs keinem Sicherheitsrisiko aus (“Sandbox”-Prinzip [69]).

Nachteile

- RIAs benötigen für die Ausführung ein im Browser installiertes Plugin. Benutzer von Systemen, auf denen diese Software-Komponente nicht installiert werden kann, haben keine Möglichkeit, die Internetanwendung auszuführen.
- Der initiale Ladevorgang dauert länger als bei einer HTML-Seite.

Die RIA für EpiDeGIS ist mittels Flash umgesetzt. Viele Benutzer haben das Flash-Plugin der Firma Adobe² [1] auf ihren Rechnern installiert (siehe Tabelle 4.1). Diese Daten sprechen für die Verwendung von Flash zur Entwicklung eines erreichbaren Informationssystems.

Alternative Plattformen zur Entwicklung von RIAs sind beispielsweise Java-Applets, “Java Web Start”-Anwendungen [17], SVG³ [34] und XUL⁴ [42]. “Java Web Start” bietet sehr vielseitige Möglichkeiten erfordert jedoch genau wie Applets ein installiertes Java-Plugin. Dieses ist ebenso wie das SVG-Plugin bis dato noch nicht so weit verbreitet wie Flash. Bei XUL-Anwendungen handelt es sich um eine Technik, die aus dem Mozilla-Projekt hervorgegangen ist. Um mit XUL erstellte RIAs lassen sich daher nur in einem Mozilla-Browser ausführen.

Als epidemiologisches GIS muss die Rich Internet Application dem Benutzer folgende Funktionen bieten:

- Anzeigen von verschiedenen Karten mit der Möglichkeit, zwischen diesen direkt umzuschalten

²Die Firma Macromedia, die Flash vormals entwickelt hat, ist am 5. Dezember 2005 von Adobe übernommen worden.

³Scalable Vector Graphics

⁴XML User Interface Language

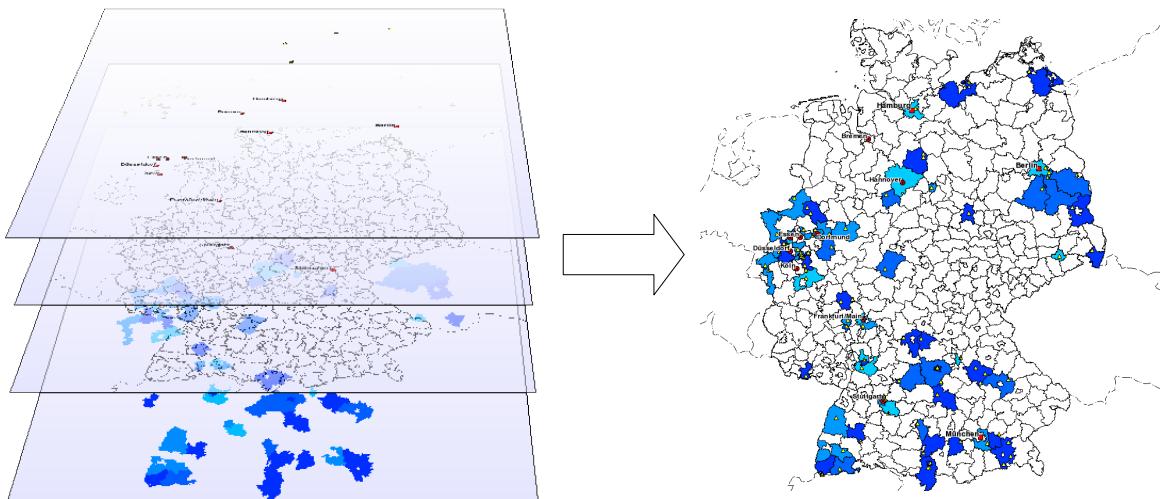


Abbildung 4.5.: Eine Karte wird im GIS aus einzelnen Ebenen aufgebaut

- Verschieben und Vergrößern des gezeigten Kartenausschnitts
- Abfragen von Detailinformationen zu einzelnen Punkten auf der Karte
- Filtern der epidemiologischen Daten nach Altersgruppen, Zeitraum und spezifischen Erreger-Merkmalen (z.B. Serogruppe)
- Ausgeben einer druckfähigen Version der Karte

Weitere Anforderungen an die Benutzeroberfläche sind Geschwindigkeit und intuitive Bedienbarkeit.

4.2.2. Geografisches Informationssystem

Das geografische Informationssystem in diesem Projekt erzeugt verschiedene epidemiologische Übersichtskarten. Das GIS muss dabei mehrere Anforderungen erfüllen.

Karten setzen sich aus Ebenen zusammen Digitale Karten sind in Geoinformationssystemen meistens mittels Ebenen organisiert. Zusammengehörige Daten liegen in einer Ebene. Die Karte in Abbildung 4.5 enthält zum Beispiel die Ebenen Inzidenz, Grenzen, Städte und Erkrankungsfälle. In einer Ebene zusammengefasste Elemente sind immer vom gleichen geometrischen Typ. Bei Grenzen und Inzidenz handelt es sich um Flächen-, bei Städten und Erkrankungsfällen um Punktinformationen. Die einzelnen Schichten erzeugen zusammen das Gesamtbild der Karte. Der Benutzer kann die Darstellung durch Ein- und Ausblenden einzelner Ebenen verändern.

Dynamische Karten Durch die Ebenen hat der Anwender Einfluss auf die angezeigten Informationen. Das GI-System muss zudem auch auf die unter 4.2.1 beschriebenen

4. Untersuchung der Voraussetzungen

Filter reagieren. Einzelne Ebenen besitzen dazu Parameter, anhand derer die Darstellung erzeugt wird. Die Inzidenz-Ebene stellt zum Beispiel die Inzidenz aller im gewählten Zeitraum aufgetretenen Fälle dar. Zusätzlich lässt sich die geografische Auflösung dieser Berechnungen (z.B. Bundesländer oder Kreise) einstellen.

Zugriff auf Attributinformationen Attributinformationen sind alle verfügbaren Daten außer den Koordinaten. Geoinformationssysteme speichern oft viele Zusatzdaten zu einem geografischen Eintrag. Die Einwohnerzahl einer Stadt ist beispielsweise ein Attribut. Verschiedene Farben und Symbole können mögliche Attributwerte repräsentieren. Ergänzend zu dieser Informationsdarstellung muss das GI-System allerdings auch Detailinformationen zu einer bestimmten Position oder Region auf der Karte liefern können. Die Inzidenz lässt sich wie beim schwedischen System in Abschnitt 4.1 mit vier Farben darstellen. Der Benutzer dieses Systems kann aber nur mit den exakten Attributinformationen feststellen, ob die Inzidenz in einem Landkreis 0,71 oder 1,49 beträgt. Der Wertebereich von 0,7 bis 1,5 entspricht einer Farbe und erlaubt daher keine Differenzierung.

4.2.3. Aktualisierung der Daten

Die Aktualität der von einem Informationssystem bereitgestellten Daten ist insbesondere bei der Überwachung von Erkrankungsfällen von großer Bedeutung. Die Epidemiologen müssen in der Lage sein, die im System gespeicherten Falldaten regelmäßig und automatisiert zu aktualisieren, um auf Veränderungen reagieren zu können.

Das Referenzzentrum/Labor überträgt die Daten in einem vereinbarten Transferformat an den Server. Eine sichere Übertragungsleitung gewährleistet den Schutz der sensiblen Informationen. Der Server führt eine Aktualisierung der Datenbank durch.

4.2.4. Automatische Identifizierung von Clustern

Für Epidemiologen ist die Identifizierung von Clustern (siehe auch Abschnitt 2.2.4) ein wichtiges Hilfsmittel. Es lenkt ihre Aufmerksamkeit auf Einzelfälle, die sie genauer untersuchen müssen. EpiDeGIS ist das erste Web-basierte epidemiologische GIS, das diese Analysen automatisiert durchführen kann. Es übernimmt damit die Aufgabe eines Frühwarnsystems.

Das NRZM wendet bei der Clusteridentifizierung zwei verschiedenen Verfahren an. Der Server muss diese mittels SaTScan automatisch, basierend auf den aktuellen Falldaten, ausführen.

Prospektive Analyse Die prospektive Analyse ist ein Frühwarnsystem. Sie kann Ausbrüche von Erkrankungsfällen, die gerade im Entstehen sind, frühzeitig erkennen [64]. Sie reagiert sehr schnell auf Veränderungen. Die Suche nach aktuell vorhandenen ("lebenden") Clustern soll wöchentlich durchgeführt werden. Das NRZM setzt eine maximale Clusterdauer von 60 Tagen fest. Das System soll immer nur das Ergebnis der aktuellsten prospektiven Analyse anzeigen.

Retrospektive Analyse Die Retrospektive Analyse liefert alle in der Vergangenheit aktiven Cluster. Diese historischen Erkrankungshäufungen sind keine akute Bedrohung der öffentlichen Gesundheit. Kulldorff et al. haben mit dieser Analyse z.B. nachgewiesen, dass ein scheinbar verstärktes Auftreten von Erkrankungen an Gehirntumoren in Los Alamos, New Mexico, reiner Zufall war. Sie konnten keinen Zusammenhang mit dem dort 1943 von der US-Regierung aufgebauten Kernforschungszentrum feststellen [63].

Die Retrospektive Analyse soll in diesem System die historische Clusterlandschaft darstellen. Hierzu sollen die Berechnungen monatlich mit allen bisher erfassten Daten durchgeführt werden.

4.3. Anwendungsrelevante freie Softwarepakete

Definition Freie Software bedeutet die Freiheit des Benutzers, die Software zu benutzen, zu kopieren, sie zu vertreiben, zu studieren, zu verändern und zu verbessern. Genauer gesagt, bezieht sich der Begriff “Freie Software” auf vier Arten von Freiheit, die der Benutzer der Software hat:

- Die Freiheit, das Programm für jeden Zweck zu benutzen (Freiheit 0).
- Die Freiheit, zu verstehen, wie das Programm funktioniert und wie man es für seine Ansprüche anpassen kann (Freiheit 1). Der Zugang zum Quellcode ist dafür Voraussetzung.
- Die Freiheit, Kopien weiterzuverbreiten, so dass man seinem Nächsten weiterhelfen kann (Freiheit 2).
- Die Freiheit, das Programm zu verbessern und die Verbesserungen der Öffentlichkeit zur Verfügung zu stellen, damit die ganze Gemeinschaft davon profitieren kann (Freiheit 3). Der Zugang zum Quellcode ist dafür Voraussetzung.”

Free Software Foundation - Die Definition freier Software [8]

Open-Source-Software [27, OSS] und freie Software [11] sind bedeutungsgleich. Die Verfechter der jeweiligen Bewegung setzen allerdings unterschiedliche Schwerpunkte. Diese Arbeit verwendet die beiden Begriffe synonym.

Bei der Entwicklung dieses Projektes war es wichtig, nicht-proprietarye Software-Produkte zu finden, welche die in diesem Kapitel definierten Anforderungen erfüllen. Nicht alle im Folgenden genannten Produkte sind verwendet worden.

4.3.1. Web-basierte Anwendungen

Struts als serverseitiges Framework

Jakarta Struts [3] ist ein Framework zur Entwicklung von auf JavaServer Pages [19, JSP] basierenden dynamischen Internetseiten. Struts implementiert in Teilen Suns “Model-2”-Architektur [80]. Hierbei handelt es sich um eine für Webanwendungen angepasste

4. Untersuchung der Voraussetzungen

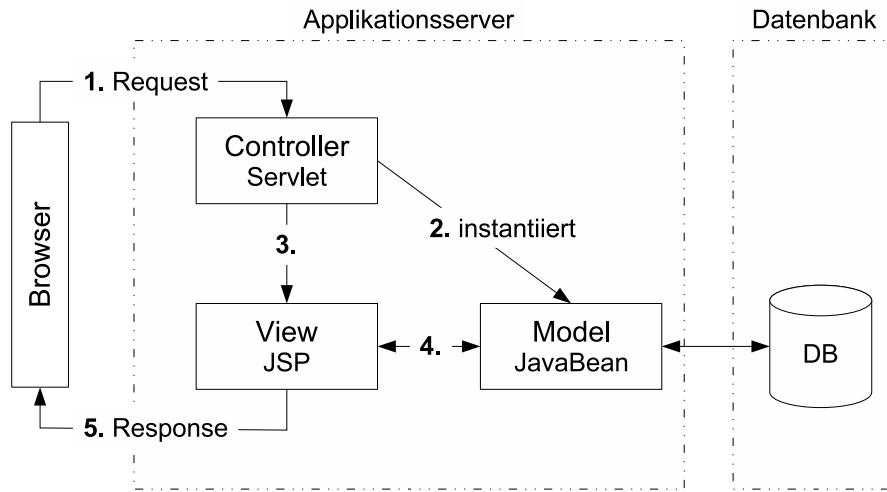


Abbildung 4.6.: "Model-2"-Architektur: Modell-View-Controller für Webanwendungen

Version des Model-View-Controller Architekturmusters [76]. Abbildung 4.6 zeigt das Zusammenspiel der MVC-Komponenten im "Model-2".

Struts bringt bei der Entwicklung von Webanwendungen verschiedene Vorteile [61]:

- Durch eine Implementierung mit wenigen überschaubaren Komponenten ist das Framework leicht zu erlernen. Gleichzeitig bietet es genug Freiraum für experimentierfreudige Entwickler.
- Alle Funktionen von Java Servlets bleiben erhalten. Struts lässt sich mit Bibliotheken kombinieren.
- Informationen lassen sich einfach zwischen Objekten und HTML-Formularen verschieben.
- Struts verwaltet alle Einstellungen in einer zentralen XML-Datei. Sie ist trotz umfangreicher Konfigurationsmöglichkeiten gut verständlich.
- Struts arbeitet gut mit anderen auf Java Servlets beruhenden Frameworks zusammen.

Die zentrale Komponente von Struts ist ein Front-Controller. Dieser ist der zentrale Zugangspunkt für die Webanwendung. Er nimmt alle Anfragen entgegen und erledigt sie selbst oder leitet sie an andere Klassen weiter.

Dieses Projekt verwendet Struts als serverseitiges Framework.

OpenLaszlo für Rich Internet Applications

OpenLaszlo [28] ist eine von Laszlo Systems, Inc. entwickelte Software-Plattform zur Entwicklung und Verbreitung von Rich Internet Applications (siehe Abschnitt 4.2.1). Laszlo Systems stellt der Öffentlichkeit die Software seit Oktober 2004 unter einer

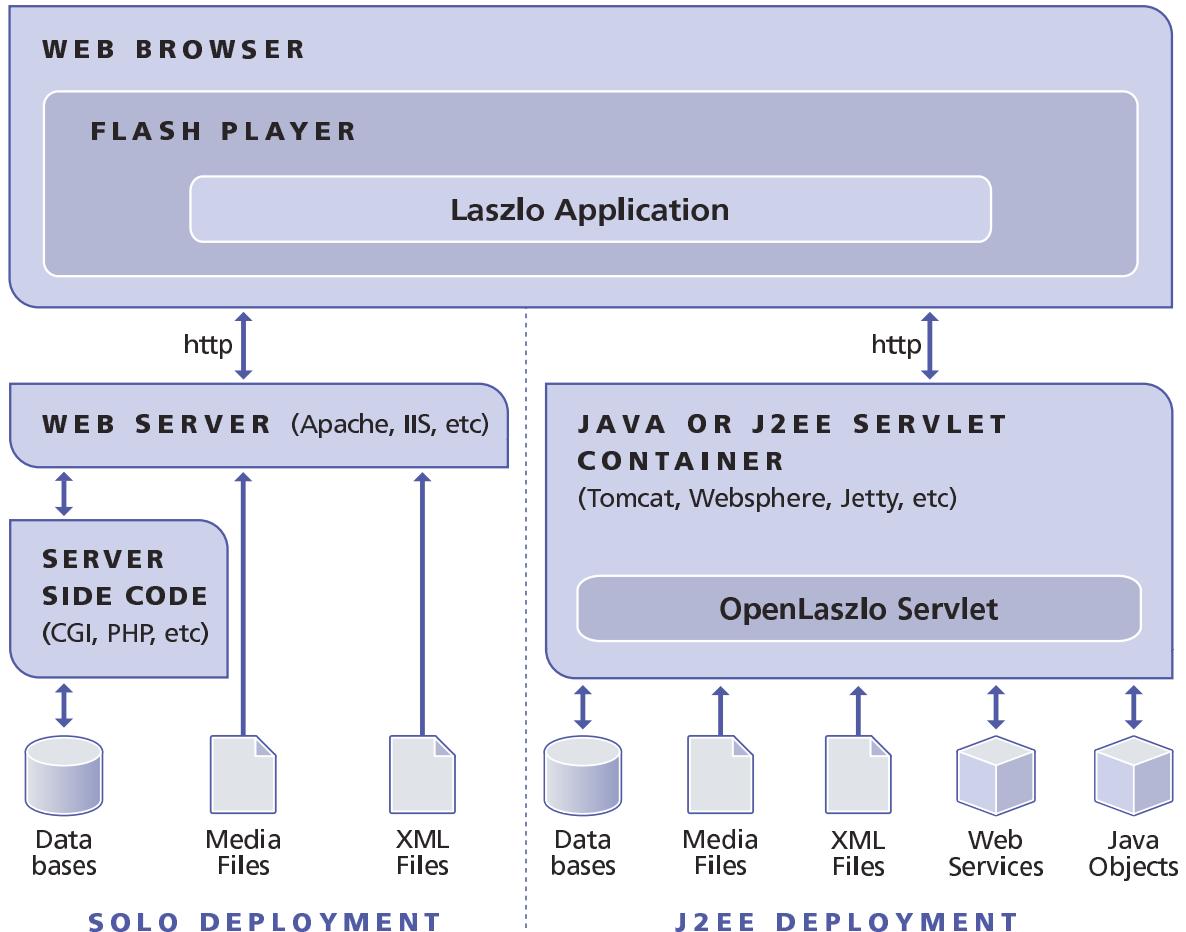


Abbildung 4.7.: Übersicht der Architektur von OpenLaszlo. Laszlo-Anwendungen lassen sich eigenständig (“Solo Deployment”) oder in Verbindung mit einem Java Servlet Container einsetzen. (Quelle: Laszlo Systems [21])

Open-Source-Lizenz⁵ zur Verfügung. Die Firma will damit die Verbreitung von RIAs und somit auch ihre anderen kommerziellen Produkte und Dienstleistungen fördern.

OpenLaszlo erweitert die Vorteile, die für den Einsatz RIAs sprechen:

- Die Open-Source Lösung ist dank langer Entwicklung stabil und gut dokumentiert.
- Die Programmiersprache ist objektorientiert und Komponenten-basiert.

OpenLaszlo verwendet Flash als Zielplattform für Internet-Anwendungen. Der Quellcode basiert auf XML und JavaScript. Die Programmiersprache mit dem Namen LZX ist eine objektorientierte, ereignisgesteuerte Auszeichnungssprache. Der Entwickler kann darin programmierte Anwendungen auf zwei verschiedene Arten verteilen (siehe Abbildung 4.7).

Bei dem “Solo Deployment” stellt ein Webserver (z.B. Apache) die bereits in das Flash-Format kompilierte Applikation bereit. Die im Browser des Benutzers laufende Flash-Anwendung kann hierbei dynamisch Daten aus XML-Dateien oder XML-generierenden serverseitigen Skripten laden.

⁵Common Public License (CPL 1.0)

4. Untersuchung der Voraussetzungen

Die zweite Verteilungsmöglichkeit benötigt einen Java Servlet Container (z.B. Apache Tomcat). Das auf diesem laufende OpenLaszlo Servlet übersetzt die Quellen der Anwendung zur Laufzeit und liefert Flash-Bytecode an den Browser aus. Diese Anwendung erlaubt im Vergleich zum “Solo Deployment” auch den Zugriff auf Java-spezifische Ressourcen.

Beispiele für OpenLaszlo-Anwendungen Die folgenden kurzen Laszlo-Programme veranschaulichen, wie die in XML programmierten Anwendungen aufgebaut sind.

Listing 4.1 zeigt das beispielhaft oft eingesetzte Hallo-Welt-Programm. Das Wurzel-Element von LZX ist `<canvas></canvas>`. Dieses Container-Element enthält alle weiteren Teile einer Anwendung, wie in diesem Fall den Text “Hallo Welt!”.

Listing 4.1: Beispiel für das obligatorische Hallo-Welt-Programm in OpenLaszlo

```
<canvas>
2   <text>Hallo Welt!</text>
</canvas>
```

Das Programm in Listing 4.2 demonstriert den Einsatz einer Ereignisbehandlungsroutine. Das `<simplelayout>`-Element sorgt für eine zeilenweise Anordnung der im Container enthaltenen Objekte. Der `<button>` reagiert auf sein `onclick`-Ereignis damit, dass er das seinem Elternobjekt untergeordnete Element mit dem Namen `helloclick` anzeigt.

Listing 4.2: Minimales ereignisgesteuertes LZX-Programm

```
<canvas>
2   <simplelayout axis="y" spacing="10" />
4     <button onclick="parent.helloclick.setVisible(true)">
5       Klick mich!
6     </button>
8     <text name="helloclick"
9       visible="false">Hallo Mausklick!</text>
</canvas>
```

Die beiden Beispiele zeigen nur die grundlegendsten Möglichkeiten von Laszlo-Anwendungen. Für weiterführende Informationen sei an dieser Stelle auf die Online-Dokumentation von OpenLaszlo verwiesen [20]. Eine kurzweilige 10-Minuten-Einführung in die Laszlo-Programmierung findet sich unter der Internetadresse <http://www.laszlosystems.com/lps/laszlo-in-ten-minutes/>. Diese Website zeigt einfache Beispielanwendungen und erlaubt es diese Online zu bearbeiten und zu kompilieren.

Bewertung

Mit OpenLaszlo existiert ein leistungsfähiges und gut dokumentiertes Open-Source-Projekt für die Entwicklung von RIAs. Es lässt sich ideal auf einem Java Servlet Container einsetzen. Laszlo bietet viele Komponenten zum Aufbau von Web-basierten Benutzerschnittstellen. Die Leistungsanforderungen der produzierten Flash-Anwendungen

in Bezug auf CPU und Bandbreite sind vertretbar, wobei sich diese Aussage auf einen Test stützt, in welchem die in diesem Projekt entwickelte Anwendung auf einem Rechner mit 300 MHz und einer Modemanbindung von 56 kbit/s ausgeführt wurde. Die Programmierung von Anwendungen mit einer Auszeichnungssprache ist ungewöhnlich. Positiv ist die daraus resultierende Möglichkeit, Anwendungen sofort “live” online kompilieren und testen zu können.

4.3.2. Datenbanksysteme

PostgreSQL

PostgreSQL ist ein objektrelationales Datenbankmanagementsystem (ORDBMS). Die University of California at Berkeley Computer Science Department entwickelte diese Software ursprünglich in einem universitären Projekt. Der Name dieser Datenbanksoftware spricht sich “post-gress-Q-L” aus. Die Namensgebung und Betonung soll auf die Fortführung des Ingres-Projektes an der selben Universität hinweisen.

PostgreSQL unterstützt die ISO/ANSI SQL92 und SQL99 Standards. Es ist das fortschrittlichste Open-Source-Datenbanksystem. Auch aus diesem Grund haben viele Entwickler Erweiterungen für PostgreSQL hervorgebracht, die dieses Datenbanksystem um Spezialfunktionen (z.B. GIS oder Statistik) ergänzen.

Eine der Erweiterungen ist PostGIS mit GEOS (siehe Abschnitt 4.3.3). Sie eröffnet in PostgreSQL die Möglichkeit, geografische Objekte zu speichern und mit diesen zu arbeiten. Diese Fähigkeit, die offene Software-Lizenz [31] und ein guter Support durch eine große Nutzergemeinschaft sind nur einige der Vorteile, aufgrund derer sich PostgreSQL als Datenbanksystem in einem epidemiologisches GIS besonders gut eignet.

MySQL

MySQL ist eines der am weitest verbreiteten Open-Source-Projekte. Der Finne Michael Widenius hat diese relationale DBMS 1994 entwickelt. Es hat sich besonders im Bereich der Webdienste in Verbindung mit Apache etabliert und erfreut sich einer bis heute ungebremsten Beliebtheit.

Geoinformationssysteme setzten dieses DBMS seltener ein, als die Kombination von PostgreSQL und PostGIS. MySQL erlaubt seit der Version 4.1 ebenfalls das Speichern von geometrischen Objekten. Diese Implementierung ist allerdings keine eigenständige Erweiterung wie bei PostgreSQL, sondern ein Teil des Gesamtsystems. Im Vergleich zu PostgreSQL bietet MySQL auch in anderen wissenschaftlichen Bereichen (wie z.B. Statistik) weniger Erweiterungsmöglichkeiten.

4. Untersuchung der Voraussetzungen

4.3.3. Geografische Informationssysteme

Offene Standards für GIS

Das Open Geospatial Consortium [26, OGC] ist ein gemeinnützig arbeitendes Gremium, das Standards im Bereich von GIS entwickelt. Offene Standards eröffnen viele Vorteile, wenn es um den Austausch von Daten zwischen Organisationen geht. Die USGS⁶ integriert z.B. in ihrem “The National Map”-Projekt mit Hilfe von OGC-Standards geografische Daten aus den unterschiedlichsten Quellen in einer GIS-Anwendung [75].

Zwei von dem OGC entwickelten Standards werden durch in diesem Projekt eingesetzten Softwarekomponenten implementiert:

- *OGC Simple Features Specification For SQL* [43, SFS]

Diese Spezifikation definiert, wie ein SQL-DBMS um eine geometrische Komponente erweitert werden kann. Die Datenbank muss dazu geometrische Daten speichern, aktualisieren und Abfragen in Bezug auf deren Topologie durchführen können. Der Standard spezifiziert auch Koordinatenprojektionen auf geografischen Daten.

- *Web Map Service* [44, WMS]

WMS bezeichnet einen Dienst, der in der Lage ist, geografische Informationen in eine digitale Karte umzuwandeln. Laut Definition handelt es sich hierbei nicht mehr um reine Geodaten. Karten sind immer grafische Darstellungen, die entweder mit Bitmaps oder Vektorgrafiken realisiert sind. Ein WMS muss nach diesem Standard folgende Operationen bereitstellen:

- *GetCapabilities*

Auf diese Anfrage hin gibt der WMS ein XML-Dokument mit den angebotenen Informationen zurück. Es beinhaltet unter anderem Metadaten des Dienstes, Auskunft über unterstützte Kartenprojektionen und angebotene Kartenebenen.

- *GetMap*

Dieser Request liefert eine digitale Karte zurück. Diese kann beispielsweise nur einen Ausschnitt der angebotenen Region mit einer Teilmenge der möglichen Ebenen darstellen.

- *GetFeatureInfo* (optional)

Ein WMS, der diese Operation unterstützt, erlaubt es, Attributinformationen (siehe Abschnitt 4.2.2) an beliebigen Positionen der mit *GetMap* erzeugten Karte abzufragen.

Ramsey [74] gibt in “The State of Open Source GIS” einen umfassenden Überblick der zur Zeit verfügbaren Softwarelösungen für freie Geoinformationssysteme. Das System EpiDeGIS nutzt folgende Werkzeuge aus diesem Bereich:

PostGIS

PostGIS erweitert PostgreSQL mit geografischen Fähigkeiten:

⁶United States Geological Survey

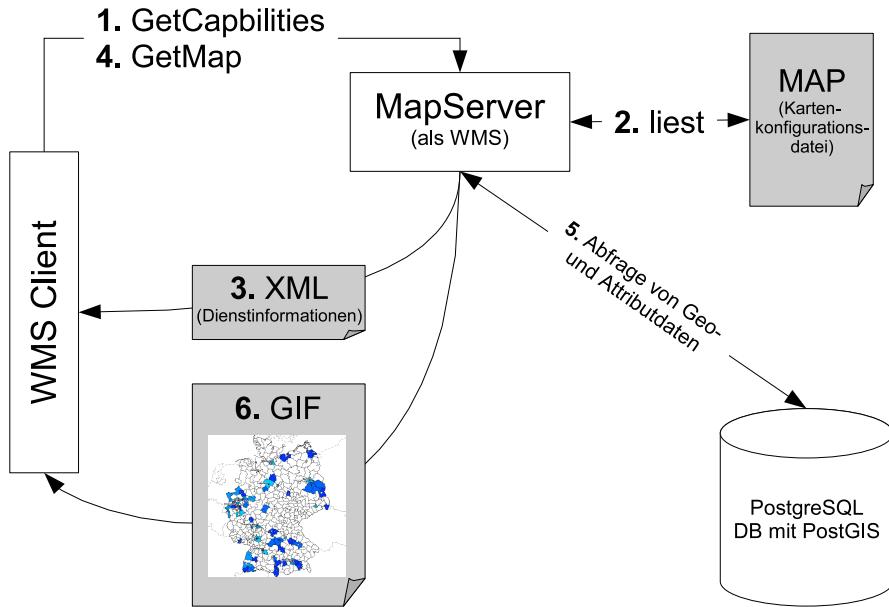


Abbildung 4.8.: Mapserver in seiner Funktion als WMS

- Speichern von geometrischen Objekten (point, line, polygon, multipoint, multiline, multipolygon, geometrycollections)
- Erzeugen von räumlichen Indizes
- Projizieren zwischen verschiedenen Koordinatensystemen mittels der Bibliothek PROJ.4 [56]
- Abfragen von topologischen Relationen mittels der GEOS-Bibliothek [73]
- Im- und Exportieren von geografischen Daten

PostGIS implementiert die SFS des OGC in großen Teilen. Viele andere Open-Source-Projekte nutzen PostgreSQL/PostGIS als geometrische Datenbank.

UMN MapServer

Der UMN MapServer ist eine Entwicklung der University of Minnesota, Minneapolis. Er ist kein vollständiges GI-System. Schon der Name hebt hervor, dass dieser Server „nur“ Karten ausgibt. In dieser Disziplin überzeugt das Projekt in Bezug auf Geschwindigkeit sowie die Vielzahl an implementierten Standards und unterstützten Datenformaten.

Der Mapserver implementiert den WMS-Standard einschließlich der Version 1.1.1. Mit den im vorigen Abschnitt beschriebenen Operationen erfüllt er bereits einen Teil der auf Seite 33 festgelegten Anforderungen an die GIS-Komponente. Abbildung 4.8 zeigt die Kommunikation einer WMS-Client-Anwendung mit dem MapServer. Der Server beantwortet die *GetCapabilities*-Anfrage indem er die in einer Konfigurationsdatei (siehe auch Listing D.2 auf Seite 97) festgelegte Kartendefinition als XML-Dokument ausliefert. Der Client fordert mittels *GetMap* bestimmte Ebenen als GIF-Grafik an. Der MapServer erzeugt anhand der Geo- und Attributinformationen aus der Datenbank ein Bild und liefert es zurück.

4. Untersuchung der Voraussetzungen

GeoTools

GeoTools [70] ist eine Java-basierte Open Source GIS-Bibliothek. Das Projekt implementiert die vom OGC verabschiedeten Standards. Es vereinfacht damit die Entwicklung von standardisierter GIS-Software in Java.

GeoTools stellt in diesem Projekt nur die initiale Verbindung zu dem WMS des MapServers her, um die Informationen über die verfügbaren Kartenebene zu erlangen.

4.3.4. Statistiksoftware

SaTScan

Die Software SaTScan analysiert räumliche, zeitliche und raum-zeitliche Daten mit Hilfe der entsprechenden statistischen Analysemethode. Martin Kulldorff hat das Werkzeug für die folgenden in gegenseitiger Beziehung stehenden Aufgaben entwickelt [65]:

- Überwachung von Erkrankungen durch die Identifizierung von räumlichen oder raum-zeitlichen Clustern und die Auswertung deren statistischer Signifikanz.
- Überprüfung, ob sich Erkrankungsfälle zufällig über Raum, Zeit oder beide Dimensionen verteilen.
- Auswertung der statistischen Signifikanz alarmierend erscheinender Erkrankungshäufungen.
- Identifizierung von Erkrankungsausbrüchen durch regelmäßige Analyse der aktuellsten Falddaten.

SaTScan ist in der Epidemiologie der de facto-Standard, wenn es um die Identifizierung von spatio-temporalen Erkrankungs-Clustern geht [47].

R

R [32] ist eine Programmiersprache und Laufzeitumgebung für statistische Berechnungen und Visualisierungen. Seine leichte Erweiterbarkeit hat dazu geführt, dass es diverse statistische Methoden zur Datenanalyse unterstützt. R kann unter anderem komplexe Zusammenhänge anhand verschiedenster Diagrammtypen veranschaulichen.

Im Zusammenhang mit einem Web-basierten System lässt sich R leider nur bedingt einsetzen. Die eigentliche Softwareumgebung ist ein Kommandozeilenprogramm mit optionaler grafischer Ausgabe.

Abhilfe schafft die von Joseph Conway entwickelte Software PL/R [53]. Hierbei handelt es sich um eine prozedurale Sprache, die PostgreSQL um die Funktionen von R erweitert. Die unterstützte Funktionalität reicht von der Berechnung des Medians einer Zahlenreihe bis zu der Erzeugung von komplexen Grafiken.

5. Konzepte

Das folgende Kapitel beschreibt alle wichtigen Konzepte, die beim Entwurf und der Entwicklung des Softwaresystems EpiDeGIS verwendet wurden. Das Hauptaugenmerk ist dabei auf die client- und serverseitige Anwendung zur Erzeugung von epidemiologischen Karten gerichtet. Die Applikation soll Änderungen der dargestellten Karten leicht ermöglichen. Zudem muss sie sich mit möglichst wenig Aufwand auf andere Krankheitserreger und/oder andere Länder übertragen lassen können.

Der erste Abschnitt beschreibt den Zusammenhang und die Einzelheiten der client- und serverseitigen Komponenten. Abschnitt 5.2 stellt die Bereiche vor, in denen die Architektur erweitert werden kann. Das letzte Unterkapitel behandelt Faktoren, die bei der Integration des MapServers als externes Werkzeug berücksichtigt werden müssen.

5.1. Datenmodell, Präsentation und Programmsteuerung

Die Überschrift dieses Abschnittes steht für die Übersetzung des Begriffs “Model-View-Controller” (MVC). Dieses Architekturmuster sieht folgende Komponenten vor:

- Das Datenmodell (Model) sind alle für eine Anwendung nutzbaren Informationen. Über das Modell hat sie beispielsweise Zugang zu einer Datenbank.
- Die Präsentation (View) ist die für den Benutzer sichtbare Anwendung. Sie bildet das Datenmodell in einer visuell erfassbaren Form ab.
- Die Programmsteuerung (Controller) steht zentral. Sie hat Zugriff auf das Datenmodell und steuert alle Prozesse.

Der MVC-Ansatz ermöglicht die Entwicklung von modularen, erweiterbaren Anwendungen. Jede Teilkomponente übernimmt dabei klar definierte Aufgaben. Für die Webanwendung dieses epidemiologischen GI-Systems sind die einzelnen Verantwortlichkeiten wie folgt verteilt:

- Der MapServer und JavaBeans [18] bilden mit der Datenbank das Modell.
- Die in OpenLaszlo implementierte RIA, über JSP erzeugte HTML-Dokumente und digitale Grafiken übernehmen die Präsentation.
- Struts steuert die serverseitigen Abläufe.

Abbildung 5.1 verdeutlicht das Zusammenspiel der Komponenten.

5. Konzepte

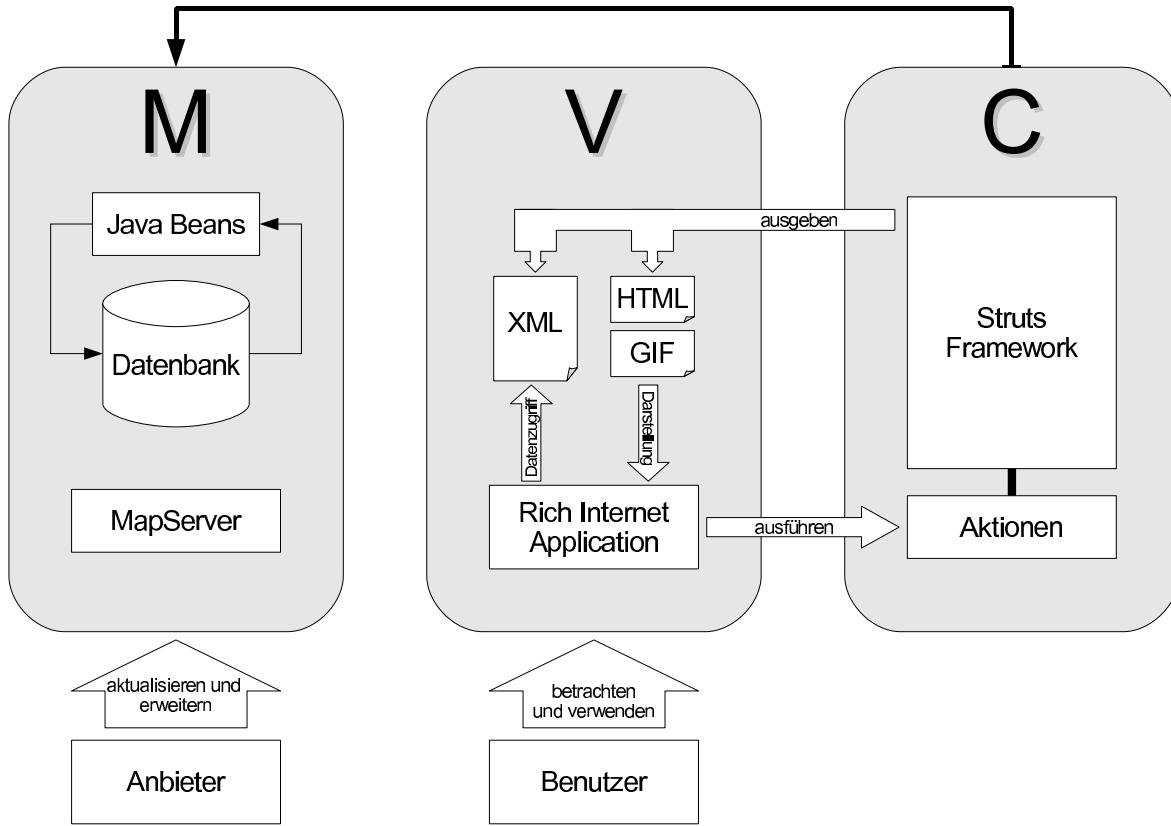


Abbildung 5.1.: Verwendung des MVC-Architekturmusters in EpiDeGIS

5.1.1. Globale und benutzerspezifische Daten

Datenbankkonzepte

Das PostgreSQL Datenbanksystem speichert alle im gesamten System benötigten und entstehenden Informationen. Diese sind in mehreren Tabellen abgelegt. EpiDeGIS setzt zwei Konzepte ein, um Änderungsmöglichkeiten des Systems auf Datenbankebene zu gewährleisten.

Übertragbarkeit auf andere Länder und/oder geografische Hierarchien Geografische Daten stehen zueinander immer in einer topologischen Beziehung. Bei Flächen-daten spiegelt diese häufig die Hierarchie wider.

Für Deutschland lautet sie z.B. *Bundesgebiet → Länder → Regierungsbezirke → Landkreise*. Auch andere Länder bzw. Strukturen sind räumlich und organisatorisch auf diese Weise geordnet (z.B. *Universitätsklinikum Würzburg → Abteilungen/Kliniken → Gebäude → Stockwerke → Zimmer*).

Die Datenbank bildet solche Strukturen mittels zwei Tabellen ab. Eine speichert die Hierarchie und die verschiedenen Arten von erfassten Strukturen. Die zweite Tabelle enthält die zu bestimmten Typen gehörenden polygonalen Daten (siehe Abbildung 5.2).

Die Daten lassen sich bei dieser Struktur leicht um ein anderes Land erweitern: Holländische Epidemiologen wenden für Meningokokken ebenfalls die PorA/FetA -

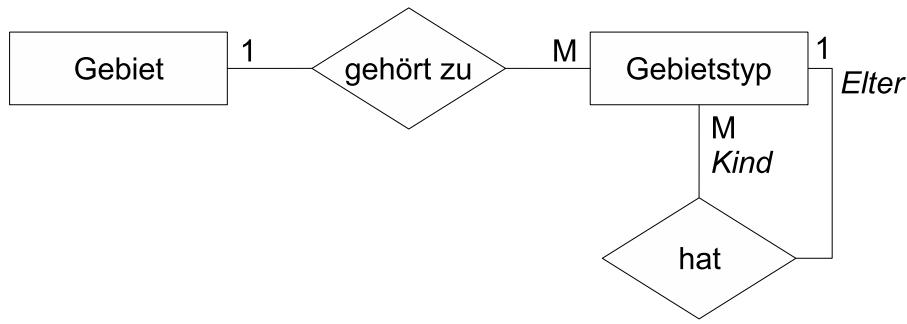


Abbildung 5.2.: Hierarchische Speicherung von Gebietsinformationen

Feintypisierungsmethode an. Integration der holländischen Daten erfordert eine Definition der Hierarchie mit den drei Einträgen *Nederlandse* → *Provincies*, → *Gemeentes* in der Tabelle der Gebietstypen. Im zweiten Schritt müssen die Grenzdaten der jeweiligen Bereiche unter den Gebieten abgelegt werden.

Erweiterbarkeit der epidemiologischen Datenstruktur um beliebig viele Attribute
 Das Datenbankkonzept erlaubt es, die gepeicherten Falldaten um beliebige Attribute zu erweitern.

Es legt dabei in einer Tabelle zu jedem Fall nur solche Daten ab, die auf alle Proben von Krankheitserregern zutreffen:

- eine eindeutige Nummer, die intern (z.B. im NRZM) eine Zuordnung zu detaillierten Probedaten erlaubt
- das Alter
- das Geschlecht
- das Eingangsdatum der Probe im Labor
- das Entnahmedatum beim Arzt
- die Koordinaten (Länge und Breite) des Ortes, in dem der Fall aufgetreten ist
- die eindeutige Kennzeichnung des Erregertyps (z.B. der Feintyp bei Meningokokken)

EpiDeGIS setzt Attribute ein, um spezifische Merkmale eines Erregertyps außerhalb der Grundstruktur abzulegen:

Ein vom NRZM ermittelter Feintyp hat beispielsweise die Struktur *{Serogruppe}:P1.{PorA}:F{FetA}*. Die Kombination der einzelnen Attribute Serogruppe, PorA und FetA kennzeichnet einen Meningokokkenerreger eindeutig. Die Serogruppe für sich ist jedoch eine Gemeinsamkeit mehrerer Feintypen.

Die Datenbank soll diesen Zusammenhang speichern, ohne die obigen epidemiologischen Grunddaten um eine Spalte mit der Serogruppe ergänzen zu müssen. Das gelingt, indem sie für jeden Erregertyp beliebig viele Attribute zulässt. Der konkrete Feintyp *B:P1.7-2,4:F1-5* besitzt dabei ein Attribut mit dem Namen “Serogruppe”, das den Wert “B”

5. Konzepte

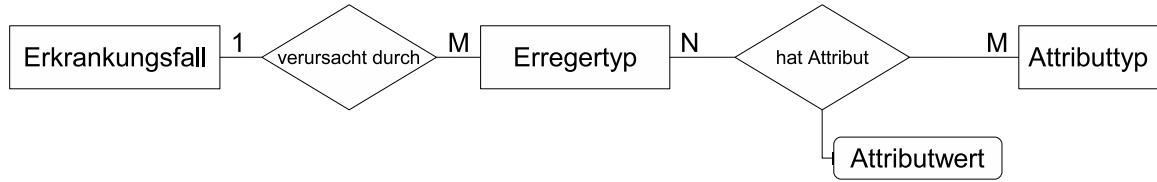


Abbildung 5.3.: Definition von beliebig vielen Attributwerten für einen Erregertyp

trägt. Genauso wäre in diesem Fall ein Attribut "Impfstoff für Deutschland vorhanden" mit dem Wert "Nein" denkbar (siehe Abbildung 5.3).

Die Datenbank erweitert bisher nur den Erregertyp um Attribute. Für einzelne Erkrankungsfälle könnte das gleiche Konzept eingesetzt werden, um z.B. den Wert eines Attributes "Antibiotikaresistenz" festzuhalten.

Daten im Servletcontainer

Die Internet-Applikation muss Daten über den Anwendungszustand aus Sicht des Benutzers oder des Gesamtsystems serverseitig abspeichern. Der Java Servlet Container definiert zum Ablegen dieser Informationen unterschiedliche Bereiche [37]:

Page Diese Daten sind nur während des Verlaufs einer einzigen JSP gültig.

Request Diese Informationen sind für die Dauer einer Anfrage im System. Diese verläuft u. U. über mehrere Seiten/Komponenten.

Wenn der Benutzer z.B. den sichtbaren Kartenausschnitt verschieben möchte, übermittelt die RIA den Wunsch an den Server. Sobald dieser die Aktion ausgeführt hat, löscht er die mit dem Request verknüpften Informationen (hier Δx und Δy der angeforderten Kartenverschiebung).

Session Alle in diesem Bereich abgelegten Daten bleiben so lange gültig, wie der Benutzer online mit der Anwendung interagiert. Über die Konfiguration des Tomcat Servers lässt sich festlegen, wie viele Minuten die Session-Daten bei Inaktivität erhalten bleiben.

Application Für die ganze Anwendung zugängliche Daten werden hier gespeichert.

5.1.2. Präsentation der Informationen

Flashanwendung mit OpenLaszlo

Die Abbildung 5.1 führt die Rich Internet Application als View auf. Der Anwender würde sie wahrscheinlich als View und Controller bezeichnen. So präsentiert sie ihm doch den Zustand des Systems bzw. seiner Session und erlaubt gleichzeitig Kontrolle durch Interaktion. Eine klarere Einordnung behandelt Client und Server getrennt.

Vom Server gesehen ist auch die RIA nichts anderes als eine View-Komponente. Im Moment einer Anfrage wird sie von dem Laszlo-Servlet kompiliert und an das Flash-Browser-Plugin auf dem Rechner des Benutzers ausgegeben.

Sobald die Laszlo-Anwendung beim Benutzer geladen ist, bedient sie sich eines weiteren serverseitigen Views. Dieses stellt den für diesen Benutzer gültigen Systemzustand als XML dar. Die Technik von OpenLaszlo generiert die Benutzeroberfläche anhand dieser Information dynamisch (Abschnitt 6.3.1).

Der Server erzeugt auch die Darstellung der Karten, welche die RIA ihrerseits wieder anzeigt. Weiterhin überträgt sie jede Interaktion des Anwenders an den Server und kontrolliert damit auch den Programmfluss.

Auf dem Client-Rechner bildet die Anwendung damit eine eigene MVC-Architektur. Ihr Controller gibt allerdings alle Eingaben des Benutzers an den Server weiter und reagiert auf dessen Antworten.

HTML-Karte für den Druck

Flash-Anwendungen bieten dem Anwender zwar viele Möglichkeiten zur Interaktion, die Ausgabe der angezeigten Seite über einen Drucker ist allerdings nur im Rahmen der durch den Browser bereitgestellten Funktionen möglich. Zudem ist es bei einem epidemiologischen GI-System nicht im Interesse des Anwenders, das Abbild einer Benutzeroberfläche auszudrucken. Vielmehr sollte ein Bericht die aktuell sichtbare Karte, die Legende und Informationen zu den eingestellten Filtern ausgeben.

Eine dynamisch über JavaServer Pages generierte HTML-Seite erfüllt diese Funktion. Sie lässt sich ausdrucken oder lokal speichern. Struts ist dafür verantwortlich, die JSP mit Daten zu versorgen. Der Server muss dafür zu jeder Zeit in einem aus Anwendersicht gültigen Zustand sein.

5.1.3. Wohl definierte Zustandsänderungen

OpenLaszlo motiviert den Entwickler dazu, die Programmlogik in der Clientanwendung zu implementieren. Dies entspricht dem Konzept von RIAs, indem es verzögerungsanfällige Kommunikation zwischen Client und Server minimiert.

Diese Arbeit setzt, wie unter 5.1.2 erläutert, die clientseitige Anwendung nur als eine Form der Präsentation ein. Das schafft ein konsistentes Gesamtsystem, dessen Zustand nur der Server kontrolliert. Die Laszlo-Anwendung fungiert dabei als Schnittstelle, die alle Eingaben des Benutzers an den serverseitigen Controller (Struts) übermittelt. Die Kommunikation erfolgt über HTTP-Requests und zurückgegebene XML-Dokumente.

Der Vorteil eines definierten serverseitigen Zustandes liegt darin, dass verschiedene Präsentationen ihn zu jeder Zeit gleich abbilden. Ein konkretes Beispielszenario ist folgendes:

Der Anwender startet die RIA und arbeitet damit. Er lässt sich beispielsweise auf einer Karte alle Erkrankungsfälle von 2-5 jährigen im Jahr 2004 anzeigen. Der Benutzer öffnet in seinem Browser versehentlich eine andere Website und schließt dabei die Flash-Anwendung. Sollte er sie noch einmal öffnen bevor sein Session-Timeout abgelaufen ist, so sind alle festgelegten Filter noch eingestellt, obwohl die clientseitige Anwendung bereits beendet war.

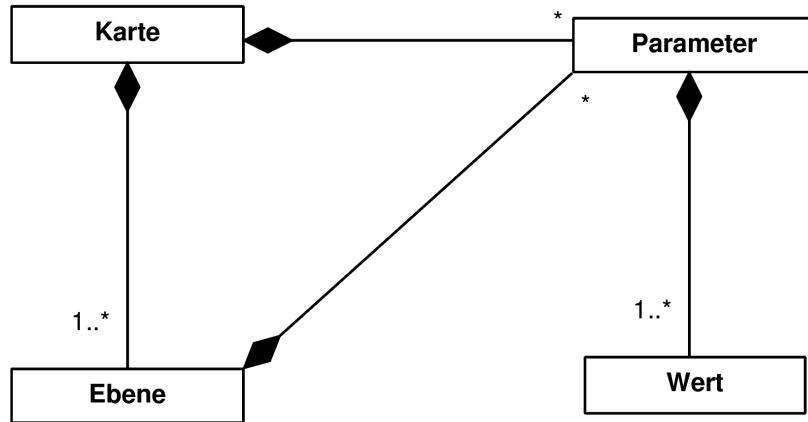


Abbildung 5.4.: Komposition von Karten, Ebenen und Parametern [29]

5.2. Leichte Erweiterbarkeit

Ein wichtiges Kriterium bei der Planung dieser Arbeit war es, ein epidemiologisches Geoinformationssystem zu schaffen, das erweiterbar ist. Abschnitt 5.1.1 hat Datenbankkonzepte vorgestellt, die einen Ausbau der abgelegten geografischen und epidemiologischen Daten zulassen.

In diesem Abschnitt soll es um Erweiterungen des Systems gehen, die im Java-Quellcode, beim MapServer oder an den Schnittstellen der Webanwendung ansetzen.

5.2.1. Dynamische Karten- und Ebenenparameter

Eine Reihe von Java-Klassen in diesem System modelliert den abstrakten Datentyp einer Karte. Sie enthält mindestens eine Ebene und hat beliebig viele “Kartenparameter”. Eine Ebene kann “Ebenenparameter” besitzen (Abbildung 5.4). Die Parameter sind eine Ansammlung von möglichen Werten für ein Attribut, das entweder der ganzen Karte oder nur einzelnen Ebenen zugeordnet ist.

Ein für die ganze Karte gültiger Auswahlparameter gibt beispielsweise an, welche geografischen Teilgebiete vergrößert betrachtet werden können. Er enthält z.B. für Deutschland die Werte *Bundesgebiet, Baden-Württemberg, Bayern, ..., Thüringen*. Wenn der Anwender ein anderes Gebiet auswählt, muss das System alle Kartenebenen aktualisieren.

Ebenenparameter arbeiten nach dem gleichen Prinzip. Von ihrem Attributwert ist allerdings nur eine Ebene abhängig. Ein Beispiel ist die Feintypen-Ebene der epidemiologischen Karte für das NRZM. Sie zeigt alle durch einen Feintyp verursachten Erkrankungsfälle im Gebiet an. Der zugehörige Parameter enthält damit alle durch den Anwender auswählbaren Feintypen.

Das Parameterkonzept sichert Erweiterbarkeit im Hinblick auf Filter, anhand derer die Kartendarstellung beeinflusst werden kann. Die RIA bietet Parameter z.B. in Form von Dropdown-Listen an. Dabei sind Kartenparameter auf der Benutzeroberfläche immer sichtbar, während Ebenenparameter nur im Bereich einer Ebene erscheinen (siehe

Abbildung 5.5).

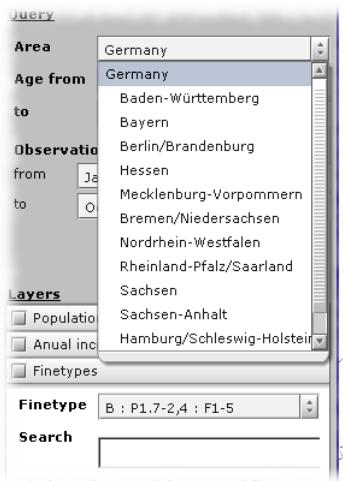


Abbildung 5.5.: Konkrete Umsetzung des Parameterkonzeptes in der Laszlo-Anwendung - oben ein Kartenparameter (Area), unten ein Ebenenparameter (Finetype).

falls aktualisiert werden. In der Implementierung ist dieses Problem mit Platzhalterparametern gelöst, die einen existierenden Parameter referenzieren. Diese Implementierung bietet allerdings noch Raum für Verbesserungen.

Die Implementierung dieses Konzeptes hat zu Schwierigkeiten geführt, aufgrund derer sich die ursprüngliche Bedeutung von Karten- und Ebenenparametern verändert. Der Begriff des Kartenparameters ist in der grafischen Benutzeroberfläche ein vom Anwender einstellbarer Wert, dessen Steuerelement immer sichtbar ist. Für den Ebenenparameter ist dieses einer Ebene untergeordnet. Ein Kartenparameter ist dabei aber nicht zwangsläufig global für alle Ebenen verantwortlich und der Parameter einer Ebene kann seinerseits auch andere Ebenen beeinflussen.

Ein Beispiel veranschaulicht die Situation: Die Karte enthält zwei Ebenen. Eine zeigt Meningokokkenfälle gruppiert nach Serogruppen, die andere die zugehörige Inzidenz nach Landkreisen an. Ändert der Anwender die Serogruppe, so muss die Inzidenz-Ebene ebenfalls aktualisiert werden.

5.2.2. Änderbarkeit der Kartendefinition des MapServers

Abbildung 4.8 auf Seite 41 zeigt, dass der MapServer die Konfiguration der von ihm angebotenen Karten über eine Definitionsdatei ("MapFile") einliest. Diese legt in einem definierten Format [54] alle Eigenschaften der über den Server publizierten Karte fest.

Dazu zählen unter anderem:

- Metadaten über den angebotenen Dienst (Name, Beschreibung)
- Globale Einstellungen der Karte (verwendetes Koordinatensystem, die Grenzkoordinaten des abgebildeten Gebietes, etc.)
- Definitionen der verschiedenen Kartenebenen

Ein Anbieter von epidemiologischen Informationen muss für Erweiterungen in der Karte hauptsächlich die Ebeneneinstellungen ändern. Diese definieren alle vom Anwender später sichtbaren Kartenelemente. Das sind beispielsweise die eingesetzten Farben und Symbole, die ein Merkmal auf der Karte auszeichnen, oder der Name einer Ebene.

Der wichtigste Bestandteil jeder Ebene ist die Definition der Datenquelle. Beim MapServer können an dieser Stelle beliebige Abfragen an die Datenbank stehen, deren Änderung vielfältige Möglichkeiten zur Einflussnahme in die Kartenerzeugung bietet (siehe Abschnitt 6.2).

5. Konzepte

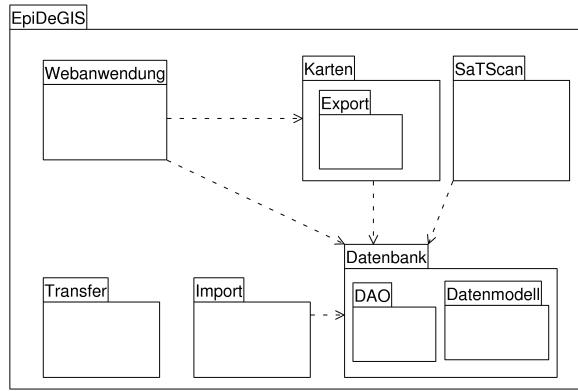


Abbildung 5.6.: Paketdiagramm des EpiDeGIS-Systems [29]

EpiDeGIS integriert den MapServer auf die Art, dass Änderungen an der Kartendefinition grundsätzlich sofort¹ in der Benutzeroberfläche sichtbar sind. Eine statische Ebene „Flüsse“, die in der „MapFile“ definiert wird, ließe sich somit augenblicklich einsetzen.

Im Falle einer parametrisierbaren Ebene, die beispielsweise die Erkrankungsfälle als Punkte begrenzt auf bestimmte Altersgruppen anzeigt, sind nur geringfügig mehr Eingriffe im System notwendig. Die Nutzung eines bereits bestehenden bzw. die Definition eines neuen Parameters wird über Änderungen im Java-Code gesteuert.

Die Definitionsdatei des MapServers erlaubt vielfältige Eingriffe in die Kartenerzeugung. Sie lässt sich dank eines beschreibenden textuellen Dateiformates auch von Personen anpassen, die selbst nur wenig Programmiererfahrung haben. Zusammen mit Java- und SQL-Kenntnissen erlaubt EpiDeGIS den Anwendern noch weitreichendere Änderungen des Systems.

5.2.3. Aufteilung der Systemkomponenten in Pakete

Pakete erlauben es, die Komponenten eines Softwaresystems zu größeren Einheiten zusammenzufassen. Das Paketdiagramm in Abbildung 5.6 zeigt die Aufteilung des in dieser Arbeit entwickelten Systems.

Die Klassen in den Paketen haben im Einzelnen folgende Aufgaben:

Datenbank Dieses Paket implementiert die Java-Schnittstelle zu der PostgreSQL-Datenbank. Die Implementierung basiert auf dem Prinzip des „Data Access Objects“-Entwurfsmusters (DAO) von Sun [7]. Dieses Entwurfsmuster versteckt alle Zugriffe auf eine Datenbank vor dem Rest der Anwendung.

Alle Datenbankzugriffe in EpiDeGIS laufen über dieses Paket. SQL-Code verschwindet damit komplett aus dem Java-Code aller anderen Klassen.

Karten Hier sind alle Klassen zusammengefasst, die für die Java-Repräsentation einer Karte benötigt werden. Sie implementieren die abstrakten Datentypen Karte, Ebenen

¹sobald ein neuer Benutzer die Anwendung öffnet

und Parameter sowie die Kommunikation mit dem MapServer.

Webanwendung Die Webanwendung enthält alles, was der Tomcat-Server zum Ausführen der serverseitigen EpiDeGIS-Komponente benötigt. Das ist die mit Struts implementierte Webanwendung, der Quellcode für die OpenLaszlo-Applikation und alle weiteren Ressourcen (JSP, CSS, Grafiken). Die Webanwendung verwendet das Karten-Paket als Schnittstelle zum Mapserver.

SaTScan Ein kleines Java-Programm ist für die automatische Ausführung von SaTScan zur Cluster-Identifizierung zuständig. Es ist in diesem Paket implementiert (siehe Abschnitt 6.4 auf Seite 70).

Transfer und Import Der Java-Code in diesen Paketen implementiert die Übertragung und den Import der Typisierungsdaten. Die Transfer-Komponente läuft clientseitig (z.B. im NRZM) ab, die Import-Anwendung nimmt auf dem Server die Daten entgegen und aktualisiert die Datenbank.

5.2.4. Erweiterungen an externen Schnittstellen

WMS des MapServers

Abschnitt 4.3.3 stellt WMS als Schnittstelle zu Kartenservern vor und beschreibt die Umsetzung dieses Standards im Mapserver. Dank dieses Dienstes existiert in EpiDeGIS eine bekannte Schnittstelle, über die beispielsweise eine Desktop-basierte GIS-Anwendung Karten des Systems abfragen kann.

Während dieser Arbeit wurde diese Möglichkeit experimentell getestet. Abbildung 5.7 zeigt die Integration der epidemiologischen Daten des NRZM in der Desktop-GIS-Anwendung uDig² [41]. Dieser Versuch demonstriert die Kommunikation mit dem MapServer über die WMS-Schnittstelle.

XML-Repräsentation des Serverzustands

Die serverseitige Komponente der EpiDeGIS-Webanwendung bietet eine XML-Repräsentation der angebotenen Karte. Diese enthält alle Ebenen und alle zulässigen Werte für die Parameter.

Externe Applikationen könnten diese Informationen verwenden, um ihrerseits mit dem System zu kommunizieren und die Karten zu nutzen.

²User-friendly Desktop Internet GIS

5. Konzepte

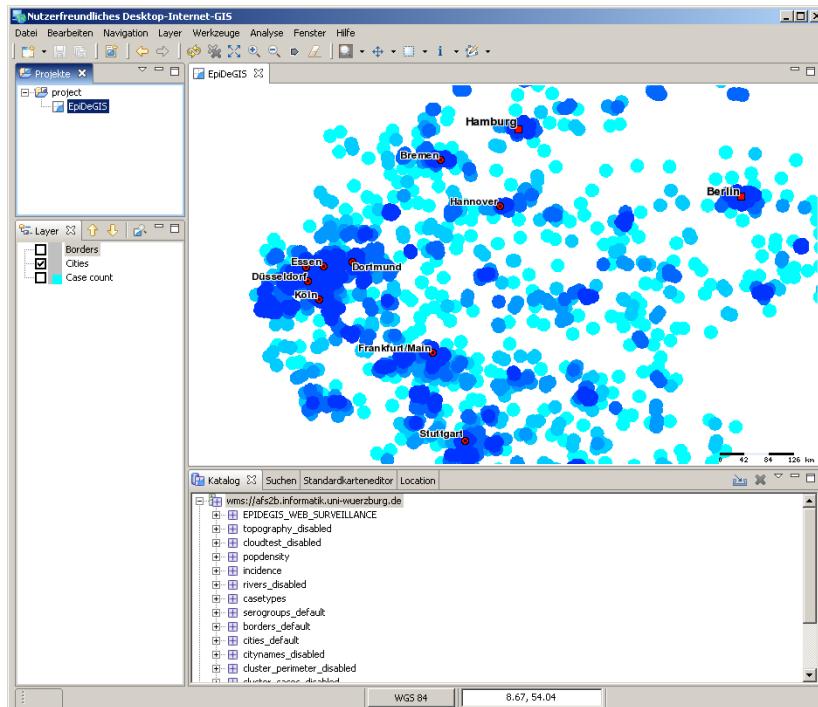


Abbildung 5.7.: Screenshot einer Desktop-GIS-Anwendung die Ebenen aus EpiDeGIS einbindet

5.3. Kapselung des MapServers und Caching

Der MapServer ist ein mächtiges Werkzeug, um komplexe epidemiologische Karten zu generieren. Die Standardschnittstellen des WMS-Standards führen jedoch zu Datenschutzrisiken, wenn dieser Server direkt über das Internet zugänglich wäre.

EpiDeGIS kapselt ihn daher so ab, dass der direkte Zugriff nur über den serverseitigen Controller erfolgt (siehe Abbildung 5.1).

5.3.1. Datenschutz durch indirekten Zugriff

Wie im Abschnitt 5.2.1 erläutert, setzt diese Arbeit das Konzept der “Parameter” ein, um Attributwerte für Karten und Ebenen zu definieren.

Die vom Anwender übermittelten Attribute werden auf Seiten des MapServers innerhalb der SQL-Abfragen eingesetzt, welche die Daten für eine Kartenebene zurückliefern.

Listing 5.1: SQL-Abfrage als Datenquelle für eine Kartenebene

```
1  SELECT case_id AS oid , the_geom
2   FROM cases
3   WHERE reportdate
4     BETWEEN to_timestamp ('%fromDate%', 'MMYYYY')
          AND to_timestamp ('%toDate%', 'MMYYYY');
```

Listing 5.1 zeigt eine SQL-Query, die alle Fälle aus der Datenbank holt, deren Melde-datum zwischen den Werten %fromDate% und %toDate% liegt.

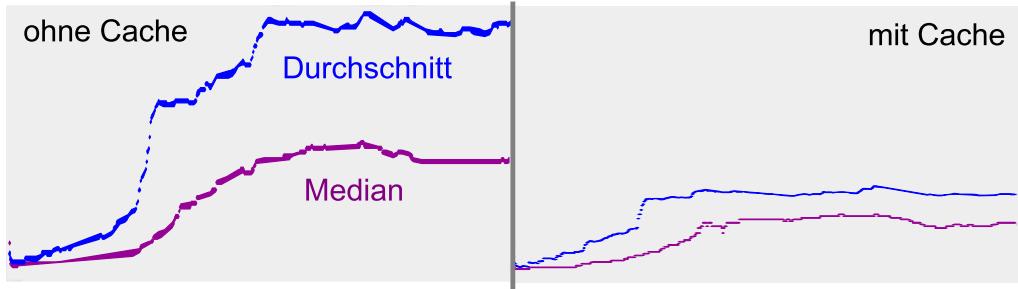


Abbildung 5.8.: Graf der Antwortzeiten mit und ohne Cache

Der MapServer ersetzt die Platzhalter mit den Werten von gleichnamigen, über HTTP-Get übermittelten Parametern. Der hier vereinfacht dargestellten URL `http://localhost/GetMap?fromDate=012005&toDate=102005` liefert eine Karte mit allen Fällen von Januar 2005 bis Oktober 2005 zurück.

Sollte ein Anwender direkten Zugriff auf den MapServer haben, wäre er aufgrund dieser Art der dynamischen Kartenerzeugung in der Lage, Parameterwerte anzugeben, die vom Anbieter der Informationen gar nicht vorgesehen sind.

EpiDeGIS verhindert das durch einen indirekten, internen Zugriff auf den Mapserver. Der serverseitige Controller erlaubt nur Parameterwerte, die vorher in der Anwendung definiert wurden.

Die Implementierung für das NRZM setzt diesen Schutzmechanismus ein, um z.B. zu verhindern, dass ein Kartenausschnitt Berlin in voller Größe abbildet. Aus Datenschutzgründen darf für die Öffentlichkeit keine Zuordnungsmöglichkeit eines Erkrankungsfalls zu einem Stadtteil möglich sein.

5.3.2. Optimierte Antwortzeiten durch Caching

Ein nützlicher Nebeneffekt der Abschottung des MapServers nach außen ist, dass sich dadurch gleich die Möglichkeit ergibt, die Geschwindigkeit der Kartenerzeugung zu verbessern.

Abschnitt 6.2 stellt Implementierungsbeispiele für Kartenebenen vor. Diese setzen teilweise komplexe topologische Abfragen an die PostGIS-Komponente der Datenbank ein und benötigen unter Umständen eine längere Ausführungszeit (über eine Sekunde).

Durch die Implementierung eines Zwischenspeichers an der Schnittstelle zum MapServer lässt sich die Zeit für die Kartenerzeugung deutlich verringern.

Der Cache-Mechanismus nimmt dazu den Hashwert des URL, der den MapServer veranlasst, eine bestimmte Karte auszugeben. Unter diesem Schlüssel wird auf dem Java Server bei der ersten Anfrage die Bilddatei der erzeugten Karte und der dabei verwendete URL abgelegt.

Bei einer folgenden Anfrage für die gleiche Karte kann das Bild direkt zurückgegeben werden, ohne es erneut über den MapServer erzeugen zu müssen.

5. Konzepte

	Durchschnitt	Median	Maximum
ohne Cache	2525 ms	1234 ms	25094 ms
mit Cache	799 ms	578 ms	9781 ms

Tabelle 5.1.: Werte für die Antwortzeiten mit und ohne Cache

Der Graph in Abbildung 5.8 zeigt die Ergebnisse einer Messung von Antwortzeiten. Hierbei haben fünf Threads parallel ein vorher aufgezeichnetes Nutzerverhalten als Last für den Server erzeugt. Der Test wurde einmal mit und ohne Cache durchgeführt. Der Anstieg zu Beginn der Kurve entsteht durch das Hochfahren der Belastung von einem auf fünf simulierten Benutzer.

Die Grafik und die Werte in Tabelle 5.1 zeigen deutlich, dass die direkte Anfrage an den Mapserver mehr Zeit benötigt. Die in beiden Fällen großen Maxima lassen sich dadurch erklären, dass während der Messung eine noch nicht optimierte Kartenebene aus einem anderen Experiment aktiv war.

6. Implementierungsdetails

Dieses Kapitel verdeutlicht anhand von Programm- und Datenbankfragmenten, wie das Softwaresystem zu dieser Arbeit implementiert ist. Die jeweils herausgenommenen Programmfragmente beziehen sich dabei immer nur auf die Teile der Umsetzung, die zum Verständnis des Textes nötig sind. Das Element [...] kennzeichnet dabei Auslassungen in Quellcode-Listings. Anhand von konkreten Beispielen wird der Einsatz aller beschriebenen Techniken demonstriert.

Zum besseren Verständnis der Zusammensetzung aller Einzelkomponenten von EpiDeGIS stellt Abbildung 6.1 eine komplette Übersicht dar.

6.1. Serverseitige Anwendung

Wenn sich ein Anwender mit der Website von EpiDeGIS verbindet, laufen serverseitig die Punkte 1-3 der Abbildung 6.1 ab. Das Laszlo-Servlet kompiliert aus dem LZX-Quellcode eine Flash-Datei und liefert diese an den Browser zurück. Falls der Quellcode nicht geändert wurde, gibt das Servlet eine gespeicherte Version der bereits kompilierten Anwendung aus.

Der Browser führt das Programm clientseitig aus und stößt erst damit die Erfassung des Benutzers in der serverseitigen Komponente von EpiDeGIS an. Das OpenLaszlo-Servlet läuft zwar ebenfalls auf dem Tomcat, ist allerdings nicht mit dem Rest des Softwaresystems verbunden. Laszlo generiert “lediglich” das Flash-Programm aus dem LZX-Quellcode.

6.1.1. Initialisierung der Karte

Initialisierungsprozess

Sobald der Benutzer die RIA von EpiDeGIS öffnet findet der eigentliche Initialisierungsvorgang statt. Dieser Prozess generiert eine Kartendefinition und speichert diese in der Session des Benutzers als Java-Objekt.

Das Sequenzdiagramm in Abbildung 6.2 zeigt die Abläufe:

- 1) Die serverseitige Webanwendung erzeugt das Objekt einer Klasse, die von *AbstractWmsMap* abgeleitet ist.
- 2) - 3) Über die GeoTools-Bibliothek wird eine Verbindung zum Mapserver hergestellt, um dessen Kartendefinition zu erfragen.

6. Implementierungsdetails

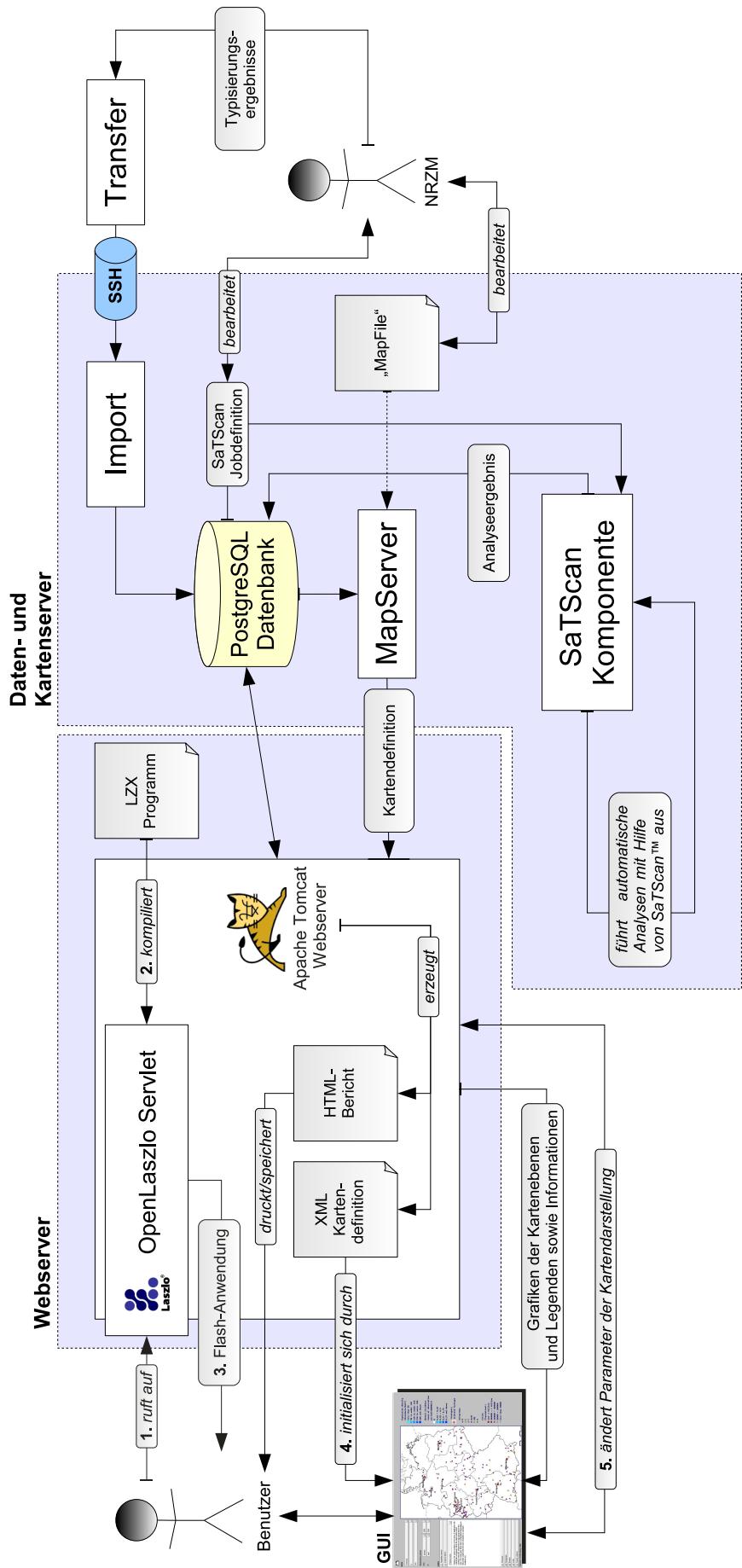


Abbildung 6.1.: Komplettübersicht des Systems EpiDeGIS

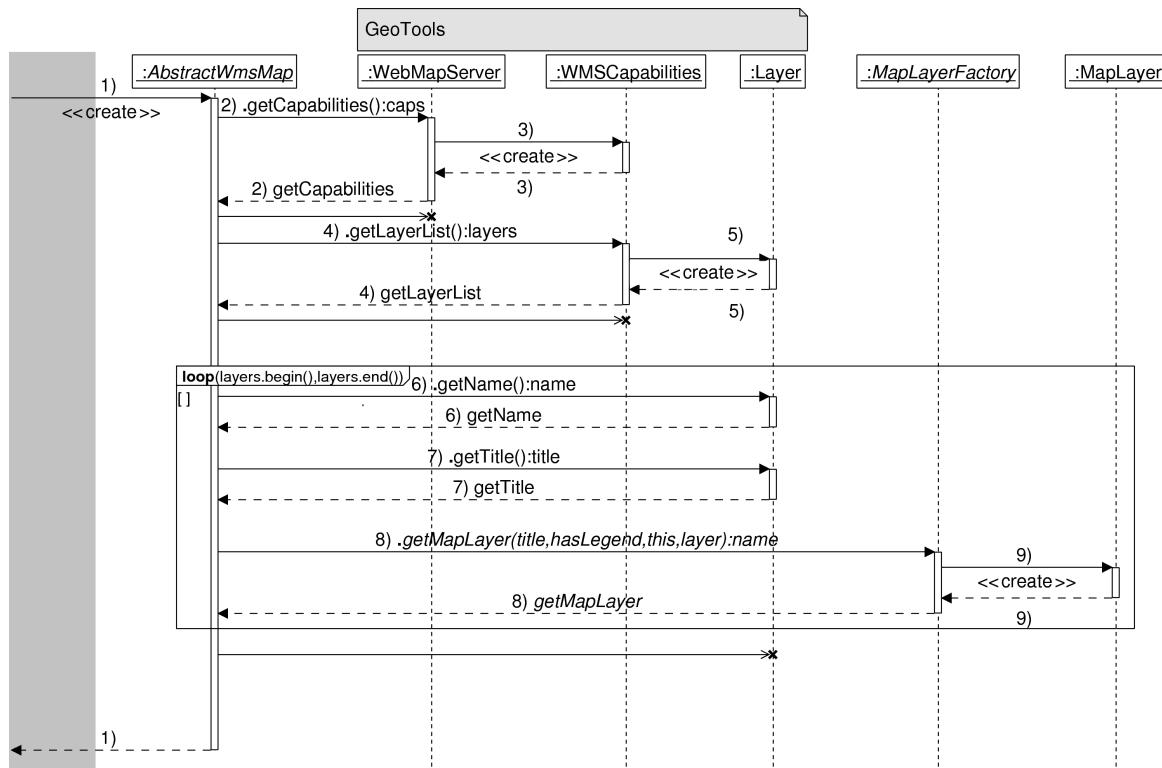


Abbildung 6.2.: Sequenzdiagramm der Karteninitialisierung [29]

4) - 5) Die `WMSCapabilities` beschreiben u. a. die verfügbaren Ebenen und geben diese als Liste von `Layer`-Objekten zurück.

Loop über alle Ebenen:

6) + 7) Das `Layer`-Objekt gibt den Namen (eindeutig) und den Titel (beliebig) einer Kartenebene zurück.

8) -9) Die `MapLayerFactory` erzeugt anhand des Namens die EpiDeGIS-Implementierung einer Kartenebene.

“Produktion” von Ebenen

Die `MapLayerFactory` ist im Bezug auf das eingesetzte Fabrikmethode-Muster [48], eine abstrakte Fabrik. Diese beauftragt eine konkrete Fabrik mit Herstellung des Java-Objekts für eine Ebene. In der Implementierung auf das NRZM erledigt das z.B. die `NRZMMapLayerFactory`. Anhand des Namens einer Ebene, die in dem MapServer konfiguriert ist, erzeugt sie von `MapLayer` abgeleitete Objekte.

So gibt ein Aufruf der Form

`getMapLayerFactory(MapLayerFactory.NRZM).getMapLayer("serogroups", "Serogroups", ...)`
z.B. ein Objekt der Klasse `SerogroupsLayer` zurück. Diese enthält einen Parameter (siehe auch Abschnitt 5.2.1 und 5.2.2), der das Filtern nach der Serogruppe ermöglicht.

Falls die Fabrik unter einem Namen keine spezielle Ebenenimplementierung vorsieht,

6. Implementierungsdetails

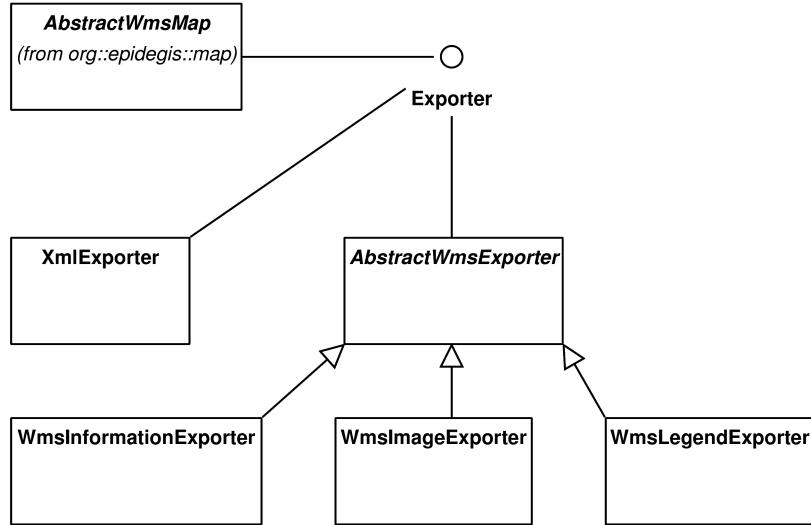


Abbildung 6.3.: Klassendiagramm des Exporter-Konzeptes [29]

liefert sie ein *MapLayer*-Objekt zurück. Diese Standard-Ebene reagiert nur auf Änderungen an dem Parameter, der in bestimmte Gebiete der Karte zoomt.

Prinzipielle Unterstützung von Benutzergruppen

Diese Implementierung erlaubt es dem Entwickler, schnell neue Ebenen mit speziellen Parametern zu definieren. Weiterhin könnten bei unterschiedlichen Benutzergruppen verschiedene Fabriken eingesetzt werden, die beispielsweise für den internen Einsatz eine Ebene mit detaillierten und für die Öffentlichkeit mit groben Filtern produzieren.

Die Erweiterung könnte zudem auch eine Ebene höher ansetzen. Das System erzeugt für jeden Benutzer ein Objekt des Typs *AbstractWmsMap*. Dieses stellt die eigentliche Schnittstelle zum MapServer dar. Je nach konkreter Implementierung könnten hier z.B. unterschiedliche "MapFiles" als Karte eingesetzt werden.

Sobald sich ein Benutzer authentifiziert, kann EpiDeGIS z.B. eine *PublicMap* durch eine *NRZMMap* austauschen und die Anwendung neu laden. Implementiert ist bisher nur die *NRZMMap*. Dieses Design ermöglicht jedoch den Einsatz von Benutzergruppen.

6.1.2. Export der Karte in unterschiedliche Dateiformate

Nach der Initialisierung liegt in der Benutzer-Session ein Kartenobjekt vor. Dieses soll als View in verschiedenen Formaten ausgegeben werden (siehe auch Abschnitt 5.1.2).

Die Implementierung dieser Funktion wendet dabei ein von Allen Holub [59] beschriebenes Konzept an. Dieses ersetzt den in der Anfangsphase eingesetzten Ansatz, der z.B. XML-Repräsentation der Karte umständlich mittels JSP und zu vielen Get-Methoden erzeugt hat. Ein früherer Artikel von Holub [58] hat dabei zu einer weit gehenden Vermeidung von Get- und Set-Methoden in der Klasse *AbstractMap* und deren Ableitungen angeregt.

Klassenname	Aufgabe
<i>XmlExporter</i>	erzeugt eine XML-Repräsentation der Karte
<i>AbstractWmsExporter</i>	nutzt den MapServer zum Kartenexport
<i>WmsInformationExporter</i>	liefert Attributinformationen zu einem Punkt auf der Karte
<i>WmsImageExporter</i>	exportiert die Karte als Bild und implementiert das Caching
<i>WmsLegendExporter</i>	gibt die Kartenlegende als Bild zurück

Tabelle 6.1.: Klassen im Paket Export und ihre Aufgaben

Die Klasse *AbstractWmsMap* stellt eine Schnittstelle namens Exporter bereit. Über die Methode `export(Exporter e)` übergibt die Karte ihre internen Daten an eine Klasse, die das Exporter-Interface implementiert. Das Paket *Export* (siehe Abbildung auf Seite 50) enthält in EpiDeGIS alle konkreten Implementierungen dieser Schnittstelle (Tabelle 6.1).

Die Webanwendung erzeugt z.B. das XML-View mit einem *XmlExporter*-Objekt. Das Programmfragment in Listing 6.1 zeigt, wie einfach sich das Exporter-Prinzip anwenden lässt.

Listing 6.1: Codefragment zum Einsatz der *XmlExporter*-Klasse

```

1 XmlExporter xmlExporter = new XmlExporter();
2 map.export(xmlExporter);
3 xmlInputStream = xmlExporter.getInputStream();
4 contentType = xmlExporter.getContentType();

```

6.1.3. Zustandsänderung durch Aktionen in Struts

Abschnitt 5.1.3 stellt das Konzept vor, in dem der Controller in der RIA alle Eingriffe in den Systemzustand über den serverseitigen Controller ausführen muss. Struts (siehe 4.3.1) nimmt als Front-Controller alle Anfragen an den Java Server entgegen und beauftragt mit deren Bearbeitung so genannte “Actions” [37].

Beispiel einer implementierten Aktion

MoveMapFormBean <i>(from org::epidegis::webapp)</i>
<code>-xoffset:double</code>
<code>-yoffset:double</code>
<code>+getXOffset():double</code>
<code>+setXOffset(xoffset:double):void</code>
<code>+getYOffset():double</code>
<code>+setYOffset(yoffset:double):void</code>

Abbildung 6.4.: “ActionForm Bean”-Klasse für die *MoveMapAction* [29]

Eine an der Schnittstelle zur Verfügung stehende Aktion heißt “MoveMap”. Über den URL <http://localhost/MoveMap.do?xoffset=0.5&yoffset=0.5> wird eine Verschiebung des Kartenausschnitts um 50% der Breite nach links¹ und um 50% der Höhe nach unten durchgeführt. Zur Implementierung der Aktion sind drei Schritte notwendig:

¹Anm. des Autors: Beim Schreiben ist mir der kleine Implementierungsfehler aufgefallen. Üblicherweise würde man bei einem positiven Offset eine Bewegung nach rechts erwarten.

6. Implementierungsdetails

Umsetzung in der Programmlogik Dieser Schritt hat noch nichts mit dem Struts-Framework zu tun. Um den Kartenausschnitt verschieben zu können, muss die *AbstractWmsMap* zuerst eine entsprechende Methode bereitstellen. Der Aufruf von *map.move(0.5, 0.5)* führt die Aufgabe aus. Die Methode verschiebt dazu einfach die Grenzkoordinaten des Kartenausschnittes um den entsprechenden Anteil.

Implementierung einer “Action” und einer “ActionForm Bean” Eine in Struts definierte Aktion leitet sich immer von der *Action*-Klasse des Frameworks ab. Sie ist die Schnittstelle zwischen einem eingehenden HTTP-Request und der Programmlogik (s. o.). Die Daten des Requests verpackt Struts automatisch in so genannte “ActionForm Beans” (Abbildung 6.4).

Die Struts-Aktion verwendet diese Java-Container um auf HTTP-Parameter zuzugreifen. Das “ActionForm Bean”-Konzept definiert noch einiges mehr, diese Funktionalität verwendet EpiDeGIS jedoch nicht.

Die Implementierung der *MoveMapAction* nimmt die mittels HTTP-Get übertragenen Werte für *xOffset* sowie *yOffset* und ruft damit *move()* aus der *AbstractWmsMap* auf. Listing 6.2 zeigt mit der *execute*-Methode den wesentlichen Teil der Aktion.

Listing 6.2: Execute-Methode der *MoveMapAction*-Klasse

```
public ActionForward execute([...])
2     throws Exception
{
4     String forward = "error";
5     MoveMapFormBean move = (MoveMapFormBean) form;
6     AbstractWmsMap map = (AbstractWmsMap) request.
7         getSession().getAttribute("map");
8     if (map != null)
9     {
10        map.move(move.getXoffset(), move.getYoffset());
11        forward = "success";
12    }
13    return (mapping.findForward(forward));
14}
```

Definition des “Action Mappings” und der “Form Bean” Damit das Struts-Framework weiß, welche Aktion es aufrufen muss, wenn eine HTTP-Anfrage über den URL *../MoveMap.do* eingeht, benötigt es einen Eintrag in seiner zentralen Konfigurationsdatei. Listing 6.3 zeigt den entsprechenden Ausschnitt aus dieser Datei.

Listing 6.3: Definition einer Aktion im Struts-Framework

```
<form-bean type="org.epidegis.webapp.MoveMapFormBean"
2             name="move"/>
[...]
4 <action path="/MoveMap"
5       type="org.epidegis.webapp.MoveMapAction"
6       name="move" scope="request">
```

```

<forward name="success" path="/WEB-INF/results/
    xmlDefaultSuccess.jsp"/>
8 <forward name="error" path="/WEB-INF/errors/
    xmlDefaultError.jsp"/>
</action>
```

In dieser minimalen Konfiguration muss lediglich die “Form Bean”- und die “Action”-Klasse festgelegt werden. Der angegebene Pfad legt fest, unter welchem Namen die Aktion über HTTP erreichbar ist.

In EpiDeGIS implementierte Aktionen

Tabelle 6.2 listet alle Aktionen auf, die in diesem System implementiert sind. Nur mittels dieser Schnittstellen können externe Anwendungen wie z.B. die OpenLaszlo-Applikation den Zustand des Servers ändern bzw. erfragen.

6.2. Definition von Karten für den MapServer

Der MapServer steuert alle Details der Kartenerzeugung über die “MapFile”. Im Bezug auf die Zeilenzahlen in Listing D.2 erlaubt diese u. a. folgende Einstellungsmöglichkeiten:

- 3 Grenzen des vom MapServer dargestellten Koordinatenbereiches (*minx, miny, maxx, maxy*)
- 4,5 Einbindung externer Dateien mit Font- und Symboldefinitionen
- 6-14 Metaangaben zu dem bereitgestellten WMS
- 15-24 Definition der angebotenen Ausgabeformate, dem Standard-Bildtyp, der Standard-Bildgröße und der Hintergrundfarbe. Die Größe der auszugebenden Bitmap lässt sich zur Laufzeit auch über den URL angeben.
- 28-37 Einstellungen zu der Legende

Alias	“Action”-Klasse	Funktion
Mapbrowser.do	(Weiterleitung)	öffnet die HTML-Seite mit der eingebetteten RIA
Map.do	CreateMapAction	initialisiert die Anwendung und gibt die Karte als XML zurück
GetMap.do	GetMapAction	liefert eine Bitmap der aktuellen Karte
GetMapLayer.do	GetMapLayerAction	liefert eine Bitmap einer einzelnen Kartenebene
GetLegendGraphic.do	GetLegendGraphicAction	liefert eine Bitmap der Legende einer Ebene
MoveMap.do	MoveMapAction	verschiebt den Kartenausschnitt
SetParameter	SetParameterAction	ändert den Wert eines Parameters
ToggleLayerState	ToggleLayerStateAction	schaltet eine Ebene an bzw. aus
GetFeatureInfo	GetFeatureInfoAction	liefert Attributinformationen zu einem Punkt auf der Karte
Mapbrowser.do	(Weiterleitung)	öffnet die HTML-Seite mit der eingebetteten RIA

Tabelle 6.2.: Implementierte Struts-Aktionen in EpiDeGIS-Webanwendung

6. Implementierungsdetails

- 38-83 *Definition einer Kartenebene:*
- 41-42 Angaben zur Datenbankverbindung (Typ, Benutzer, Passwort und Host)
- 43-47 Definition der Datenquelle
- 48 Geometriertyp der Ebene (z.B. Punkte)
- 49 Datenbankspalte, die den zu der Geometrie darstellenden Text (z.B. den Stadtnamen) enthält
- 50-53 WMS-Metainformationen zu dieser Ebene
- 54-56 Projektion, in der die Quelldaten vorliegen
- 57-58 Größe der dargestellten Symbole und Text-Label
- 59-82 *Angaben zur Darstellung der Geometrien*
- 60 Voraussetzung an die Attributdaten aus der darzustellenden Datenmenge (z.B. Einwohnerzahl einer Stadt größer als 1 Million), damit diese Angaben verwendet werden
- 64-70 Stil, in dem die Geometrien dargestellt werden (z.B. rotes Quadrat der Größe 8-16 Pixel mit einem schwarzen Rand)
- 71-81 Definition der Schriftart der angezeigten Textinformation

Die wichtigste Einstellung ist die Definition der Datenquelle. Diese erlaubt die direkte Abfrage der Datenbank unter Angabe dynamischer Parameter. Listing 6.4 zeigt die Verwendung der in Abschnitt 5.3.1 bereits als Beispiel eingesetzten SQL-Abfrage in einem *DATA*-Element.

Listing 6.4: Definition der Datenquelle in der MapFile

```
1 DATA "the_geom from (
2   SELECT case_id, the_geom
3   FROM cases
4   WHERE reportdate
5     BETWEEN to_timestamp('%fromDate%', 'MMYYYY')
6       AND to_timestamp('%toDate%', 'MMYYYY')
7 ) AS foo USING SRID=4326 USING UNIQUE case_id"
```

Das Listing zeigt die Metaangaben, die der MapServer benötigt, um die SQL-Query als Datenquelle verwenden zu können. **USING SRID=4326** gibt an, in welchem Koordinatensystem die Daten vorliegen (4326 entspricht geografische Länge und Breite), **USING UNIQUE case_id** legt fest, welche Spalte einen eindeutigen Schlüsselwert enthält.

Die folgenden Abschnitte zeigen, wie ausgewählte Ebenentypen für die epidemiologischen Karten des NRZM im Hinblick auf die SQL-Abfragen implementiert sind.

6.2.1. Inzidenzdarstellung

EpiDeGIS kann die Inzidenz für alle in der Datenbank erfassten Gebiete berechnen und darstellen. Hierzu müssen zu einem Bereich (wie z.B. einem Bundesland) lediglich die

Einwohnerzahlen vorliegen.

Da bei der Implementierung für die Meningokokkenerkrankung alle registrierten Erkrankungsfälle auch Neuerkrankungen sind, errechnet sich die Jahresinzidenz für ein Gebiet durch folgende Formel:

$$\frac{\text{Anzahl der Erkrankungsfälle in einem Jahr} \times 100.000}{\text{Anzahl der Einwohner}}$$

Dynamische Inzidenzberechnung

Nach dem in Abschnitt 5.1.1 vorgestellten Datenbankkonzept, das flexible Hierarchien und Gebiete vorsieht, ist es nicht möglich, die Inzidenz in vorher festgelegten Tabellen abzuspeichern. Außerdem würden starre Tabellen es nicht ermöglichen, Inzidenzberechnungen über andere Zeiträume anzustellen. Die Inzidenzberechnung für beliebige Gebiete und frei wählbare Zeiträume steht vor der Aufgabe, dynamisch feststellen zu müssen, ob ein Erkrankungsfall in einer geografischen Region aufgetreten ist.

Gelöst ist diese Problemstellung mittels der Topologie. Jeder Fall in der Datenbank besitzt eine Koordinate und jedes Gebiet ist ein Polygon (jeweils in der Spalte `the_geom`). Die topologische Beziehung “Erkrankungsfall ist in einem Gebiet aufgetreten” liefert wahr oder falsch zurück. Die Abfrage in Listing 6.5 errechnet beispielsweise die Anzahl aller Fälle in Bayern.

Listing 6.5: SQL-Query zur Ermittlung der Fallzahl in Bayern

```

1 SELECT count(*) FROM areas , cases
  WHERE areas.the_geom && cases.the_geom
3   AND areas.identifier = 'Bayern'
   AND contains(areas.the_geom , cases.the_geom)

```

Diese spezifische Query lässt sich durch weitere Bedingungen leicht so erweitern, dass sie die Fallzählung für jedes Gebiete vom Typ² “Bundesland” auf einen bestimmten Zeitraum begrenzt durchführt.

Preis der Dynamik

Leider benötigt die Funktion `contains()` sehr viel Rechenzeit. Tabelle 6.3 führt diese für drei in Deutschland vorkommende Hierarchieebenen auf.

Gebietstyp	Datenbankeinträge	Rechendauer
Bundesland	16	7609 ms
Landkreis	439	23156 ms
fünfstellige Postleitzahlen	8270	22563 ms

Tabelle 6.3.: Rechenzeiten zur Zählung der Erkrankungsfälle in einem bestimmten Gebietstyp

²siehe auch Abbildung 5.2 auf Seite 45

6. Implementierungsdetails

Auch wenn weitere einschränkende Bedingungen (z.B. Zeitraum) die Anzahl der Fälle und damit die Rechenzeit verringern, ist sie noch zu lang, um daraus dynamisch Karten zu erzeugen.

Mit einer Zuordnungstabelle lässt sich die Abfragedauer entscheidend verbessern. Die Zuordnung eines Erkrankungsfalles zu einem Gebiet muss dabei nur erneuert werden, wenn neue Gebiete bzw. Falldaten in die Datenbank eingetragen werden. Tabelle 6.4 zeigt die Messergebnisse mit Hilfe der Zuordnungstabelle `contains_area_case`. Diese liegen in dem brauchbaren Bereich von unter einer Sekunde.

Gebietstyp	Datenbankeinträge	Rechendauer
Bundesland	16	203 ms
Landkreis	439	328 ms
fünfstellige Postleitzahlen	8270	391 ms

Tabelle 6.4.: Durch Zuordnungstabelle optimierte Rechenzeit (vgl. Tabelle 6.3)

Finale Implementierung

Die Datenquellendefinition für die Inzidenz-Ebene für den Einsatz im NRZM ist in Listing D.3 auf Seite 99 abgedruckt.

Die Abfrage führt folgende Aufgaben aus:

- Zeile 3-6 Diese vier Zeilen berechnen die Jahresinzidenz zwischen zwei begrenzenden Daten von monatlicher Auflösung. Die Berechnung führt eine einfache Inter- bzw. Extrapolation aus, wenn der Zeitraum größer oder kleiner einem Jahr ist.
- Z. 12-14 Die Inzidenzberechnung erfolgt nur für Fälle der über den Parameter `%SEROGROUPS%` angegebenen Serogruppe(n) (z.B. "B", "C"), deren...
- Z. 14-17 ... Meldedatum (als Monate umgeschrieben) zwischen den angegebenen Grenzdaten liegt.
- Zeile 18 Die Abfrage verwendet nur Gebiete der Hierarchieebene, die über den Wert des Parameters mit dem Namen `%INCITIER%` festgelegt ist.

Die Inzidenz-Ebene stellt mittels dieser Implementierung die Inzidenz in einem beliebigen Gebiet in der Datenbank durch eine Färbung dar. Die Berechnung arbeitet unabhängig von den zugrunde liegenden Grenzpolygonen. Dank der Zuordnungstabelle wird die Karte trotz dieser Dynamik schnell erzeugt.

6.2.2. Darstellung von Clustern

Die Cluster-Ebene stellt die von der SaTScan-Komponente identifizierten Erkrankungshäufungen dar. Ein Cluster besitzt zwei für die geografische Darstellung wichtige Attribute: Das Gebiet, in dem der Mittelpunkt liegt, und einen Radius, welcher die Ausbreitung markiert.

PostGIS erzeugt Kreise

Sobald SaTScan einen Cluster entdeckt hat, legt EpiDeGIS diesen in der Datenbank ab. An dieser Stelle wäre es ohne weiteres möglich, den Umkreis als Geometrie mit in der Datenbank zu speichern. Um Datenredundanz zu vermeiden, geht die Implementierung in eine andere Richtung.

Der Clusterumkreis ist letztlich nichts anderes als ein Kreis um den Mittelpunkt des zentralsten Gebietes mit dem entsprechenden Radius in Kilometern. Das PostGIS Benutzerhandbuch [30] definiert die folgende Methode:

Buffer(geometry,double,[integer]) Gibt eine Geometrie zurück die alle Punkte beinhaltet, deren Abstand von der im ersten Argument angegebenen Geometrie kleiner oder gleich dem Wert im zweiten Argument ist. Berechnungen werden im Koordinatensystem der Geometrie ausgeführt. Der optionale dritte Parameter gibt die Anzahl der Segmente an, mit denen ein Viertelkreis angenähert wird (Standard ist 8).

Diese Funktion leistet genau das Gewünschte, wenn sie mit dem Mittelpunkt des Clusters und dessen Radius aufgerufen wird. Listing 6.6 zeigt den entsprechenden Ausschnitt aus der SQL-Query.

Listing 6.6: Erzeugung von Kreisen mit Hilfe von PostGIS

```

1 transform(
2   setSRID(
3     buffer(
4       transform(centroid(the_geom), 31467), radius*1000)
5       , 31467)
6       , 4326)
```

Die Erdkrümmung ist dafür verantwortlich, dass neben den eigentlichen Methoden `centroid` und `buffer` noch zwei weitere benötigt werden.

In dem auf geometrische Länge und Breite basierendem Koordinatensystem mit dem Code 4326 würde wegen der unterschiedlichen Abstände von Längen- und Breitenkreisen eine Ellipse anstelle des Kreises entstehen. Aus diesem Grund wird in Zeile vier der Mittelpunkt nach Gauß-Krüger transformiert, um dort den Kreis mit dem Radius³ zu erzeugen. Anschließend wird der “Buffer” wieder rücktransformiert.

Sonderfall: Radius 0 Kilometer

SaTScan gibt bei Clustern, die nur in einem Landkreis auftreten, einen Radius von 0 Kilometern zurück. Mit der obigen Implementierung wären sie auf der Karte nicht darstellbar. Mittels einer Fallunterscheidung im SELECT-Statement der Cluster-Ebene werden diese Cluster gesondert behandelt. Dabei wird ein Kreis erzeugt, der den Landkreis annähernd umschließt.

³die Einheit im Gauß-Krüger-Koordinatensystem ist Meter, daher die Multiplikation mit 1000

6. Implementierungsdetails

6.2.3. Streuung von Punktinformationen

Bei der NRZM-Umsetzung der Serogruppen-Ebene, die alle Meningokokken-Erkrankungsfälle als Symbole auf der Karte anzeigt, hat sich in Verbindung mit dem MapServer ein Problem ergeben:

Wenn mehrere Fälle in der gleichen Stadt aufgetreten sind, hat der MapServer diese nur mit einem Symbol an der entsprechenden Koordinate dargestellt. Die Bewertung von Häufungen bestimmter Serogruppen ist dadurch nicht möglich, da der Anwender nicht unterscheiden kann, ob z.B. einer oder zehn Fälle aufgetreten sind.

Da der MapServer innerhalb einer Ebene kein Mittel bereit stellt, Punktinformationen um die eigentliche Koordinate zu streuen, muss die Datenbankabfrage diese Aufgabe erledigen.

Listing 6.7: Zufällige Streuung von Punktinformationen über die SQL-Query

```
CREATE OR REPLACE VIEW cases_random AS
2  SELECT setsrid(
     translate(
4       cases.the_geom,
      random() / 30, random() / 30, 0), 4326) AS the_geom,
6  [...]
FROM cases;
```

Listing 6.7 zeigt die Definition eines “Views”, welches die Koordinaten aller in der `cases`-Tabelle erfassten Fälle um einen zufälligen Betrag in x- und y-Richtung transformiert. Ein “View” bezeichnet dabei in SQL eine nach bestimmten Kriterien gefilterte oder veränderte virtuelle (An)sicht einer oder mehrerer physikalischer Tabellen.

6.3. Clientanwendung mit OpenLaszlo

Der folgende Abschnitt geht auf zwei Details der Anwendungsprogrammierung mit OpenLaszlo ein. Er bietet keine Anleitung zur Programmierung in der LZX-Sprache, sondern vielmehr eine Vorstellung von zwei interessanten Konzepten. Details zur Programmierung in OpenLaszlo finden sich in der umfassenden Dokumentation [20]. Diese wird jedoch für das Verständnis des Folgenden nicht benötigt.

6.3.1. Dynamischer Aufbau der Benutzeroberfläche

In Abbildung 6.1 auf Seite 56 ist zu erkennen, dass sich die Benutzeroberfläche der RIA mittels der XML-Repräsentation des Serverzustandes initialisiert. Die Abschnitte 5.2.1 und 5.2.2 behandeln das Konzept von dynamischen Ebenen und Parametern.

Tim	Nice guy.
Ron	Not so bright.
Kelly	Tall.
Adam	Bombastic.

Abbildung 6.5.: Ausgabe des Programms in Listing 6.8

Replikation durch XML-Daten

OpenLaszlo ermöglicht es durch eine Technik namens “databinding”, dynamisch Elemente in der Benutzeroberfläche eine Flash-Anwendung zu erzeugen. Listing 6.8 zeigt ein anschauliches Beispiel [84] dieser Funktion:

Listing 6.8: Replikation von Elementen in OpenLaszlo durch “databinding”

```

1 <canvas layout="spacing: 2">
2   <dataset name="myData">
3     <people>
4       <Tim> Nice guy. </Tim>
5       <Ron> Not so bright. </Ron>
6       <Kelly> Tall. </Kelly>
7       <Adam> Bombastic. </Adam>
8     </people>
9   </dataset>
10  <view datapath="myData:/people/*" layout="axis:'x'">
11    <text datapath="name()" />
12    <text datapath="text()" />
13  </view>
</canvas>
```

- Zeilen 2-9 definieren inline im Programm ein “dataset”, das beliebige XML-Daten enthalten kann.
- Z. 10-13 erzeugen ein so genanntes “view”. Der Begriff bezeichnet in OpenLaszlo ein später in der Anwendung sichtbares Containerelement. Das Attribut **datapath** dieses Elements löst die Replikation aus.
- Z. 11+12 sorgen dafür, dass die Daten aus dem XML in der Flash-Anwendung angezeigt werden.

Abbildung 6.5 zeigen die Ausgabe der entstehenden Flash-Anwendung. Sie verdeutlicht das Prinzip des “databinding”:

Über das Attribut **datapath** weiß die Anwendung, dass das “view” in Zeile 10-13 explizit mit den Daten unter dem Pfad **myData:/people/*** verbunden ist. Die Sprache mit der OpenLaszlo den Pfad beschreibt ist XPath [52].

Der in Laszlo integrierte Replikationsmechanismus arbeitet mit diesen Pfadangaben. Das “view” wird von ihm für jedes Datenelement kopiert, das sich unter **<people>** befindet. In dem Beispiel werden dabei jeweils für Tim, Ron, Kelly und Adam zwei Textelemente horizontal aneinander gereiht (**layout="axis:'x'"**).

6. Implementierungsdetails

Das erste schreibt den Namen des XML-Elementes und das zweite den Inhalt. Die Pfadangaben innerhalb des Containers werden dabei relativ zu dessen Pfad interpretiert.

Durch die Angabe `layout='spacing:2'` ordnet das `<canvas>`-Element die vier replizierten `<view>`'s untereinander (die Standardachse ist y) mit einem Abstand von zwei an.

XML Datenquellen

OpenLaszlo nimmt die Daten für die Replikation aus drei möglichen Quellen entgegen. Entweder Inline-XML (s. o.), auf dem Webserver liegende XML-Dateien oder XML-Daten, die über einen URL geladen werden.

EpiDeGIS setzt die letztgenannte Quelle ein. Die vom Server angebotene Karte wird als XML von der Flash-Anwendung eingelesen und zum Aufbau der grafischen Benutzeroberfläche verwendet.

Implementierung am Beispiel eines Auswahlparameters

Listing D.4 auf Seite 100 zeigt die XML-Repräsentation einer einfachen Karte. Am Beispiel des Kartenparameters mit dem Namen `areaId` (Zeile 3-10), soll die Replikation von Elementen der Benutzeroberfläche in EpiDeGIS demonstriert werden.

Listing 6.9 enthält die beiden Programmstellen, die dafür verantwortlich sind, dass in der GUI ein Auswahlfeld für den zu betrachtenden Kartenausschnitt erscheint (siehe Abbildung 5.5):

Zeile 2-7 definieren eine Datenquelle, deren Ziel die Aktion von Struts ist, welche die Karte als XML zurückgibt (siehe Tabelle 6.2). In Zeile vier wird nach obigen Prinzip ein Element der Klasse `selectparameter` so oft repliziert, wie es `<selectparameter>`-Elemente unter den Wurzelement `<map>` in Listing D.4 auf Seite 100 gibt. In diesem Fall ist es nur der eine, welcher die Auswahl des Gebietes ermöglicht.

Listing 6.9: Fragment des Kartenbetrachter-Quellcodes zur dynamischen Erzeugung der GUI

```
[...]
2 <include href="includes/parameters.lzx"/>
3 <datasource name="ds">
4   <method event="oninit">
5     this.mapserverData.setSrc(baseUrl() + 'Map.do');
6   </method>
7   <dataset type="http" name="mapserverData"/>
8 </datasource>
9 [...]
10 <selectparameter width="${parent.width}">
11   datapath="mapserverData:/map/selectparameter"
12   mapview="${classroot}"/>
13 [...]
```

Die Klasse *selectparameter* ist in einer externen LZX-Quelldatei implementiert. Sie erzeugt die eigentliche Dropdown-Liste anhand der <value>-Elemente und informiert den Rest der Applikation, wenn der Anwender den Parameterwert ändert.

6.3.2. Kommunikation zwischen Laszlo und Struts

Die Laszlo-Applikation kommuniziert mit Struts über die in Abschnitt 6.1.3 angegebenen Aktionen. Der Server antwortet dabei auf Anfragen mit XML-Fragmenten. Diese informieren über den Erfolg, Misserfolg und Ergebnisse der gewünschten Zustandsänderung. Die RIA verarbeitet die Serverantworten mittels des gleichen Prinzips, das schon bei dem dynamischen Aufbau der GUI in Abbildung 6.3.1 Anwendung findet.

Listing 6.10 zeigt den Programmteil des Kartenbetrachters, in dem die *SetParameter*-Aktion von Struts (siehe Tabelle 6.2) angewendet und deren Ausgabe entgegengenommen wird:

Listing 6.10: Fragment des Kartenbetrachter-Quellcodes zur Kommunikation mit Struts

```

1 [...] 
2 <datasource name="ds">
3   [...]
4     <dataset type="http" name="SetParameter"/>
5 </datasource>
6   [...]
7 var d= canvas.ds.SetParameter;
8 var p=new LzParam();
9 p.addValue("name", strName, true);
10 p.addValue("value", strValue, true);
11 d.setQueryString(p);
12 d.doRequest();
13 [...]
14 <datapointer id="setp" xpath="SetParameter:/result">
15   <method event="ondata" args="d">
16     <!-- Behandelt die Antwort auf die SetParameter -Aktion -->
17   </method>
18 </datapointer>
19 [...]
```

Zeile 4 definiert wie Zeile 7 in Listing 6.9 die Datenquelle.

Z. 7-12 zeigen den Ausschnitt einer Methode, die eine Anfrage an die Datenquelle stellt. Zwei HTTP-Parameter werden für die Query mit angegeben. Struts verändert mittels der *SetParameter*-Aktion den Wert eines Parameters (*name*) auf den angegebenen Wert (*value*).

Z. 14 definiert einen Zeiger auf den Pfad der in Zeile 4 angegebenen Datenquelle.

Z. 15 Sobald sich die Daten ändern, auf die der *datapointer* zeigt, wird das Ereignis *ondata* ausgelöst und die entsprechende Methode aufgerufen.

6. Implementierungsdetails

Listing 6.11 zeigt die Antwort der Aktion `../SetParameter.do?name=SEROGROUPS&value='B'`. Diese setzt den Parameter mit dem Namen *SEROGROUP* auf den Wert "B". Der Server nimmt die Zustandsänderung an und antwortet mit einem bestimmten XML-Fragment.

Listing 6.11: Antwort des Servers auf eine *SetParameter*-Aktion

```
1 <result>
  2   <update>serogroups </update>
3   <update>incidence </update>
</result>
```

Dieses enthält als Wurzel das Element `<result>`. Darin folgen zwei `<update>`-Elemente. Diese deuten der Laszlo-Anwendung an, dass sich durch die Aktion sowohl die Serogruppen- als auch die Inzidenz-Ebene geändert hat.

Sobald die Serverantwort bei Laszlo ankommt, löst sie das ondata-Ereignis aller mit der Datenquelle SetParameter verknüpften `datapointer` aus. Auch die Methode in Listing 6.10 , Zeile 14 wird angestoßen und kann auf die Antwort reagieren. Im obigen Beispiel wird sie die beiden Ebenen neu laden.

6.4. Serverseitige Clusteridentifizierung mittels SaTScan

Mit SaTScan setzt EpiDeGIS ein Werkzeug ein, das bereits viele Epidemiologen zur Identifizierung von spatio-temporalen Clustern verwenden. SaTScan lässt sich mit einer grafische Benutzeroberfläche bedienen, über die sich alle Parameter der Analyse einstellen lassen. Nach einem Durchlauf gibt es die Ergebnisse in einem Bericht aus (siehe Listing D.1 auf Seite 97).

Für den Einsatz am eigenen Rechner ist die Analyse mit Hilfe einer GUI sehr komfortabel, im automatisierten serverseitigen Einsatz jedoch ungeeignet. Da sich SaTScan aber auch über die Kommandozeile ausführen lässt, ist ein automatisierter Aufruf aus Java heraus möglich.

Terminplanung von Analysen

Das Paket Satscan im Softwaresystem EpiDeGIS implementiert die Verbindung von SaTScan und Java. Für die Implementierung der automatisierten Ausführung war es wichtig, dass der Server die Clusterberechnungen zu einer beliebigen Zeit ausführen kann. Die hardwarelastigen Berechnungen können so in der Nacht getätigter werden. Das bedeutet, dass die Ablaufplanung einen Analysevorgang unter Umständen vor dem Ende abbricht. Dies darf unter keinen Umständen zu einer Unterschlagung einer Berechnung führen.

Die Implementierung des automatisierten SaTScan-Aufrufs speichert aus diesem Grund alle geplanten Analysen in der Datenbank. Die Grundlage bildet dabei eine Tabelle, die

6.4. Serverseitige Clusteridentifizierung mittels SaTScan

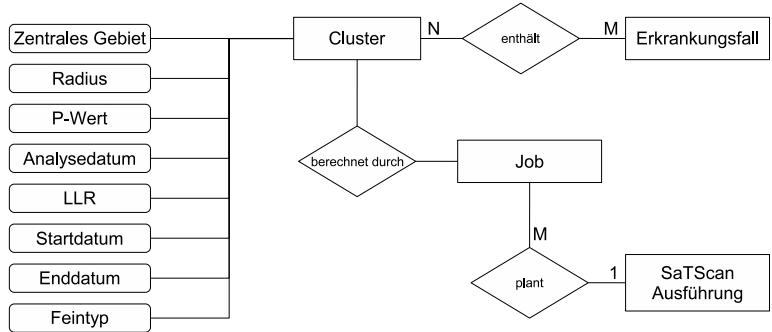


Abbildung 6.6.: Datenbankstruktur zum Planen von SaTScan-Ausführungen und Speichern der Ergebnisse

so genannte “SaTScan-Jobs” enthält. Hierbei handelt es sich um geplante Ausführungen, die in bestimmten Zeitabständen durchgeführt werden sollen. Der Job speichert neben der Ausführungsart (wöchentlich, monatlich, jährlich) und dem letzten Ausführungsdatum, auch alle in diesem Job für die Ausführung von SaTScan verwendeten Einstellungen.

Ein eingesetzter Job für das NRZM lautet beispielsweise “weekly prospective” (siehe Abschnitt 4.2.4). In diesem ist festgelegt, dass EpiDeGIS jede Woche eine prospektive Clusteridentifizierung durchführen soll. Diese muss jeden einzelnen in einem Beobachtungszeitraum von 100 Tagen aufgetretenen Feintyp das Programm SaTScan ausführen.

Ausführung von SaTScan

Eine weitere Tabelle verwaltet alle diese atomaren Ausführungen. Sie speichert in drei Spalten jeweils die Jobnummer, das ursprünglich geplante Ausführungsdatum und die Nummer des zu analysierenden Feintyps. Die Jobverarbeitung füllt die Tabelle mit den Daten der für die zur Erledigung eines Jobs nötigen Analysen.

Die eigentliche Ausführung von SaTScan geht folgendermaßen von statt:

Die erste auf dem Ablaufplan stehende Analyse wird aus der Tabelle abgefragt. EpiDeGIS kann dabei durch die Jobnummer, den Feintyp und das Datum rückschließen, welche Erkrankungsfälle es für die Clusteridentifizierung berücksichtigen muss. Anhand dieser Daten lassen sich die von SaTScan benötigten Eingabedateien generieren.

SaTScan wird von Java aus als externes Programm aufgerufen. Es führt die Analyse mit Hilfe der Eingabedateien aus. Nach einem erfolgreichen Lauf generiert es verschiedene Ergebnisdateien [65]. EpiDeGIS nimmt alle Informationen aus den jeweiligen Dateien und erzeugt daraus ein Objekt der Klasse *SatscanCluster*. Über diesen Container wird das Ergebnis in die Datenbank übertragen.

Erst an diesem Punkt ist die Analyse eines Feintyps beendet und der entsprechende Eintrag aus der Tabelle der geplanten Ausführungen wird gelöscht. Abbildung 6.6 zeigt die Relationen in der Datenbank. Das zentrale Gebiet bzw. der Feintyp in der Cluster-Tabelle, sind dabei Fremdschlüssel für andere, hier nicht abgebildete Tabellen.

6. Implementierungsdetails

Vorteile und Implementierung

Die atomare SaTScan-Analyse hat den Vorteil, dass sie die zum Teil zeitintensiven Berechnungen Stück für Stück ausführt. Eine retrospektive Analyse über einen großen Zeitraum kann insgesamt mehrere Tage benötigen. Der Anteil für einen Feintyp beträgt dabei nur eine halbe Stunde.

7. Zusammenfassung und Ausblick

7.1. Zusammenfassung

Die vorliegende Diplomarbeit hat den Prozess beschrieben, in dem ein Web-basierter GIS-Service zur Überwachung von Infektionen entstanden ist.

Dessen Entwicklung wurde in enger Zusammenarbeit mit dem Nationalen Referenzzentrum für Meningokokken durchgeführt. Das Ergebnis dieser Zusammenarbeit ist eine einsatzfähige Version einer über das Internet zugänglichen Software, die es ermöglicht, die Meningokokken-Erkrankungssituation in Deutschland automatisiert zu visualisieren und zu überwachen.

Web-basierter, einer breiten Benutzergruppe zugänglicher, epidemiologischer GIS-Service

Das in einem Internetbrowser ausführbare Programm ist eine mittels Flash realisierte “Rich Internet Application”. Das Flash-Format sichert dabei eine Erreichbarkeit von 97% aller Client-Rechner. RIAs bieten dem Anwender eine von Desktop-Anwendungen bekannte Benutzbarkeit.

Die Applikation stellt dabei viele Interaktionsmöglichkeiten mit dem zugrunde liegenden epidemiologischen Geoinformationssystem zur Verfügung. Anwender können hierdurch unterschiedliche thematische Karten visualisieren. Diese bestehen aus einzelnen Ebenen zur Darstellung der verschiedensten Informationen:

- Inzidenz der Erkrankungsfälle durch Einfärbung von Bundesländern oder Landkreisen
- Bevölkerungsdichte, ebenfalls für Bundesländer und Landkreise
- Verteilung der Fälle eines ausgewählten Meningokokken-Feintyps
- Verteilung der Fälle einer oder mehrerer ausgewählter Serogruppen
- Ergebnisse der wöchentlich prospektiven und monatlich retrospektiven Clusteranalyse
- Statische Informationen wie Grenzen oder Städte

Weiterhin kann der Anwender die Darstellung der Karte beeinflussen indem er...

- einen Zeitraum auswählt, für den die Serogruppen, Feintypen und die Inzidenz dargestellt werden soll.

7. Zusammenfassung und Ausblick

- eine Altersgruppe angibt, auf welche die Darstellung der Fälle (Serogruppen bzw. Feintypen) begrenzt ist.
- ein Bundesland auswählt, für das die Kartendarstellung vergrößert wird.

Die intuitive Bedienbarkeit der Applikation gewährleistet dabei, dass die verschiedensten Interessengruppen das Informationssystem sofort nutzen können. Mögliche Anwendungsfälle reichen dabei von besorgten Eltern, die sich ein Bild über die deutschlandweite Verteilung der Meningokokken machen möchten, bis zu Vertretern des öffentlichen Gesundheitsdienstes, die Entscheidungen anhand der dargestellten Cluster treffen.

Die Epidemiologen am NRZM haben während der Entwicklungszeit die Anwendung bereits eingesetzt. Mit deren Hilfe konnten sie beispielsweise die Vermutung visuell stützen, dass ein großer Teil der Serogruppe-B-Meningokokkenfälle im Westen Nordrhein-Westfalens bisher durch den Feintyp *B:P1.7-2,4:F1-5* ausgelöst worden sind, wohingegen dieser Feintyp im Osten des Bundeslandes relativ selten auftritt.

Identifizierung von Clustern

Während dieser Diplomarbeit ist es gelungen, die Software SaTScan dergestalt in ein Gesamtsystem zu integrieren, dass automatisierte spatio-temporale Clusteridentifizierungen erstmals möglich sind.

Der Server ist in der Lage, selbstständig Berechnungen zu planen und auszuführen. Ermittelte Ergebnisse lassen sich sofort über die Webanwendung visualisieren.

Die auf diese Weise identifizierten Cluster sind identisch mit denen, die das NRZM bisher durch eine halbautomatische SaTScan-Ausführung registrieren konnte. Der Beweis, ob EpiDeGIS als Frühwarnsystem richtige Prognosen trifft, lässt sich jedoch erst anhand zukünftiger Daten erbringen.

Flexibler Anwendungsentwurf

Es ist in dieser Arbeit gelungen, ein Softwaresystem zu schaffen, das flexibel ist. Die vorgestellten Konzepte und deren Implementierung haben viele Stellen aufgezeigt, an denen im System leicht Änderungen möglich sind. Das schließt auch die Übertragbarkeit auf andere Länder bzw. Krankheitserreger mit ein.

Erfolgreiche Kooperation

Diese Arbeit ist auch im Hinblick auf eine interdisziplinäre Zusammenarbeit zweier Institute der Universität Würzburg ein Erfolg. Durch die Kooperation und den Wissensaustausch zwischen dem Institut für Informatik und dem Institut für Hygiene ist ein neuartiges Projekt entstanden. Positive Resonanz aller Beteiligten und nationaler wie internationaler Behörden (RKI bzw. ECDC) stützen diese Aussage.

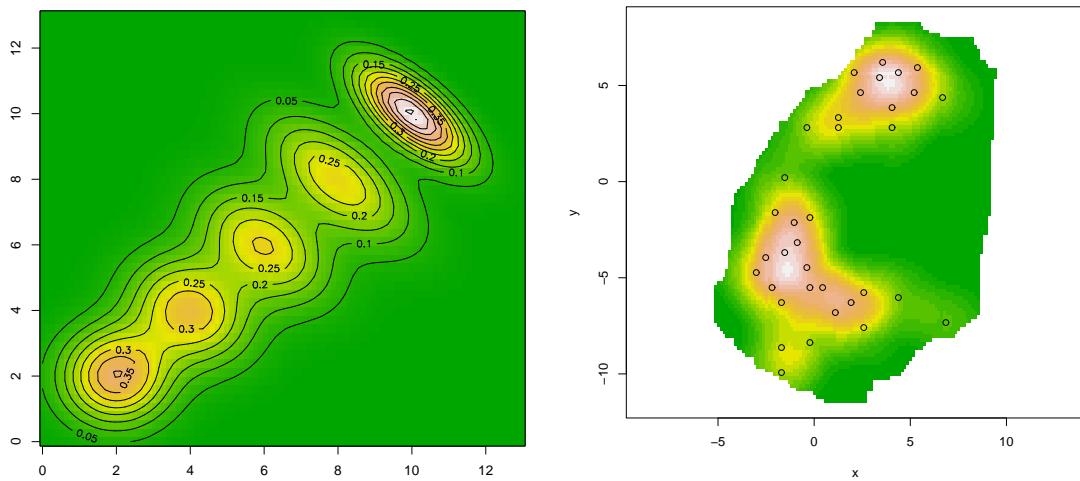


Abbildung 7.1.: Beispiele für mit den R-Paketen GenKern (links) und splancs (rechts) erzeugte “kernel-density”-Grafen

7.2. Erweiterungsmöglichkeiten

Während der Arbeit an diesem System haben sich bereits einige Erweiterungsmöglichkeiten aufgetan. Diese konnten im Rahmen dieser Arbeit leider nicht mehr implementiert werden, bieten allerdings interessante Ansätze für einen Ausbau der Funktionalität.

7.2.1. Integration des Statistikpaketes R

Auf Seite 42 wird das Statistikpaket R kurz vorgestellt. Zudem wird die Integrationsmöglichkeit der Software in die PostgreSQL Datenbank angesprochen. Während dieser Diplomarbeit wurden bereits Versuche zur Integration unternommen.

Die Fusion der Datenbank mit dem Statistikpaket eröffnet bei Visualisierung epidemiologischer Daten vielfältige Möglichkeiten. Diese könnten dem Anwender weitere Hilfestellung zur Bewertung einer Erkrankungssituation geben.

Anhand von Grafen ließe sich beispielsweise die Altersverteilung von Erkrankungsfällen aufzeigen. Diagramme, die z.B. die Serogruppenverteilung anzeigen, könnten direkt in die Webanwendung integriert werden.

Sehr interessant sind auch die Möglichkeiten, die sich durch die R-Pakete GenKern und splancs¹ bieten. Sie erlauben es, die im Abschnitt 6.4 beschriebenen “flächige” Darstellung von Punktinformationen zu erzeugen. Splancs bietet dabei ein sehr reichhaltiges Funktionsangebot zur Analyse von räumlichen und raum-zeitlichen Punktmustern. Das Paket kann u. a. eine zeitliche Folge von “kernel density”-Grafen aneinander gereiht dargestellten.

Die Abbildung 7.1 zeigt zwei durch diese Pakete generierte Grafen. Sie sind mittels der grafischen Oberfläche von R erstellt worden. In der EpiDeGIS-Datenbank ist über eine SQL-Funktion bereits die Erzeugung einer einfachen Grafik gelungen.

¹Spatial and Space-Time Point Pattern Analysis

7. Zusammenfassung und Ausblick

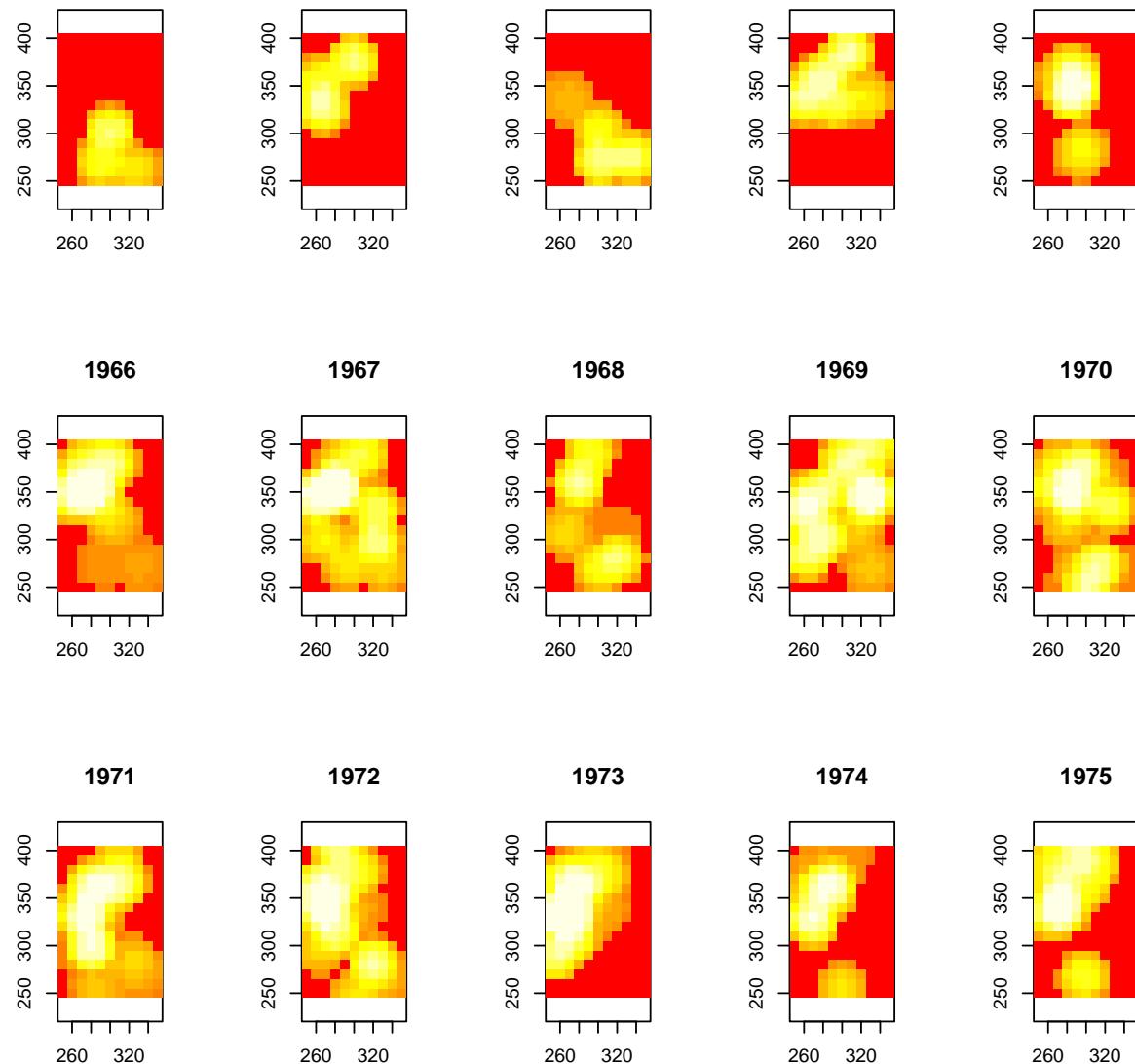


Abbildung 7.2.: Mit dem splancs-Paket erzeugte zeitliche Folge von “kernel-density”-Grafen

7.2.2. Animationen über die Dimension Zeit

Die Idee, die Dimension Zeit in dieses epidemiologische System zu integrieren, ist ein weiterer Ansatz für eine Erweiterung. Anhand von Animationen könnte es die Weboberfläche dem Anwender erleichtern, eine mögliche zeitliche Dynamik des Auftretens von Erkrankungsfällen zu erfassen.

OpenLaszlo bietet durch Flash bereits vielfältige Möglichkeiten Animationen abzuspielen. Was die serverseitige Anwendung betrifft, müsste nur noch eine Schnittstelle geschaffen werden, die es erlaubt, bestimmte definierte Karten direkt, d.h. unabhängig von der vom Benutzer gerade betrachteten Karte, zu erzeugen.

Der Vorgang könnte folgendermaßen ablaufen:

1. Der Anwender möchte eine Animation aller Erkrankungsfälle der Serogruppe über einen bestimmten Zeitraum mit monatlicher Auflösung betrachten.
2. Er öffnet eine Maske, die es ihm erlaubt, die genauen Einstellungen zu tätigen.
3. Die RIA kommuniziert mit dem Server und lädt z.B. für jeden Monat von Januar 2005 bis Dezember 2005 eine Karte mit den gewünschten Vorgaben. Hierbei sollte eine kleinere Auflösung gewählt werden, um Ladezeiten zu minimieren.
4. Sobald alle Karten geladen sind, wird dem Anwender die Animation in einem Fenster angezeigt. Dieses enthält Möglichkeiten, den Ablauf zu pausieren oder an bestimmte Zeitpunkte zu springen.
5. Die objektorientierten Fähigkeiten von OpenLaszlo würden es erlauben, auch mehrere Animationen parallel zu betrachten.
6. Ein möglicher Export der Animation als Flash-Datei rundet diese Komponente ab.

7.2.3. Ausbau der Benutzeroberfläche

Gerade durch die vielfältigen Möglichkeiten, die OpenLaszlo bietet, sind weitere Ausbaumöglichkeiten der Benutzerschnittstelle denkbar:

Speedtyping

Speedtyping bezeichnet eine Technik, durch die das schnelle Suchen in großen Datensätzen möglich ist. Die zu durchsuchende Datenmenge grenzt sich dabei, während der Suchbegriff eingetippt wird, anhand der bisher vom Anwender eingegebenen Buchstaben ein. Das Speedtyping könnte beispielsweise für die Suche nach einem bestimmten Feintyp des Meningokokkenerregers eingesetzt werden.

Bis dato enthält die EpiDeGIS-Datenbank 404 Feintypen. Aus diesen einen bestimmten Typ über eine Dropdown-Liste auswählen zu müssen, ist mehr als müßig. Ein Suchfeld erleichtert bisher das Auffinden eines Typs, dessen genauer Bezeichner bekannt ist.

Diese Suche kann durch Speedtyping verbessert werden. OpenLaszlo bietet dabei reichhaltige Möglichkeiten, Datenmengen zu verwalten, zu verändern und darzustellen [83].

7. Zusammenfassung und Ausblick

Automatisierte Alarmmeldungen

Zur Erweiterung des Frühwarnsystems ließe sich die Programmoberfläche noch um ein halbautomatisches Alarmsystem erweitern. Dieses könnte so funktionieren, dass berechtigte Benutzer, wie beispielsweise die Epidemiologen des NRZM, einen vom System gefundenen Cluster an eine in der Datenbank hinterlegte, offizielle Stelle weiterleiten.

Das System generiert auf den entsprechenden Befehl hin einen Bericht mit allen wichtigen Daten zu dem Cluster und versendet diesen per Email.

Danksagung

Danken möchte ich Herrn Prof. Dr. Jürgen Albert und Dipl.-Inf. Jörg Rothgänger sowie allen Betreuern für eine sehr gute Zusammenarbeit und viele hilfreiche Anregungen.

Ich danke Herrn Prof. Dr. Dag Harmsen insbesondere für die Möglichkeit, meine Arbeit schon vor deren Abschluss auf dem GIS-Tag Unterfranken 2005 öffentlich vorstellen zu dürfen.

Herrn Prof. Dr. med. Ulrich Vogel danke ich für aufschlussreiche fachliche Einblicke in die Epidemiologie.

Herrn Dr. med. Johannes Elias gebührt Dank dafür, dass er seine Erfahrungen mit der Software SaTScan mit mir geteilt hat.

Ich danke Herrn Dipl.-Inform. (FH) Fritz Kleemann für seine bereitwillige Hilfe bei auftauchenden Schwierigkeiten im Bereich der Serveradministration.

Meiner Familie und meinen Freunden möchte ich für ihr Vertrauen und ihre Unterstützung danken.

Schließlich und besonders danke ich meiner Verlobten Lena für ihre Liebe.

A. Softwaresystem EpiDeGIS

A.1. Installationsanleitung unter Linux

Der folgende Abschnitt stellt eine Installationsanleitung für alle Komponenten dar, die für den Einsatz von EpiDeGIS benötigt werden. Das sind alle externen Bibliotheken und Anwendungen sowie die einzelnen Softwarekomponenten des Systems.

Die Anleitung beschränkt sich dabei auf das letztlich verwendete Betriebssystem Linux. In der Anfangsphase der Entwicklung wurde allerdings ein ähnlicher Systemaufbau auch unter Windows durchgeführt. Die wichtigen Komponenten MapServer, PostgreSQL, SaTScan und der Tomcat Servlet Container laufen auch unter dem System von Microsoft. Alle Java-basierten Werkzeuge sind ohnehin plattformunabhängig.

A.1.1. Grundlagen

Manuelle Installation mit configure: Für die meisten externen Bibliotheken gibt es kein Installationspaket für Linux-Distributionen. Aus diesem Grund soll hier kurz auf den manuellen Kompilationsvorgang eingegangen werden. Dieser ist dank mitgelieferter Konfigurationsskripten mit wenig Aufwand verbunden. Der grundlegende Ablauf zur Installation eines Softwarepaketes aus dessen Quellen folgt folgendem Muster.

1. Der Programmquellcode wird von der jeweiligen Website herunter geladen.
2. Die Dateien liegen meistens gepackt im Format `tar.gz` vor und müssen dann mit dem Befehl `tar -zxvf paket-xy-1.0.tar.gz` in ein Verzeichnis entpackt werden.
3. Nach einem Wechsel in dieses Verzeichnis muss der Befehl `./configure` ausgeführt werden. Das Konfigurationsskript untersucht das System daraufhin, ob es alle Voraussetzungen erfüllt und gibt gegebenenfalls Hinweise auf unerfüllte Abhängigkeiten.
Kommandozeilenargumente der Form `--with-xyz=/usr/local` weisen dabei auf den Pfad, unter welchem eine abhängige Bibliothek (im Beispiel “xyz”) zu finden ist.
4. Ein fehlerfreier Ablauf des Konfigurationsskriptes produziert eine Datei mit dem Namen `Makefile`. Mit dem Befehl `make` wird die Software kompiliert.
5. Abschließend kann das Paket mit `make install` auf dem System installiert werden. Hierzu sind idR. Administrationsrechte erforderlich.

A. Softwaresystem EpiDeGIS

A.1.2. Installation der benötigten Softwarekomponenten

Die folgende Reihenfolge muss bei der Installation der einzelnen Komponenten eingehalten werden.

GEOS geometrische Bibliothek [73]

Der Installationsvorgang dieser Bibliothek entspricht dem unter A.1.1 angegebenen Ablauf. Nach der Installation befinden sich die Dateien `libgeos.a`, `libgeos.la`, `libgeos.so`, `libgeos.so.2` und `libgeos.so.2.1.4` im Verzeichnis `/usr/local/lib`.

PROJ.4 Kartenprojektions-Bibliothek [56]

Auch die Installation von PROJ.4 läuft nach dem Grundschema aus A.1.1 ab. Als Ergebnis eines erfolgreichen Installationsvorgangs befinden sich die Dateien `libproj.a`, `libproj.la`, `libproj.so`, `libproj.so.0` und `libproj.so.0.5.0` unter dem Pfad `/usr/local/lib`.

PostgreSQL (Version 8.x) [31]

PostgreSQL lässt sich am einfachsten mittels eines von der entsprechenden Linux-Distribution angebotenen Installationspaketes installieren. Wichtig ist in diesem Zusammenhang, dass die Quellen der entsprechenden PostgreSQL-Version für die folgende Kompilation von PostGIS trotzdem heruntergeladen, entpackt und konfiguriert (`./configure`) werden müssen. Eine Installation aus den Quellen darf jedoch **nicht** durchgeführt werden, wenn bereits das Distributionspaket installiert ist.

PostGIS für PostgreSQL [78]

Wie im vorherigen Abschnitt erwähnt, benötigt PostGIS für eine erfolgreiche Kompilation die konfigurierten Quellen von PostgreSQL. Hierbei müssen die Quellen von PostGIS innerhalb des PostgreSQL-Quellcodes in den Ordner `contrib` entpackt werden.

Da das `./configure`-Skript von PostGIS nicht gut funktioniert, ist es am einfachsten die Abhängigkeiten, mit denen PostGIS kompiliert werden soll, in der `Makefile` anzugeben. Listing A.1 zeigt die beiden Einträge in der Datei `Makefile` die festlegen, dass PostGIS inklusive einer Anbindung an GEOS und PROJ.4 kompiliert werden soll.

Listing A.1: Wichtige Makefile-Einträge für die PostGIS-Kompilation

```
USE_PROJ ?= 1
2 PROJ_DIR ?= /usr/local
4 USE_GEOS ?= 1
GEOS_DIR ?= /usr
```

Der für GEOS angegebene Pfad lautet nur `/usr`, weil sich das Programm `geos-config`, welches alle Informationen zur installierten GEOS-Version liefert, im Verzeichnis `/usr/bin/geos-config` befindet.

Der Befehl `make` führt bei diesen Einstellungen zu einer erfolgreichen Kompilation der PostGIS-Erweiterung für PostgreSQL. Mittels `make install` wird diese für die Datenbank installiert:

- Die Bibliotheksdateien werden unter dem Verzeichnis abgelegt, dass der Befehl `pg_config --pkglibdir` angezeigt.
- Programmdateien wie Export- und Importwerkzeuge liegen in dem durch den Befehl `pg_config --bindir` angezeigtem Verzeichnis.
- Alle weiteren Dateien sind unter `[prefix]/share/contrib`, `[prefix]/man` und `[prefix]/share/doc` abgelegt. Prefix ist die Ausgabe des Befehls `pg_config --configure`.

Um eine Datenbank mit den erforderlichen Tabellen und Funktionen zur Speicherung und Verwendung von Geodaten auszustatten, müssen unter dem `postgres`-Benutzer die Befehle in Listing A.2 ausgeführt werden. `<Datenbankname>` ist dabei die zu erweiternde Datenbank, die zu importierende Datei `lwpostgis.sql` befindet sich im Verzeichnis `[prefix]/share/contrib`.

Listing A.2: Befehl zur Erweiterung einer Datenbank mit PostGIS-Funktionen und Tabellen

```
1 createlang plpgsql <Datenbankname>
  psql -f lwpostgis.sql -d <Datenbankname>
```

UMN MapServer (Version 4.6x) [68]

Der UMN MapServer muss als letztes kompiliert werden, da er alle vorher installierten Bibliotheken nutzt. Der Befehl in Listing A.3 führt zur Konfiguration des Mapservers, wobei bei `--with-xyz` jeweils die Pfade eingesetzt werden müssen, unter denen sich die entsprechenden Abhängigkeiten befinden.

Listing A.3: Befehl zur Konfiguration der MapServer-Quellen

```
./configure --with-proj=/usr/local --with-geos=/usr/bin/
  geos-config --with-postgis=/usr/local/pgsql/bin/
    pg_config
```

Nach der erfolgreichen Kompilation des Servers befindet sich ein Programm mit dem Namen `mapserv` im Quellverzeichnis. Wenn der Befehl `mapserv -v` die unter Listing A.4 angezeigte Ausgabe produziert, hat alles geklappt.

A. Softwaresystem EpiDeGIS

Listing A.4: Ausgabe zur Verifikation der MapServer-Funktionalität

```
1 epidegis@localhost:/src/mapserver/mapserver-4.6.1 # ./mapserv -v
MapServer version 4.6.1 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG
OUTPUT=WBMP OUTPUT=SVG SUPPORTS=PROJ SUPPORTS=FREETYPE
SUPPORTS=WMS_SERVER SUPPORTS=GEOS INPUT=EPPL7 INPUT=
POSTGIS INPUT=SHAPEFILE
3 epidegis@localhost:/src/mapserver/mapserver-4.6.1 #
```

Das Programm `mapserv` ist eine CGI-Anwendung. Um diese als Server nutzen zu können, muss sie z.B. durch den Apache-Webserver ausgeführt werden.

Für die “Installation” des MapServers sind folgende Schritte nötig:

1. Einrichtung eines Virtual-Hosts für den installierten Apache-Webserver.
2. Das Hauptverzeichnis des Virtual-Hosts muss dabei mindestens die Ordner `cgi-bin` und `maps` enthalten.
3. Das `mapserv` Programm wird unter `cgi-bin` abgelegt.
4. ”MapFiles” werden unter `maps` gespeichert.

Wenn die Installation erfolgreich war, ist der Mapserver über den URL

`http://localhost/cgi-bin/ms4`

erreichbar und meldet sich mit:

“No query information to decode. QUERY_STRING is set, but empty.”.

Tomcat Servlet Container (Version 5.x) [4]

Die Installation des Tomcat Servlet Containers kann entweder manuell, oder wieder mittels eines von der Linux-Distribution bereitgestellten Installationspaketes erfolgen. In jedem Fall sollte nach der Installation unter dem URL `http://localhost:8080` die Willkommensseite des Apache Tomcat zu sehen sein.

A.1.3. Installation und Konfiguration der EpiDeGIS-Webanwendung

Import der Datenbank

Eine komplette Kopie, der in dieser Diplomarbeit genutzten Datenbank befindet sich auf dem zughörigen Datenträger unter `/db/epidegis_db.sql`. Die Importanleitung steht in `import.txt`.

Kartendefinition für den Mapserver

Damit der MapServer in der von EpiDeGIS geforderten Konfiguration arbeitet, benötigt er die entsprechende Kartendefinitionsdatei (“MapFile”). Die aktuellste Version, der für die Implementierung am NRZM eingesetzten Kartendefinition liegt auf dem Datenträger unter /mapserver/nrzm.map. Die Installation erfolgt, in dem sie in das maps-Verzeichnis des Apache Virtual-Hosts kopiert wird (*siehe Abschnitt A.1.2 → UMN MapServer*). Bei der hier beschriebenen Konfiguration sollte der Aufruf des URL

```
http://localhost/cgi-bin/mapserv?map=../maps/nrzm.map&version=1.1.1&service=WMS&REQUEST=GetCapabilities
```

ein XML-Dokument ausgeben, welches die “Capabilities” des EpiDeGIS-WMS enthält.

Sollte der Mapserver unter einem anderen als dem obigen URL installiert sein, ist es erforderlich die folgende Zeile in der Datei nrzm.map entsprechend anzupassen:

```
"wms_onlineresource" "http://localhost/cgi-bin/mapserv?map=../map/nrzm.map&"
```

Serverseitige Webanwendung

Die serverseitige Webanwendung wird durch das WAR-Dateiformat im Tomcat Servlet Container installiert. Hierzu muss die durch die Kompilation des Unterprojekts *epidegis-web* (siehe Abschnitt C) entstehende Datei *epidegis-web.war* in das webapps-Verzeichnis des Tomcat kopiert werden. Eine bereits kompilierte Version der Webapplikation befindet sich auf der Projekt-CD zu dieser Diplomarbeit unter /webapp/*epidegis-web.war*.

Der Java Server entpackt das Archiv in das Verzeichnis *webapps/epidegis-web*. Die beiden zu konfigurierenden Dateien sind:

- *webapps/epidegis-web/WEB-INF/classes/epidegis-web.properties*
Hier muss der Schlüssel *wms.url* in den URL geändert werden, unter dem der MapServer erreichbar ist.
- *webapps/epidegis-web/WEB-INF/classes/epidegis-db.properties*
Hier müssen nur Änderungen gemacht werden, wenn eine andere Datenbankbindung als die JNDI-Datenquelle des Tomcats verwendet werden soll.
- *webapps/epidegis-web/META-INF/context.xml*
Definiert den Servlet-Context für *epidegis-web*. Hier ist die Datenbankbindung der serverseitigen Anwendung festgelegt. Hier müssen die Daten zu dem URL, dem Benutzernamen und dem Passwort zu der eingesetzten Datenbankverbindung angegeben werden.

Konfiguration im Tomcat Zur Nutzung der EpiDeGIS-Webanwendung müssen zwei Konfigurationen im Tomcat Servlet Container durchgeführt werden:

- Installation des PostgreSQL-JDBC-Treibers
Der PostgreSQL-Treiber muss als JAR-Archiv im Verzeichnis *common\lib* abgelegt sein. Der Treiber ist für die entsprechende Datenbankversion über die Website von PostgreSQL zu beziehen [31].

A. Softwaresystem EpiDeGIS

- Anlegen eines EpiDeGIS-Benutzers

In der Datei conf/tomcat-users.xml sind Benutzer definiert. Die EpiDeGIS-Webanwendung öffnet sich per Konfiguration nur für Benutzer, welche die Rolle (“role”) *epidegis_beta* einnehmen. In der XML-Datei ist deshalb ein Eintrag der Form `<user username="test" password="geheim" roles="epidegis_beta"/>` erforderlich.

A.2. Vorstellung des Kartenbetrachters

Der folgende Abschnitt stellt die Funktionalität der in diesem Projekt entwickelten Web-basierten GIS-Anwendung zur Überwachung von Meningokokken-Erkrankungsfällen in Deutschland vor.

A.2.1. Gesamtübersicht

Die Oberfläche des Kartenbetrachters bietet folgende Einstellungsmöglichkeiten (Abbildung A.1):

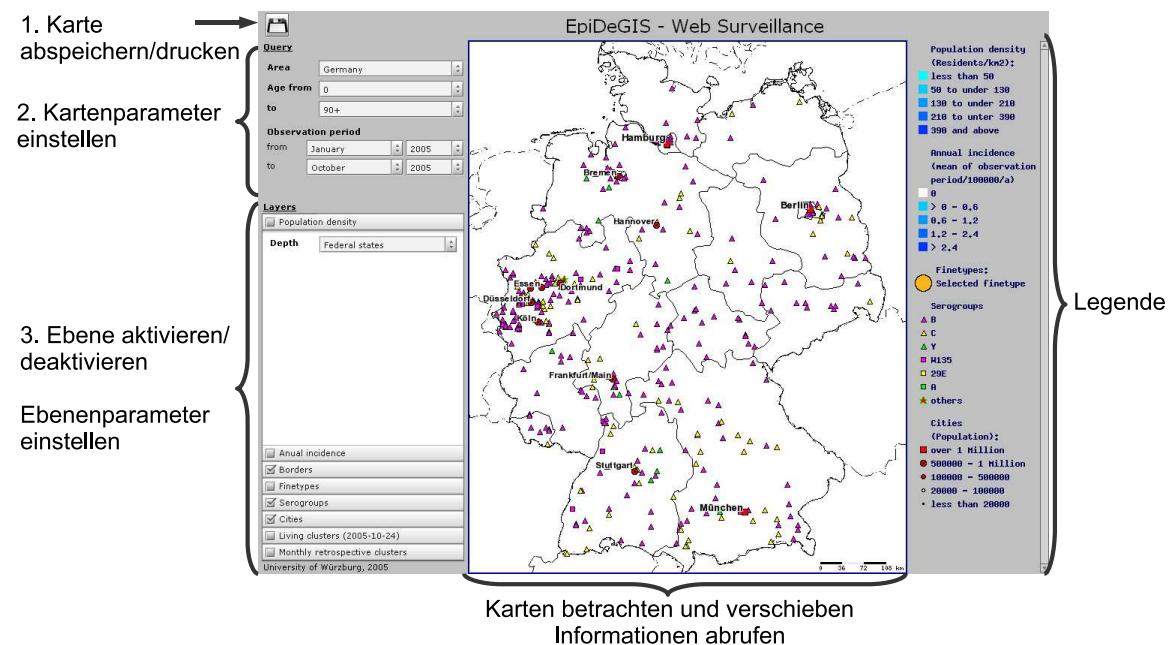


Abbildung A.1.: Screenshot der EpiDeGIS-RIA

1. Über diese Schaltfläche kann die Karte in einer druckfähigen Version aufgerufen werden.
2. Über diese Dropdown-Listen können Filter eingestellt werden, welche die angezeigten Karten verändern:
 - a) dargestelltes Gebiet (Bundesgebiet oder einzelne Bundesländer)
 - b) Altersgrenzen (von, bis), auf welche die dargestellten Erkrankungsfälle eingegrenzt werden

- c) Zeitraum zur Begrenzung der erzeugten epidemiologischer Informationen
3. In diesem Bereich ist für jede vorhandene Kartenebene ein Reiter zu sehen. In der Kopfzeile lässt sich die Ebene ein- bzw. ausblenden. Zudem wird anhand eines Ladebalkens angezeigt, ob sich eine Ebene gerade aktualisiert. Einstellungen zu den einzelnen Ebenen sind unten aufgelistet.
 4. Der Hauptbereich der Anwendung zeigt die aktuelle, anhand aller Filtereinstellungen generierte Karte. Durch klicken und ziehen der Maus lässt sich der Kartenausschnitt verschieben. Ein einfacher Mausklick ohne Bewegung zeigt Informationen zu einem gewählten Punkt auf der Karte an. Hierbei werden alle gerade aktiven Ebenen nach Informationen abgefragt.
 5. Die Legende am rechten Rand zeigt die Bedeutung, der in den Ebenen verwendeten Farben und Symbole.

A.2.2. Ebenen

Im Folgenden werden die verschiedenen Ebenen aufgelistet, deren spezifische Filtermöglichkeiten beschrieben und Beispielabbildungen gezeigt:

Population density

Stellt die Bevölkerungsdichte dar. Diese lässt sich entweder auf Bundesland- oder Landkreisebene visualisieren. Zwei Beispiele zeigt Abbildung A.2.

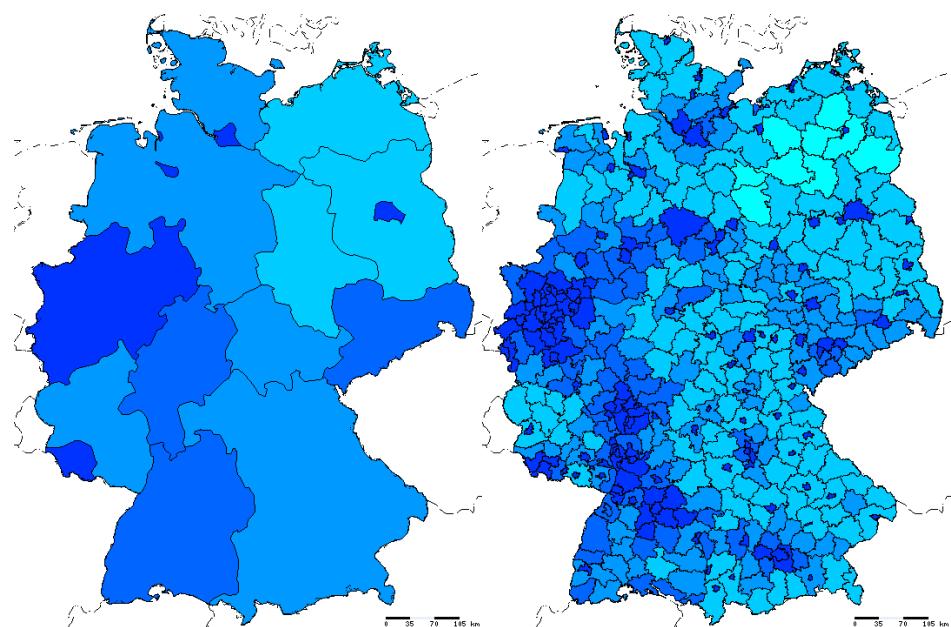


Abbildung A.2.: Bevölkerungsdichte auf Bundesland- und Landkreisebene mit Grenzlinien

A. Softwaresystem EpiDeGIS

Anual incidence

Zeigt die Jahresinzidenz der Erkrankungsfälle in Bundesländer oder Landkreisen an. Die Berechnung bezieht sich dabei auf alle Fälle des ausgewählten Feintyps die innerhalb des angegebenen Zeitraums aufgetreten sind. Falls die Zeitspanne unter einem Jahr ist, wird der Wert hochgerechnet. Bei über einem Jahr wird der Mittelwert gebildet. Ein Beispiel zeigt Abbildung A.3.

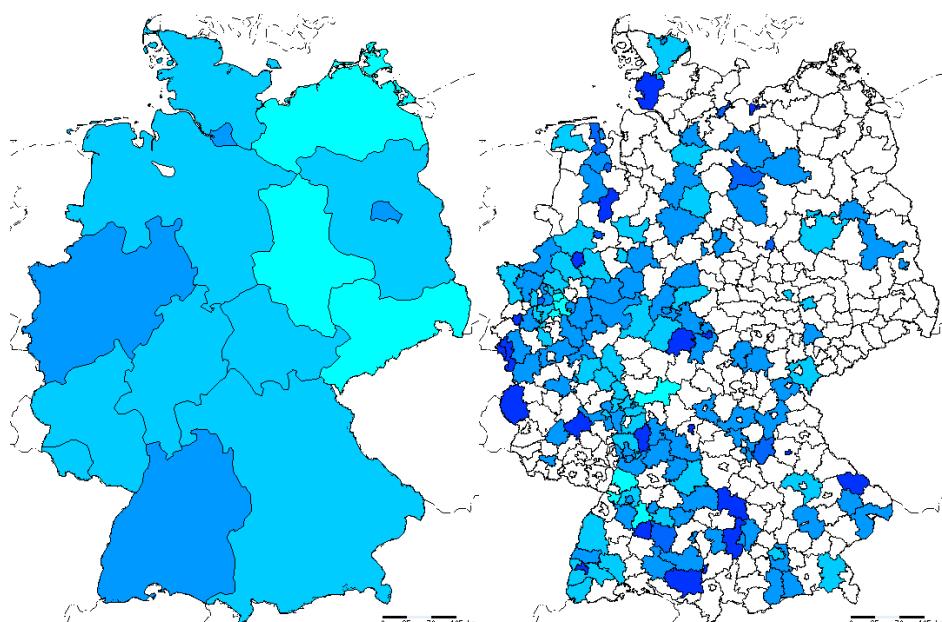


Abbildung A.3.: Jahresinzidenz 2002 (Januar-Dezember) für alle Meningokokkenfälle für Bundesländer und Landkreise

Finetypes

Stellt die Verteilung eines Meningokokken-Feintyps auf der Landkarte dar. Über ein Suchfeld lässt sich bei den Ebeneneinstellungen ein spezifischer Feintyp aus den bisher 404 im System erfassten anzeigen. Zwei Beispiele zeigen Abbildung A.4.

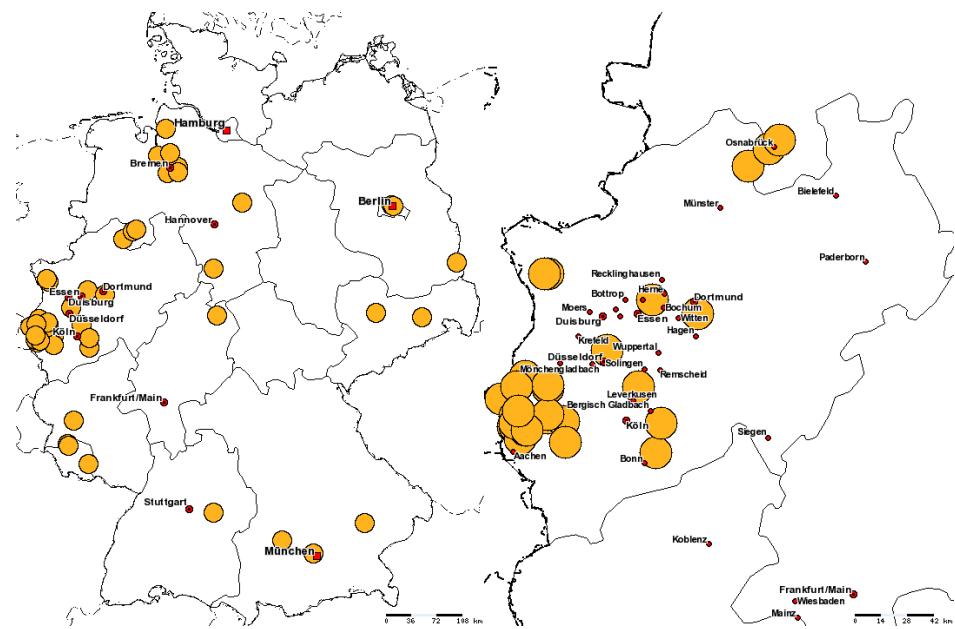


Abbildung A.4.: Verteilung des Feintyps B:P1.7-2,4:F1-5 in Deutschland und in Nordrhein-Westfalen (Januar bis Oktober 2005)

Serogroups

Visualisiert Erkrankungsfälle zusammengefasst nach Serogruppen. Die Menge der angezeigten Fälle wird durch die Altersgrenzen und den Zeitraum gefiltert. Es ist möglich die Fälle verschiedener Serogruppen gleichzeitig auf der Karte anzuzeigen. Unterschiedliche Symbole kennzeichnen die Gruppen B, C, Y, W135, 29E, A und Z eindeutig. Zwei Beispiele zeigt Abbildung A.5.

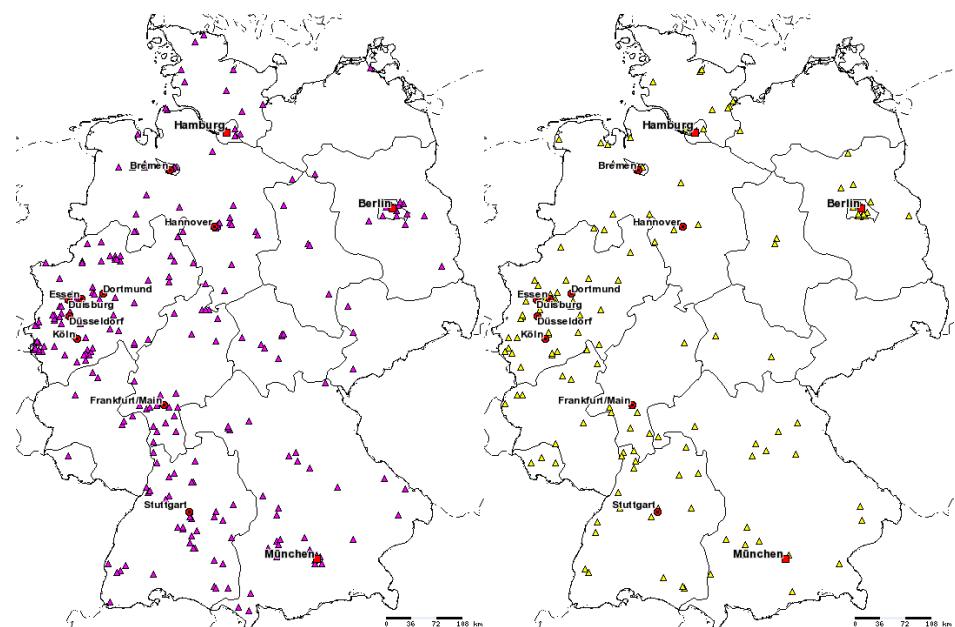


Abbildung A.5.: Serogruppe B (links) und C Fälle in Deutschland im Jahr 2002

A. Softwaresystem EpiDeGIS

Living clusters

Diese Ebene erlaubt keine weiteren Einstellungsmöglichkeiten. Sie zeigt immer die durch die zuletzt durchgeführte prospektiven Analyse identifizierten Cluster. Das Datum, an dem die Berechnung aufgeführt wurde, wird in der Überschrift der Ebene angezeigt. Zwei Beispiele zeigen Abbildung A.6.

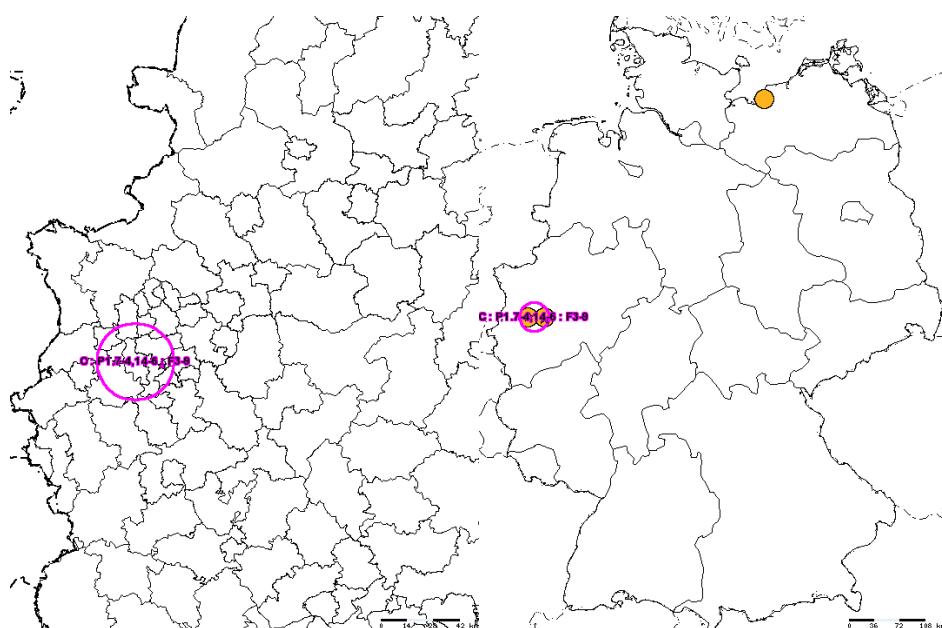


Abbildung A.6.: Ein bei einer prospektiven Analyse am 24. Oktober 2005 entdeckter Cluster, Feintyp $C:P1.7-4,14-6:F3-9$, Dauer bis dato vom 27. September - 24. Oktober 2005; rechts die deutschlandweite Verteilung dieser Feintyps von August-Oktober 2005

Monthly retrospective clusters

Einen historisch rückblickende Darstellung der Clusterlandschaft wird mittels dieser Ebene visualisiert. Über ein Auswahlfeld lässt sich wählen, welche der monatlich durchgeführten Retrospektiven Analysen angezeigt werden soll. Ein Beispiel zeigt Abbildung A.7.

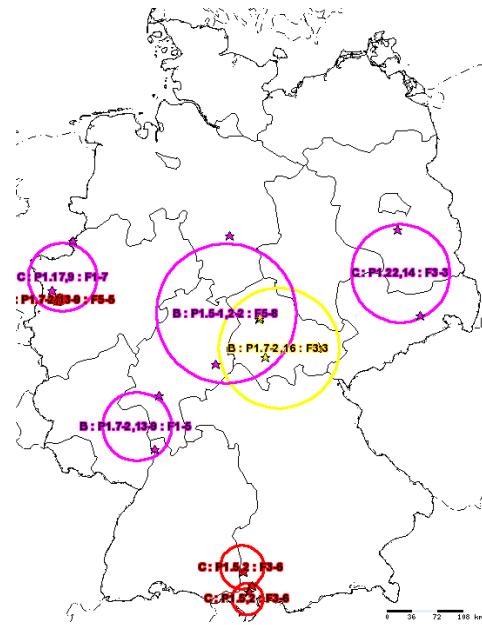


Abbildung A.7.: Durch eine retrospektive Analyse erkannte historische Cluster. Die Berechnung hat vom 1. September 2005 an rückblickend alle Fälle des letzten Jahres eingeslossen.

A.3. Informationen für Entwickler

Dieser Abschnitt ist speziell an Entwickler gerichtet, die sich einen Überblick über vorhandenen Strukturen und Programmquellen verschaffen möchten. Mit dem offiziellen Ende dieser Diplomarbeit befindet sich die zu diesem Projekt entwickelte Software in einem Beta-Stadium. Jede der in dieser Arbeit vorgestellten Komponenten leistet das geforderte in manchen Bereichen bedarf es jedoch noch Überarbeitungen.

Aufgrund dessen behandelt die hier vorliegende gedruckte Form auch nur Hinweise darauf, wo sich weitere Informationen zu diesem Projekt finden lassen.

A.3.1. Projektverwaltung mit Maven

Diese Arbeit verwendet das Werkzeug Maven [2] um den Entwicklungsvorgang einfacher verwalten zu können (siehe auch 3.8.2). Während dieser Arbeit hat Maven einen großen Versionssprung von 1.02 auf 2.0 gemacht. Das in dieser Arbeit verwendetet Projektmodell basiert bis dato noch auch Maven 1.02, zuletzt wurde Maven 1.1 Beta 2 eingesetzt. Diese Version befindet sich als Windows-Installer oder alternativ als `tar.bz2`-gepackte Datei im Ordner `/maven` des beigefügten Datenträgers. Das Verzeichnis enthält außerdem das Handbuch mit Installationsanleitung und einer Kurzbeschreibung. Zur einfachen Kompilation des EpiDeGIS-Quellcodes ist Maven sehr hilfreich.

A. Softwaresystem *EpiDeGIS*

A.3.2. HTML-Dokumentation des Projektes EpiDeGIS mit Quellcode und JavaDoc

Im Verzeichnis doc der Projekt-CD befindet sich eine HTML-Dokumentation zu den in Java implementierten Komponenten von EpiDeGIS. Es ist geplant, diese nach Abgabe dieses schriftlichen Teils noch weiter zu überarbeiten. Der Aufruf der Datei `/doc/index.html` in einem Internetbrowser öffnet die Einstiegsseite zur Dokumentation. Von hier aus lassen sich Informationen zu jedem einzelnen Unterprojekt aufrufen.

Die beiden wichtigsten Informationsquellen sind dabei die *JavaDoc*- und die *Source Xref*-Dokumentation. Diese finden sich nach dem Öffnen eines Unterprojektes (z.B. *EpiDeGIS Database Access*) in dem Menü *Project Reports*. Besonders die *JavaDocs* werden noch erweitert.

Weiterhin bieten die Einstiegsseiten zu jedem Teilprojekt Informationen dazu, wie es kompiliert und angewendet werden kann bzw. welche Punkte noch zu überarbeiten sind.

B. Geografische Daten und Zensusdaten

Geografische Daten

Diese Diplomarbeit hat die freien Deutschlanddaten des Mapbender-Projekts [24] eingesetzt. Sie sind frei über dessen Homepage erhältlich. Der Datensatz enthält unter anderem die Postleitzahlbereiche, Bundesländer und Städte für Deutschland.

Die Daten liegen im SHP-Dateiformat [10] vor und können mit dem Programm `shp2pgsql` aus dem PostGIS-Paket leicht in die Datenbank übertragen werden.

Um Landkreispolygone zu gewinnen, mussten die Postleitzahlen manuell zu Landkreisen zusammengefasst werden. Die Aufgabe wurde mit Hilfe einer über das NRZM zur Verfügung gestellten Zuordnungstabelle Postleitzahl \iff Landkreis und entsprechenden PostGIS-SQL-Abfragen, in der Anfangsphase dieser Arbeit gelöst.

Exaktere geografische Daten eines “Datenhändlers” könnten die Qualität der von EpiDeGIS erzeugten Karten verbessern.

Zensusdaten

Die zu Berechnung der Inzidenz und der Bevölkerungsdichte herangezogenen Zensusdaten stammen vom Statistisches Bundesamt Deutschland [36]. Das Amt stellt alle Einwohnerzahlen auf Landkreisebene zur freien Verfügung.

Auch für diese Daten gilt, dass eine Verbesserung der Datenqualität, wie beispielsweise Zensusdaten aufgeschlüsselt nach Altersgruppen, die Möglichkeiten des Systems EpiDeGIS erweitern könnten. Diese Daten sind, genau wie professionelle Geodaten, kostenpflichtig zu erwerben.

C. Inhalt des Datenträgers zu dieser Diplomarbeit

Folgende Inhalte sind auf dem Datenträger zu dieser Diplomarbeit abgelegt:

/ausarbeitung

- ./literatur hier sind ein Großteil der in dieser Arbeit referenzierten Quellen in Form von PDF-Dateien bzw. als HTML-Dokumente abgelegt. Sortierung nach Autor und Jahr bzw. u. U. Titel der Arbeit.
- ./diplom.pdf die vorliegende Arbeit im PDF-Format
- ./biblio.bib Literaturangaben im BibTeX-Format

/db

- ./epidegis_db.sql Backup der Datenbank
- ./import.txt Hinweise zum Import der Datenbank
- ./struktur.pdf Informationen zu der Datenbankstruktur

/doc

HTML-Dokumentation des Projektes EpiDeGIS (*index.html*)

/gis-tag Präsentation, mit der diese Arbeit am 8. November auf dem GIS-Tag Unterfranken vorgestellt wurde.

/mapserver

- ./nrzm.map “MapFile”, die für das epidemiologischen GI-System des NRZM eingesetzt wird

/maven

Programmdateien und Handbuch für das Maven-Werkzeug

/project

EpiDeGIS Projektordner mit den Quellcodes zu allen Teilprojekten

- ./epidegis-db Datenbank-Teilprojekt: implementiert “Data Access Objects”, über die alle anderen Java-Module von EpiDeGIS auf die Datenbank zugreifen.
- ./epidegis-importer Import-Anwendung: importiert eine CSV-Datei in die Datenbank von EpiDeGIS .
- ./epidegis-satscan SaTScan-Modul: führt das Programm SaTScan anhand von Job-Definitionen aus der Datenbank aus und speichert gefundene Cluster.
- ./epidegis-transfer Transfer-Applikation: stellt eine SSH-Verbindung her, um Daten vom NRZM an den Datenserver zu übertragen.

C. Inhalt des Datenträgers zu dieser Diplomarbeit

./epidegis-web Webanwendung: implementiert die serverseitige Webapplikation, die RIA mit OpenLaszlo, den abstrakten Datentyp einer Karte und die Kommunikation mit dem MapServer.

./xdocs enthält Quellen für die HTML-Dokumentation

D. Datei-Listings

Listing D.1: Kommentiertes Ausgabeformat von SaTScan

```
SUMMARY OF DATA

Study period.....: 2001/11/1 - 2001/11/24
Number of locations....: 191
Total population.....: 203
Total cases.....: 62

MOST LIKELY CLUSTER

1. Location IDs included.: 19, 18, 14, 26
   Coordinates / radius...: (5026,4301) / 4.47
   Population.....: 4
   Number of cases.....: 4
   Expected cases.....: 1.22
   Observed / expected....: 3.274
   Log likelihood ratio...: 4.836516
   Monte Carlo rank.....: 674/1000
   P-value.....: 0.674

SECONDARY CLUSTERS

2. Location IDs included.: 170, 173, 172, 169
   Coordinates / radius...: (5176,4669) / 5.83
   Population.....: 4
   Number of cases.....: 4
   Expected cases.....: 1.22
   Observed / expected....: 3.274
   Log likelihood ratio...: 4.836516
   Monte Carlo rank.....: 674/1000
   P-value.....: 0.674

...
```

Listing D.2: Beispiel einer Konfigurationsdatei für den MapServer mit WMS-Unterstützung und einer Ebene

```
MAP
2 NAME EPIDEGIS_WEB_SURVEILLANCE
EXTENT 3280173.25 5238019 3921337.5 6103262.5
4 FONTSET "fonts/fonts.list"
```

D. Datei-Listings

```

SYMBOLSET "symbols/symbols35.sym"
6 WEB
    MINSCALE 1000
8 MAXSCALE 15000000
METADATA
10   "wms_title" "EpiDeGIS Web Surveillance WMS"
    "wms_onlineresource" "http://localhost/cgi-bin/ms4?
        map=../maps/simple.map&"
12   "wms_srs" "EPSG:4326 EPSG:31467"
    END
14 END
OUTPUTFORMAT
16   NAME png
    DRIVER "GD/PNG"
18   MIMETYPE "image/png"
    IMAGEMODE RGB
20   EXTENSION "png"
    END
22 IMAGETYPE gif
    SIZE 460 500
24 IMAGECOLOR 255 255 255
PROJECTION
26   "init=epsg:4326"
    END
28 LEGEND
    KEYSIZE 12 12
30   LABEL
        TYPE BITMAP
32     SIZE MEDIUM
34     COLOR 0 0 89
    END
36     IMAGECOLOR 192 192 192
    STATUS OFF
    END
38 LAYER
    NAME cities
40     STATUS ON
    CONNECTIONTYPE postgis
42     CONNECTION "user=mapserver dbname=epidegis_works host=
        localhost"
    DATA "the_geom from (
44         SELECT gid, the_geom,
        full_name AS name, population_n AS population
        FROM cities
46     ) AS foo USING SRID=4326 USING UNIQUE gid"
48     TYPE POINT
    LABELITEM "name"
    METADATA
50

```

```

      "WMS_SRS"      "epsg:4326"
52    "WMS_TITLE"     "Cities"
      END
54  PROJECTION
      "init=epsg:4326"
56  END
      LABELMAXSCALE 15000000
58  SYMBOLSCALE 1000
      CLASS
60    EXPRESSION ([population] > 1000000)
      NAME 'over 1 Million'
62    MINSCALE 1000
      MAXSCALE 15000000
64    STYLE
      SYMBOL 'square'
66      MINSIZE 8
      MAXSIZE 16
68      COLOR 255 0 0
      OUTLINECOLOR 0 0 0
70    END
      LABEL
72        TYPE TRUETYPE
      FONT "arial-bold"
74        MINSIZE 10
      MAXSIZE 15
76        POSITION AUTO
      MINDISTANCE 2
78        OUTLINECOLOR 230 230 230
      COLOR 0 0 0
80        PARTIALS FALSE
      END
82    END
      END
84 END

```

Listing D.3: MapServer-Datenquelle für die Inzidenz-Ebene

```

DATA "the_geom from (
2   SELECT awt.area_id AS oid, awt.the_geom,
      count(*)/(awt.population::real/100000)*
4     ( 12/(%toYear%*12+%toMonth%)
       (%fromYear%*12+%fromMonth%)+1))
6     AS incidence
FROM contains_area_case AS cac, areas_with_types AS awt,
8     cases, types_with_attributes AS twa
WHERE cac.area_id=awt.area_id
10    AND cac.case_id=cases.case_id
      AND cases.case_type_id = twa.case_type_id
12    AND twa.attname LIKE 'Serogroup'

```

D. Datei-Listings

```

14      AND twa.attvalue IN (%SEROGROUPS%)
15      AND CAST(to_char(reportdate, 'YYYY') AS int) * 12
16          + CAST(to_char(reportdate, 'MM') AS int)
17      BETWEEN %fromYear%*12+%fromMonth%
18          AND %toYear%*12+%toMonth%
19      AND awt.tier=%INCITIER%
20 GROUP BY awt.area_id, awt.the_geom, population
) AS foo USING SRID=4326 USING UNIQUE oid"

```

Listing D.4: Eine abgespeckte Karte in XML-Form

```

<map>
  <size width="640" height="480"/>
  <selectparameter name="areaId">
    <title>Area</title>
    <value selected="true" name="Germany">9470</value>
    <value selected="false" name="Bayern">8713</value>
    <value selected="false" name="Nordrhein-Westfalen">
      8718
    </value>
  </selectparameter>
  <layer title="Population density" active="false"
    name="popdensity" description="">
    <legend/>
    <selectparameter name="POPDENTIER">
      <title>Depth</title>
      <value selected="true"
        name="Federal states">3</value>
      <value selected="false"
        name="Counties">4</value>
    </selectparameter>
  </layer>
  <layer title="Cities" active="true"
    name="cities_default" description="">
    <legend/>
  </layer>
</map>

```

E. Abkürzungen

AGI	Arbeitsgemeinschaft Influenza
ANSI	American National Standards Institute
BMGS	Bundesministerium für Gesundheit und Soziale Sicherung
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
CSV	Character Separated Values bzw. Comma Separated Values
DAO	Data Access Objects
DBMS	Datenbankmanagementsystem
ECDC	European Centre for Disease Prevention and Control
EpiDeGIS	Arbeitstitel für das in dieser Diplomarbeit erstellte Softwaresystem
ESRI	Environmental Systems Research Institute
GIF	Graphics Interchange Format
GIS	Geografisches Informationssystem
GUI	Graphical User Interface, Grafische Benutzeroberfläche
HTML	Hypertext Markup Language
IDC	International Data Corporation
IfSG	Infektionsschutzgesetz oder Gesetz zur Verhütung und Bekämpfung von Infektionskrankheiten beim Menschen
ISO	International Organization for Standardization
JSP	JavaServer Pages
LLR	Log Likelihood Ratio
MVC	Model View Controller (Architekturmuster)
NASA	National Aeronautics and Space Administration
NRZM	Nationales Referenzzentrum für Meningokokken
OGC	Open Geospatial Consortium
ORDBMS	objektrelationales Datenbankmanagementsystem
OSS	Open Source Software

E. Abkürzungen

RIA(s)	Rich Internet Application(s)
RKI	Robert Koch-Institut
SFS	Simple Features Specification
SMI	Swedish Institute for Infectious Disease Control
SQL	Structured Query Language
SVG	Scalable Vector Graphics
SWF	Small Web Format / Shockwave Flash
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WMS	Web Map Service
WWW	World Wide Web
XML	Extensible Markup Language
XUL	XML User Interface Language

Selbstständigkeitserklärung

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Würzburg, den 23. Dezember 2005 _____ (Markus Reinhardt)

Abbildungsverzeichnis

1.1.	Darstellung der Weltkugel als Informationsquelle in “Google Earth”	1
1.2.	Ausbreitung der Geflügelpest/Vogelgrippe (aviäre Influenza Subtyp A/H5N1), Stand: Oktober 2005 [6]	2
2.1.	Erfassung von Infektionserkrankungen in Deutschland.	6
2.2.	Bisheriger Informationsfluss bei der Typisierung von eingesandten Proben im NRZM	10
2.3.	Vergleich eines Polygons in Vektor- und Rasterdarstellung	12
2.5.	Mercator-Projektion [60]	13
2.4.	Beispiele für das erzeugte Bild der Erdoberfläche bei einer Zylinder-, Azimutal- oder Kegelprojektion [60]	13
4.1.	Neuer Informationsfluss, ermöglicht durch den Einsatz eines Webbasierten Systems (am Beispiel des NRZM, vgl. Abbildung 2.2 auf Seite 10)	28
4.2.	Benutzeroberfläche von SurvStat. Das System erzeugt anhand der in der Maske eingestellten Parameter unter anderem Kreisdiagramme. (Bildschirmfoto, Stand: 05. Dezember 2005)	29
4.3.	Aktivität der Influenza in der 6. Kalenderwoche 2005 (AGI-Website, Stand: 05. Dezember 2005)	30
4.4.	Inzidenz der Meningokokken-Erkrankungsfälle in Schweden von Januar bis Oktober 2005 (Bildschirmfoto, Stand: 06. Dezember 2005)	30
4.5.	Eine Karte wird im GIS aus einzelnen Ebenen aufgebaut	33
4.6.	“Model-2”-Architektur: Modell-View-Controller für Webanwendungen	36
4.7.	Übersicht der Architektur von OpenLaszlo. Laszlo-Anwendungen lassen sich eigenständig (“Solo Deployment”) oder in Verbindung mit einem Java Servlet Container einsetzen. (Quelle: Laszlo Systems [21])	37
4.8.	Mapserver in seiner Funktion als WMS	41
5.1.	Verwendung des MVC-Architekturmusters in EpiDeGIS	44
5.2.	Hierarchische Speicherung von Gebietsinformationen	45
5.3.	Definition von beliebig vielen Attributwerten für einen Erregertyp	46
5.4.	Komposition von Karten, Ebenen und Parametern [29]	48
5.5.	Konkrete Umsetzung des Parameterkonzeptes in der Laszlo-Anwendung - oben ein Kartenparameter (Area), unten ein Ebenenparameter (Finetype).	49
5.6.	Paketdiagramm des EpiDeGIS-Systems [29]	50
5.7.	Screenshot einer Desktop-GIS-Anwendung die Ebenen aus EpiDeGIS einbindet	52

Abbildungsverzeichnis

5.8. Graf der Antwortzeiten mit und ohne Cache	53
6.1. Komplettübersicht des Systems EpiDeGIS	56
6.2. Sequenzdiagramm der Karteninitialisierung [29]	57
6.3. Klassendiagramm des Exporter-Konzeptes [29]	58
6.4. "ActionForm Bean"-Klasse für die <i>MoveMapAction</i> [29]	59
6.5. Ausgabe des Programms in Listing 6.8	67
6.6. Datenbankstruktur zum Planen von SaTScan-Ausführungen und Speichern der Ergebnisse	71
7.1. Beispiele für mit den R-Paketen GenKern (links) und splancs (rechts) erzeugte "kernel-density"-Grafen	75
7.2. Mit dem splancs-Paket erzeugte zeitliche Folge von "kernel-density"-Grafen	76
A.1. Screenshot der EpiDeGIS-RIA	86
A.2. Bevölkerungsdichte auf Bundesland- und Landkreisebene mit Grenzlinien	87
A.3. Jahresinzidenz 2002 (Januar-Dezember) für alle Meningokokkenfälle für Bundesländer und Landkreise	88
A.4. Verteilung des Feintyps B:P1.7-2,4:F1-5 in Deutschland und in Nordrhein-Westfalen (Januar bis Oktober 2005)	89
A.5. Serogruppe B (links) und C Fälle in Deutschland im Jahr 2002	89
A.6. Ein bei einer prospektiven Analyse am 24. Oktober 2005 entdeckter Cluster, Feintyp <i>C:P1.7-4,14-6:F3-9</i> , Dauer bis dato vom 27. September - 24. Oktober 2005; rechts die deutschlandweite Verteilung dieser Feintyps von August-Oktober 2005	90
A.7. Durch eine retrospektive Analyse erkannte historische Cluster. Die Berechnung hat vom 1. September 2005 an rückblickend alle Fälle des letzten Jahres eingeschlossen.	91

Tabellenverzeichnis

3.1. Geforderte Produktqualität nach Kategorien	22
4.1. Ergebnisse einer weltweiten Umfrage über die Verbreitung des Flash-Plugins [23] (Stand: September 2005)	32
5.1. Werte für die Antwortzeiten mit und ohne Cache	54
6.1. Klassen im Paket Export und ihre Aufgaben	59
6.2. Implementierte Struts-Aktionen in EpiDeGIS-Webanwendung	61
6.3. Rechenzeiten zur Zählung der Erkrankungsfälle in einem bestimmten Gebietstyp	63
6.4. Durch Zuordnungstabelle optimierte Rechenzeit (vgl. Tabelle 6.3) . . .	64

Listings

4.1.	Beispiel für das obligatorische Hallo-Welt-Programm in OpenLaszlo	38
4.2.	Minimales ereignisgesteuertes LZX-Programm	38
5.1.	SQL-Abfrage als Datenquelle für eine Kartenebene	52
6.1.	Codefragment zum Einsatz der <i>XmlExporter</i> -Klasse	59
6.2.	Execute-Methode der <i>MoveMapAction</i> -Klasse	60
6.3.	Definition einer Aktion im Struts-Framework	60
6.4.	Definition der Datenquelle in der MapFile	62
6.5.	SQL-Query zur Ermittlung der Fallzahl in Bayern	63
6.6.	Erzeugung von Kreisen mit Hilfe von PostGIS	65
6.7.	Zufällige Streuung von Punktinformationen über die SQL-Query	66
6.8.	Replikation von Elementen in OpenLaszlo durch “databinding”	67
6.9.	Fragment des Kartenbetrachter-Quellcodes zur dynamischen Erzeugung der GUI	68
6.10.	Fragment des Kartenbetrachter-Quellcodes zur Kommunikation mit Struts	69
6.11.	Antwort des Servers auf eine <i>SetParameter</i> -Aktion	70
A.1.	Wichtige Makefile-Einträge für die PostGIS-Kompilation	82
A.2.	Befehl zur Erweiterung einer Datenbank mit PostGIS-Funktionen und Tabellen	83
A.3.	Befehl zur Konfiguration der MapServer-Quellen	83
A.4.	Ausgabe zur Verifikation der MapServer-Funktionalität	84
D.1.	Kommentiertes Ausgabeformat von SaTScan	97
D.2.	Beispiel einer Konfigurationsdatei für den MapServer mit WMS-Unterstützung und einer Ebene	97
D.3.	MapServer-Datenquelle für die Inzidenz-Ebene	99
D.4.	Eine abgespeckte Karte in XML-Form	100

Literatur- und Quellenverzeichnis

- 1 *Adobe Systems Incorporated.* <http://www.adobe.com/>. – Online–Ressource, Abruf: 8. Dez. 2005
- 2 *Apache Maven.* <http://maven.apache.org/>. – Online–Ressource, Abruf: 5. Dez. 2005
- 3 *Apache Struts Project.* <http://struts.apache.org/>. – Online–Ressource, Abruf: 10. Dez. 2005
- 4 *Apache Tomcat Servlet Container.* <http://tomcat.apache.org/>. – Online–Ressource, Abruf: 5. Dez. 2005
- 5 *Arbeitsgemeinschaft Influenza.* <http://www.influenza.rki.de/agii/>. – Online–Ressource, Abruf: 5. Dez. 2005
- 6 *Ausbreitung der Vogelgrippe (Stand Oktober 2005).* <http://de.wikipedia.org/wiki/Bild:VogelgrippeWelt.png>. – Online–Ressource, Abruf: 20. Dez. 2005. – Dieses Bild ist unter der GNU-Lizenz für freie Dokumentation veröffentlicht.
- 7 *Core J2EE Patterns - Data Access Object.* <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>. – Online–Ressource, Abruf: 15. Dez. 2005
- 8 *Die Definition Freier Software.* <http://www.gnu.org/philosophy/free-sw.de.html>. – Online–Ressource. – Copyright © 1996, 1997, 1998, 1999, 2000, 2001, 2002 Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110, USA
- 9 *Eclipse Community.* <http://www.eclipse.org/>. – Online–Ressource, Abruf: 5. Dez. 2005
- 10 *ESRI Shapefile Technical Description.* <http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf>. – Online–Ressource, Abruf: 22. Dez. 2005. – An ESRI White Paper, July 1998
- 11 *Free Software Foundation.* <http://www.fsf.org/>. – Online–Ressource, Abruf: 10. Dez. 2005
- 12 *Getting Started Guide.* <http://maven.apache.org/guides/getting-started/index.html>. – Online–Ressource, Abruf: 5. Dez. 2005. – Apache Maven Projekt
- 13 *H5N1 avian influenza: timeline.* http://www.who.int/csr/disease/avian_influenza/Timeline_28_10a.pdf. – Online–Ressource, Abruf: 20. Dez. 2005

Literatur- und Quellenverzeichnis

- 14 International Data Corporation. <http://www.idc.com/>. – Online–Ressource, Abruf: 8. Dez. 2005
- 15 Jakarta Commons. <http://jakarta.apache.org/commons/>. – Online–Ressource, Abruf: 5. Dez. 2005
- 16 Java Technology. <http://java.sun.com/>. – Online–Ressource, Abruf: 8. Dez. 2005
- 17 Java Web Start Technology. <http://java.sun.com/products/javawebstart/>. – Online–Ressource, Abruf: 20. Dez. 2005
- 18 JavaBeans Technology. <http://java.sun.com/products/javabeans/index.jsp>. – Online–Ressource, Abruf: 13. Dez. 2005
- 19 JavaServer Pages Technology. <http://java.sun.com/products/jsp/>. – Online–Ressource, Abruf: 10. Dez. 2005
- 20 Laszlo Documentation. <http://www.laszlosystems.com/lps-3.0/docs/>. – Online–Ressource, Abruf: 18. Dez. 2005
- 21 Laszlo Systems Media Kit. http://www.laszlosystems.com/company/press/media_kit.php. – Online–Ressource, Abruf: 10. Dez. 2005. – Copyright © 2000–2005 Laszlo Systems, Inc., 2600 Campus Drive Suite 200, San Mateo, California, 94403, USA. All Rights Reserved.
- 22 Liste der Nationalen Referenzzentren und Konsiliarlaboratorien in der aktuellen Berufungsperiode (2005 bis 2007). <http://www.rki.de/>. – Online–Ressource, Abruf: 24. Nov. 2005. – Robert Koch-Institut
- 23 Macromedia Flash Player Version Penetration. http://www.macromedia.com/software/player_census/flashplayer/version_penetration.html. – Online–Ressource, Abruf: 8. Dez. 2005. – Umfrage der NPD Group
- 24 Mapbender. <https://sourceforge.net/projects/mapbender/>. – Online–Ressource, Abruf: 22. Dez. 2005
- 25 Nationales Referenzzentrum für Meningokokken. <http://www.meningococcus.de/>. – Online–Ressource, Abruf: 21. Dez. 2005
- 26 Open Geospatial Consortium, Inc. <http://www.opengeospatial.org/>. – Online–Ressource, Abruf: 11. Dez. 2005
- 27 Open Source Initiative. <http://www.opensource.org/>. – Online–Ressource, Abruf: 10. Dez. 2005
- 28 OpenLaszlo. <http://www.openlaszlo.org/>. – Online–Ressource, Abruf: 10. Dez. 2005
- 29 Poseidon for UML - by Gentleware. <http://gentleware.com/>. – Online–Ressource, Abruf: 22. Dez. 2005. – Erstellt mit Poseidon for UML Community Edition. Nicht zur kommerziellen Nutzung.
- 30 PostGIS Manual. <http://postgis.refractions.net/docs/postgis.pdf>. – Online–Ressource, Abruf: 18. Dez. 2005. – Refractions Research Inc.
- 31 PostgreSQL database system. <http://www.postgresql.org/>. – Online–Ressource, Abruf: 5. Dez. 2005

- 32** *The R Project for Statistical Computing.* <http://www.r-project.org/>. – Online-Ressource, Abruf: 11. Dez. 2005
- 33** *Ridom: Software for the Bioinformatics Field.* <http://www.ridom.de/>. – Online-Ressource, Abruf: 21. Dez. 2005
- 34** *Scalable Vector Graphics.* <http://www.w3.org/Graphics/SVG/>. – Online-Ressource, Abruf: 20. Dez. 2005
- 35** *Smittskyddsinstitutet.* <http://www.smittskyddsinstitutet.se/>. – Online-Ressource, Abruf: 5. Dez. 2005
- 36** *Statistisches Bundesamt Deutschland.* <http://www.destatis.de/>. – Online-Ressource, Abruf: 22. Dez. 2005
- 37** *Struts Action Framework - User Guide.* <http://struts.apache.org/struts-action/userGuide/>. – Online-Ressource, Abruf: 18. Dez. 2005. – The Apache Software Foundation
- 38** *SurvStat@RKI.* <http://www3.rki.de/SurvStat/>. – Online-Ressource, Abruf: 5. Dez. 2005. – Robert Koch-Institut
- 39** *The Apache Jakarta Project.* <http://jakarta.apache.org/>. – Online-Ressource, Abruf: 5. Dez. 2005
- 40** *The Apache Software Foundation.* <http://www.apache.org/>. – Online-Ressource, Abruf: 5. Dez. 2005
- 41** *User-friendly Desktop Internet GIS.* <http://udig.refractions.net/confluence/display/UDIG/Home>. – Online-Ressource, Abruf: 14. Dez. 2005
- 42** *XML User Interface Language (XUL) Project.* <http://www.mozilla.org/projects/xul/>. – Online-Ressource, Abruf: 20. Dez. 2005
- 43** *OpenGIS Simple Features Specification For SQL.* Mai 1999. – White Paper der Open Geospatial Consortium, Inc.
- 44** *Web Map Service.* Aug. 2004. – White Paper der Open Geospatial Consortium, Inc.
- 45** Invasive Meningokokken-Erkrankungen im Jahr 2004. In: *Epidemiologisches Bulletin* (2005), Nr. 34
- 46** Navigating the Geospatial Timeline. In: *Geospatial Solutions* 15 (2005), Nr. 7, S. 26–39
- 47** ANSELIN, L. : Review of Cluster Analysis Software / Anselin and Associates, LLC. 2004. – Forschungsbericht
- 48** BALZERT, H. : *Lehrbuch der Software-Technik.* 2. Spektrum Akademischer Verlag, 2000 (Lehrbücher der Informatik)
- 49** BARNES, S. : The Many Layers of GIS. In: *Geospatial Solutions* 15 (2005), Jul., Nr. 7, S. 21–29
- 50** BGBL I 2000, 1045 (Hrsg.): *Gesetz zur Verhütung und Bekämpfung von Infektionskrankheiten beim Menschen.* Bundesministerium für Gesundheit und Soziale

Literatur- und Quellenverzeichnis

Sicherung, 2000. – Sachgebiet: FNA 2126-13, GESTA M020, Stand: Zuletzt geändert durch Art. 2 § 3 Abs. 4 G v. 1. 9.2005 I 2618

- 51 CHUNG, K. ; YANG, D.-H. ; BELL, R. : Health and GIS: Toward Spatial Statistical Analyses. In: *Journal of Medical Systems* 28 (2004), Aug., Nr. 4, S. 349–359
- 52 CLARK, J. ; DEROSSE, S. : *XML Path Language (XPath) Version 1.0.* – W3C Recommendation
- 53 CONWAY, J. E.: *PL/R - R Procedural Language for PostgreSQL.* <http://www.joeconway.com/plr/>. – Online-Ressource, Abruf: 11. Dez. 2005
- 54 DOYON, J.-F. ; MCKENNA, J. : *MapFile Reference.* Version: MapServer 4.6, Jun. 2005. <http://mapserver.gis.umn.edu/doc46/mapfile-reference.html>. – Online-Ressource
- 55 DUHL, J. : *The Business Impact of Rich Internet Applications.* Apr. 2003. – White Paper
- 56 EVENDEN, G. : *PROJ.4 - Cartographic Projections Library.* <http://www.remotesensing.org/proj/>. – Online-Ressource, Abruf: 5. Dez. 2005
- 57 HEYWOOD, I. ; CORNELIUS, S. ; CARVER, S. : *An Introduction to Geographical Information Systems.* Prentice Hall, 1999 (Geographic Information Science). – ISBN 0 130 16238 8
- 58 HOLUB, A. : Why getter and setter methods are evil. In: *JavaWorld* (2003), Sept.
- 59 HOLUB, A. : More on getters and setters. In: *JavaWorld* (2004), Jan.
- 60 KÜHN, S. : *Abbildungen der Mercator-Projektion, Azimutalprojektion mit normaler Lage, Kegelprojektion und Zylinderprojektion.* http://de.wikipedia.org/wiki/Benutzer:Stefan_K%C3%BChn. – Online-Ressource, Abruf: 11. Dez. 2005. – Diese Bilder sind unter der GNU-Lizenz für freie Dokumentation veröffentlicht.
- 61 KUHN, T. : *Entwicklung von Webapplikationen mit dem Struts Framework.* <http://www.tkuhn.de/informatik/>. – Online-Ressource, Abruf: 10. Dez. 2005. – Ausarbeitung für ein Seminar an der Universität Karlsruhe
- 62 KULLDORFF, M. : A spatial scan statistic. In: *Communications in Statistics: Theory and Methods* (1997), Nr. 26, S. 1481–1496
- 63 KULLDORFF, M. ; ATHAS, W. F. ; FEUER, E. J. ; MILLER, B. A. ; KEY, C. R.: Evaluating cluster alarms: A space-time scan statistic and brain cancer in Los Alamos. In: *American Journal of Public Health* 88 (1998), Sept., Nr. 9, S. 1377–1380
- 64 KULLDORFF, M. ; ATHAS, W. F. ; FEUER, E. J. ; MILLER, B. A. ; KEY, C. R.: Prospective time-periodic geographical disease surveillance using a scan statistic. In: *Journal of the Royal Statistical Society* (2001), Nr. 164, S. 61–72
- 65 KULLDORFF, M. ; HEFFERNAN, R. ; HARTMAN, J. ; ASSUNÇAO, R. ; MOSTASHARI, F. : A space-time permutation scan statistic for the early detection of disease outbreaks. In: *PLoS Medicine* 2 (2005), März, Nr. 3, S. 216–224

- 66** KULLDORFF, M. ; INFORMATION MANAGEMENT SERVICES, INC.: *SaTScanTM - Software for the spatial, temporal, and space-time scan statistics.* Version: 2005. <http://www.satscan.org/>. – Online-Ressource, Abruf: 5. Dez. 2005
- 67** LAST, J. M. (Hrsg.): *A Dictionary of Epidemiology*. Oxford University Press, 2000
- 68** LIME, S. : *MapServer*. <http://mapserver.gis.umn.edu/>. – Online-Ressource, Abruf: 10. Dez. 2005
- 69** LUDWIG, A. : *Macromedia Flash Player 8 Security.* Version: Sept. 2005. http://www.macromedia.com/devnet/flashplayer/articles/flash_player_8_security.pdf. – Online-Ressource, Abruf: 13. Dez. 2005. – Whitepaper
- 70** MACGILL, J. : *GeoTools - The open source Java GIS toolkit*. <http://www.geotools.org/>. – Online-Ressource, Abruf: 11. Dez. 2005
- 71** NORSTRÖM, M. ; PFEIFFER, D. U. ; JARP, J. : A space-time cluster investigation of an outbreak of acute respiratory disease in Norwegian cattle herds. In: *Preventive Veterinary Medicine* (1999), Nov., Nr. 47, S. 107–119
- 72** PANITZKI, M. : *Navigation*. http://home.arcor.de/m.panitzki/html/navigation/index_navigation.htm. – Online-Ressource, Abruf: 1. Dez. 2005
- 73** RAMSEY, P. ; REFRACTIONS RESEARCH INC. (Hrsg.): *Geometry Engine Open Source*. <http://geos.refractions.net/>. – Online-Ressource, Abruf: 5. Dez. 2005
- 74** RAMSEY, P. : *The State of Open Source GIS*. Febr. 2005. – White Paper
- 75** REED, C. : Open Standards-Based Architectures Scale Up. In: *Geospatial Solutions* 15 (2005), März, Nr. 3, S. 36–41
- 76** REENSKAUG, T. : MVC / Xerox Palo Alto Research Center. Version: 1978. <http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html>. – Forschungsbericht. – Elektronische Ressource
- 77** ROLFHAMRE, P. ; GRABOWSKA, K. ; EKDAHL, K. : Implementing a public web based GIS service for feedback of surveillance data on communicable diseases in Sweden. In: *BMC Infectious Diseases* 4 (2004), Jun., Nr. 17
- 78** SANTILLI, S. ; HODGSON, C. ; RAMSEY, P. ; LOUNSBURY, J. ; REFRACTIONS RESEARCH (Hrsg.): *PostGIS*. <http://postgis.refractions.net/>. – Online-Ressource, Abruf: 10. Dez. 2005
- 79** SCHRAUDER, A. ; CLAUS, H. ; ELIAS, J. ; VOGEL, U. ; HAAS, W. ; HELLENBRAND, W. : *Capture-Recapture Analysis to estimate the Incidence of Invasive Meningococcal Disease in Germany, 2003*. Dez. 2005. – Manuskript in Vorbereitung
- 80** SESHDARI, G. : Understanding JavaServer Pages Model 2 architecture. In: *Java-World* (1999), Dez.
- 81** WARTALA, R. : Apaches Jakarta-Basisbibliothek Commons - Die gemeinsame Feder. In: *iX Magazin für professionelle Informationstechnik* (2005), Jan., Nr. 1, S. 70–74
- 82** WILK, C. : Arbeiten mit Google Earth und World Wind. In: *iX Magazin für professionelle Informationstechnik* (2005), Dez., Nr. 12, S. 50–62

Literatur- und Quellenverzeichnis

- 83** WOLFF, A. : *Our Feature Presentation*. <http://secretartofscience.com/blog/?p=19>. – Online-Ressource, Abruf: 21. Dez. 2005
- 84** WOLFF, A. : *Programs as Data*. <http://secretartofscience.com/blog/?p=9>. – Online-Ressource, Abruf: 18. Dez. 2005