Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Marcus Rohwer

Bearbeitung von skizzierten Landkarten und ihre Einbettung in Kartendienste unter Android

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Marcus Rohwer

Bearbeitung von skizzierten Landkarten und ihre Einbettung in Kartendienste unter Android

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Olaf Zukunft
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Abgegeben am 07.12.2011

Marcus Rohwer

Thema der Bachelorarbeit
Bearbeitung von skizzierten Landkarten und ihre Einbettung in Kartendienste unter Android
Stichworte
Geoinformatik, Java, Android
Kurzzusammenfassung
Ziel dieser Arbeit ist die Einbettung von handschriftlich erstellten Landkarten in Kartendienste auf mobilen Endgeräten. Zu diesem Zweck werden zwei Anwendungen konzipiert, realisiert und bewertet. Die erste Anwendung öffnet eine skizzierte Landkarte und nähert sie durch Koordinatentransformation an die Realität an, so dass sie auf die exakte Karte eines Kartendienstes projiziert werden kann. Die zweite Anwendung läuft auf der Android-Plattform und realisiert die Einbettung der bearbeiteten Skizze in den Kartendienst Google Maps.

Marcus Rohwer

Title of the paper
Processing of drafted maps and their embedding in web mapping services under Android
Keywords
Geoinformatics, Java, Android
Abstract
The objective of this thesis is the embedding of handwritten maps in web mapping services on mobile end devices. For this purpose two applications are devised, implemented and evaluated. The first application opens a drafted map and draws it near to reality through transformation of coordinates that enables its projection onto the exact map of a web mapping service. The second application is based on the Android-platform and realises the embedding of the processed sketch in the web mapping service Google Maps.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der eine Gegenstand dieser Arbeit sind skizzierte Landkarten. Der andere Gegenstand sind Kartendienste wie Google Maps oder OpenStreetMap, die die Welt präzise abbilden. Ziel ist es, die von Menschenhand erstellte ungenaue Skizze in einen Kartendienst einzubetten. Dazu muss zwischen diesen beiden Ebenen vermittelt werden, indem die Skizze der Realität angenähert wird. Die Einbettung der Skizze in den Kartendienst soll auf einem Android-Endgerät stattfinden.

1.1 Motivation

Verschiedene Szenarien sind denkbar, in denen es von Nutzen ist, eine skizzierte Karte in einen Kartendienst einbetten und diese Information dann auf einem mobilen Endgerät abrufen zu können. Dies kann im spielerischen Umfeld aber auch mit ernstem Hintergrund erfolgen:

- für einen Kindergeburtstag wurde eine Schatzkarte gezeichnet
- eine fiktive Spielwelt soll in der realen Welt abgebildet werden
- eine historische Karte soll über eine aktuelle Karte gelegt werden
- nach einer Katastrophe gibt eine Skizze Auskunft über verbleibende Infrastruktur

Die Verknüpfung einer Skizze mit einem Kartendienst reichert diesen mit zusätzlicher Information an. Andersherum lässt sich die Skizze genau in die reale Welt einordnen.

1.2 Ziele

Es sollen zwei Anwendungen konzipiert und implementiert werden:

- Mit der ersten Anwendung soll eine handschriftliche Skizze geöffnet und so transformiert werden, dass sie der Realität angenähert ist. Sie soll dann in einer Form vorliegen, in der sie über eine exakte Karte eines Kartendienstes gelegt werden kann. Dabei soll ein Objekt, das in der Realität die geografischen Koordinaten xy hat, beim Auflegen auf die exakte Karte möglichst auch an diesen geografischen Koordinaten abgebildet werden.
- Die zweite Anwendung soll auf Android-Endgeräten zum Einsatz kommen. Sie soll die transformierte Skizze der ersten Anwendung auf einen Kartendienst projizieren.

Auf diese Weise soll der Schritt von einer skizzierten Landkarte zu einer mobil nutzbaren Anwendung erfolgen, die die Skizze mit einem Kartendienst verbindet.

1.3 Gliederung

Im folgenden Kapitel sollen die Grundlagen erarbeitet werden. Für die erste Anwendung wird der Begriff *Landkarte* vorgestellt. Methoden werden aufgezeigt, mit denen sich diese Karten transformieren lassen. Für die zweite Anwendung wird die Softwareplattform Android vorgestellt.

Im Kapitel drei wird die Anwendung zur Bearbeitung der Skizze konzipiert und realisiert. Es wird ein fachliches und ein technisches Konzept erarbeitet. Anschließend wird die Implementierung geschildert. Diese wird dann im folgenden Abschnitt getestet. Im Kapitel vier erfolgen diese Schritte für die Anwendung zur Darstellung der Skizze unter Android.

Kapitel fünf fasst dann die Umsetzung der funktionalen und nichtfunktionalen Anforderungen durch die beiden erstellten Anwendungen zusammen und bewertet diese.

Eine Zusammenfassung der Arbeit erfolgt im Kapitel sechs. Darüberhinaus wird ein Ausblick auf mögliche Ansatzpunkte für darauf aufbauende weitere Arbeiten gegeben.

Kapitel 2

Grundlagen

2.1 Landkarten

Karten sind die geläufigsten Träger von geografischer Information. Es gibt sie vor allem in gedruckter Form, aber vermehrt auch als digitalisierte Karten, die mit entsprechenden Anwendungen betrachtet werden können. Karten sind Berichte, Repräsentationen oder Visualisierungen zu Grunde liegender Daten auf einer zweidimensionalen Oberfläche. Sie beinhalten eine Auswahl von Daten für eine bestimmte Zielsetzung und werden gemäß Gestaltungsgrundsätzen der Kartografie in Bezug auf Abstraktion und Verwendung von Symbolen, Farben, Markierungen, Linientypen sowie Beschriftungen erstellt. Karten sind das Endprodukt der Sammlung, Verarbeitung und Analyse von geografisch referenzierten Daten, die die Lage und Beschaffenheit der Landschaft dokumentieren (vgl. Hill 2006, S. 36). Abbildung 2.1 auf der nächsten Seite zeigt eine Landkarte zum Thema Bodennutzung.

Auf einer Karte werden räumliche Objekte, auch Geoobjekte genannt, dargestellt. Sie treten in folgenden Ausprägungen auf (de Lange 2005, S. 159):

> Punkte z.B. Grenzstein, Zähl- oder Messstelle, Quellort eines Emittenten
>
> Linien z.B. Profillinie, Grenzlinie, Baumreihe, Wasserleitung, Verbindungslinie
>
> Flächen z.B. Flurstück, Biotop, Gemeindegebiet, Einzugsgebiet
>
> Körper z.B. Schadstoffwolke, Grundwasserkörper, Lagerstätte, Gebäude [da sich diese Arbeit mit einer zweidimensionalen Darstellung befasst, entfällt der Objekttyp Körper, M.R.]

„Die Darstellung der Geoobjekte erfolgt im sog. Vektormodell oder im sog. Rastermodell" (de Lange 2005, S. 161). In beiden Modellen bildet ein kartesisches Koordinatensystem die Grundlage zur Darstellung der Geoobjekte (vgl. de Lange 2005, S. 167).

Abbildung 2.1: Landkarte zum Thema Bodennutzung (vgl. Diercke 2002, S. 24)

Im Vektormodell wird das Geoobjekt durch gerichtete Strecken (Vektoren) ausgezeichnet. Diese werden durch Angabe von Anfangs- und Endpunkt festgelegt. Ein Punkt wird durch einen Vektor, der seinen Anfang im Ursprung des Koordinatensystems hat, gekennzeichnet. Die Verbindung zweier Punkte ist ebenfalls ein Vektor, der durch Anfangs- und Endpunkt beschrieben wird. Die Angabe von Vektorkoordinaten ist zur Beschreibung der Geoobjekte jedoch nicht ausreichend. Es müssen zusätzliche Informationen gespeichert werden, aus denen hervorgeht, welche Punkte welche Linie definieren und welche Linien welche Flächen bilden (vgl. de Lange 2005, S. 161). Abbildung 2.2 auf der nächsten Seite veranschaulicht das Vektormodell.

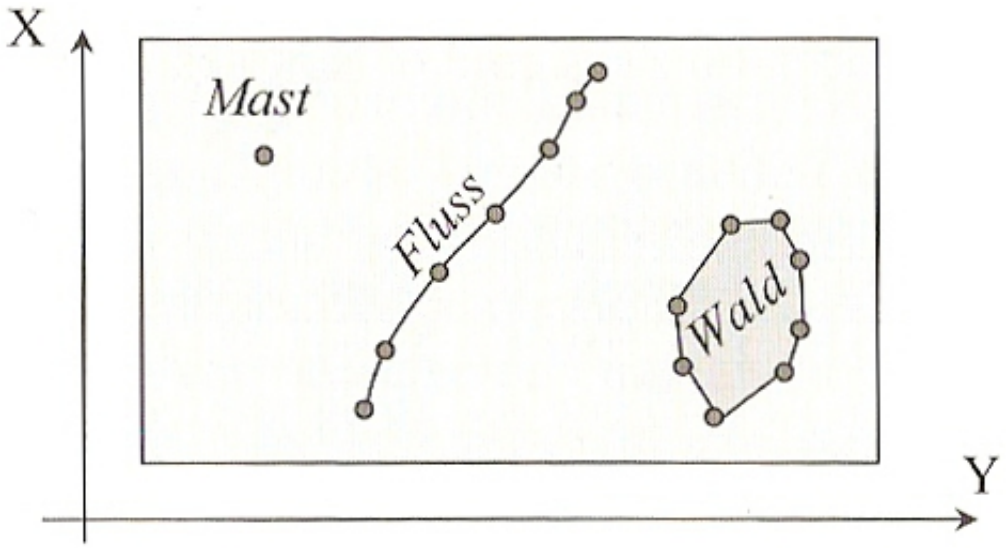

Abbildung 2.2: Punkt, Linie und Fläche im Vektormodell (vgl. Resnik und Bill 2003, S. 203)

Das Rastermodell wird aus einem Mosaik aus Flächen fester Form und Größe gebildet. Dies sind i. d. R. Quadrate bzw. quadratische Pixel. Somit liegt automatisch ein kartesisches Koordinatensystem zugrunde. Ein Punkt wird durch ein einzelnes Pixel angenähert. Linien und Flächen werden aus Anordnungen zusammenhängender Pixel gebildet. Im Gegensatz zum Vektormodell sind zur Darstellung der Geoobjekte keine weiteren Zusatzinformationen nötig (vgl. de Lange 2005, S. 162-163). Die Abbildung 2.3 auf der nächsten Seite zeigt das Rastermodell.

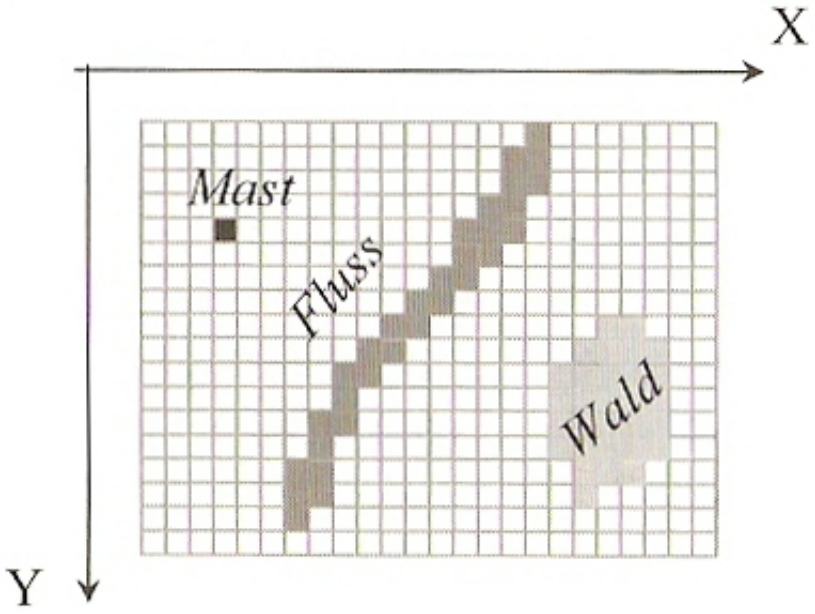

Abbildung 2.3: Punkt, Linie und Fläche im Rastermodell (vgl. Resnik und Bill 2003, S. 203)

Die beiden Modelle weisen keine eindeutigen Vor- bzw. Nachteile auf. Fragestellungen können sowohl mit dem Vektor- als auch mit dem Rastermodell bearbeitet werden (vgl. de Lange 2005, S. 335). Die Tabellen 2.1 und 2.2 zeigen die jeweiligen Vor- und Nachteile auf.

Vektormodell - Vorteile	**Vektormodell - Nachteile**
hohe geometrische Genauigkeit	komplexere Datenstrukturen
eindeutige Objektbeschreibung	aufwändige Erfassung von Geometrie und Topologie
geringe Datenmengen	aufwändige und rechenintensive logische und algebraische Operationen
größere Ähnlichkeit der graphischen Präsentation mit traditionellen Karten	parallele geometrische und topologische Beschreibung der Geoobjekte

Tabelle 2.1: Vor- und Nachteile des Vektormodells (vgl. de Lange 2005, S. 336)

Rastermodell - Vorteile	**Rastermodell - Nachteile**
einfache Datenstrukturen	keine Form- und Lagetreue der Geoobjekte
geringer Aufwand bei Erfassung der Geometrie und Topologie	höherer Speicheraufwand
	kleine Pixelgröße mit explodierenden Datenmengen für höhere Genauigkeitsanforderungen
kompatibel mit Fernerkundungs- und Scannerdaten	
	weniger zufrieden stellende graphische Präsentation (abhängig von der Pixelgröße)
einfaches Überlagern und Verschneiden von Geoobjekten	
	aufwändige Koordinatentransformationen
einfache logische und algebraische Operationen	

Tabelle 2.2: Vor- und Nachteile des Rastermodells (vgl. de Lange 2005, S. 336)

2.2 Verfahren zur Koordinatentransformation in der Ebene

Die Lage von Geoobjekten werden in irgendeinem rechtwinkligen Koordinatensystem erfasst, abhängig von der jeweiligen Methode. Oft interessieren aber nicht diese Koordinaten, sondern die Koordinaten in einem anderen Koordinatensystem. Aus diesem Grund ist eine Umrechnung der erfassten Koordinaten erforderlich (vgl. de Lange 2005, S. 171). Grundlage ist dabei die Erfassung von Pass- bzw. Referenzpunkten. Dies sind Punkte, deren Koordinaten sowohl im Ausgangs- wie im Zielkoordinatensystem vorliegen. Die hier vorgestellten Verfahren zur Koordinatentransformation sind dem Buch Luhmann 2010 (S. 27-33) entnommen.

2.2.1 Ähnlichkeitstransformation

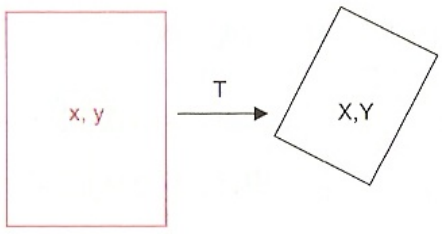

Abbildung 2.4: Ähnlichkeitstransformation (Luhmann 2010, S. 27)

Bei dieser Transformation wird das vorliegende Koordinatensystem in X- und Y-Richtung um die Beträge a_0 bzw. b_0 verschoben. Es wird um den Winkel α gedreht und mit dem Maß-

stabsfaktor m verkleinert bzw. vergrößert. Winkel- und Streckenverhältnisse bleiben erhalten. Man benötigt mindestens zwei Passpunkte, um die vier Transformationsparameter berechnen zu können. Für einen Punkt mit den Koordinaten x_s und y_s werden seine transformierten X- und Y-Werte folgendermaßen berechnet:

$$X = a_0 + m(x_s cos\alpha - y_s sin\alpha)$$

$$Y = b_0 + m(x_s sin\alpha + y_s cos\alpha)$$

Diese Gleichungen lassen sich in lineare Form umformen, indem man folgende Parameter einführt:

$$a_1 = m * cos\alpha$$

$$b_1 = m * sin\alpha$$

Jetzt lässt sich die Transformation durch dieses lineare Gleichungssystem beschreiben:

$$X = a_0 + a_1 x_s - b_1 y_s$$

$$Y = b_0 + b_1 x_s + a_1 y_s$$

2.2.2 Affintransformation

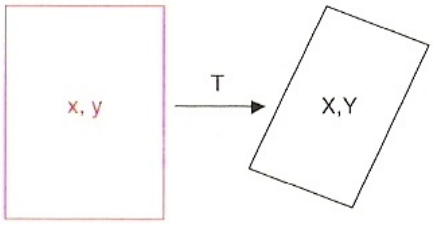

Abbildung 2.5: Affintransformation (Luhmann 2010, S. 29)

Bei der Affintransformation bleiben die Parameter a_0, b_0 und α der Ähnlichkeitstransformation erhalten. Jedoch gibt es hier zwei Maßstabsfaktoren, m_x für die X-Richtung und m_y für die

Y-Richtung. Die beiden Koordinatenachsen werden außerdem um den Scherungswinkel β versetzt, sind also nicht mehr rechtwinklig zueinander. Es werden mindestens drei Passpunkte benötigt um die sechs Transformationsparameter zu berechnen. Die X- und Y-Koordinaten werden folgendermaßen transformiert:

$$X = a_0 + m_x x_s cos\alpha - m_y y_s sin(\alpha + \beta)$$

$$Y = b_0 + m_x x_s sin\alpha + m_y y_s cos(\alpha + \beta)$$

Auch diese Gleichungen lassen sich in lineare Form überführen, indem man folgende Parameter einführt:

$$a_1 = m_x cos\alpha$$

$$a_2 = -m_y sin(\alpha + \beta)$$

$$b_1 = m_x sin\alpha$$

$$b_2 = m_y cos(\alpha + \beta)$$

Jetzt lässt sich die Transformation durch dieses lineare Gleichungssystem beschreiben:

$$X = a_0 + a_1 x_s + a_2 y_s$$

$$Y = b_0 + b_1 x_s + b_2 y_s$$

2.2.3 Weitere Arten der Transformation

Koordinatentransformationen können außerdem als Polynomtransformation, bilineare Transformation und Projektivtransformation durchgeführt werden.

Mit der Polynomtransformation können auch nicht-lineare Verformungen durchgeführt werden. Liegt der Transformation ein Polynom ersten Grades zugrunde, entspricht dies der Affintransformation. Um eine Transformation auf Basis eines Polynoms zweiten Grades durchzuführen, sind zwölf Parameter zu bestimmen, wozu mindestens sechs Passpunkte benötigt werden.

Bei der bilinearen Transformation wird die Affintransformation um ein gemischtes Glied erweitert. Es werden acht Parameter benötigt, für deren Bestimmung vier Passpunkte vorliegen müssen. Anwendung findet diese Transformationsart bei der zwangsfreien Transformation und Interpolation von Viereckmaschen.

Bei der ebenen Projektivtransformation werden zwei Koordinatensysteme zentralprojektiv aufeinander abgebildet. Die Abbildungsstrahlen durchlaufen dabei geradlinig das Projektionszentrum. Bei dieser Transformationsmethode bleiben Geradlinigkeit und Schnittpunkte von Geraden erhalten. Winkel, Strecken- und Flächenproportionen werden jedoch verändert. Es werden vier Passpunkte benötigt, um die acht Transformationsparameter zu bestimmen.

2.2.4 Ausgleichsrechnung

Liegen mehr Passpunkte vor, als für das jeweilige Transformationsverfahren erforderlich, lässt sich das Ergebnis optimieren. Die Transformationsparameter werden iterativ mit den verschiedenen Kombinationen von Passpunkten bestimmt. Für die jeweils berechneten Transformationsparameter werden die Abweichungen, mit denen die übrigen Passpunkte ins Zielsystem transformiert werden, überprüft. Es werden dann die Passpunkte zur Bestimmung der Transformationsparameter ausgewählt, bei denen die Summe dieser Abweichungen am geringsten ausgefallen ist (vgl. de Lange 2005, S. 175-176).

2.3 Android

In diesem Abschnitt wird die Softwareplattform Android vorgestellt. Die folgenden Informationen sind jeweils dem ersten Kapitel der Bücher Mosemann und Kose 2009 sowie Becker und Pant 2010 entnommen.

2.3.1 Der Start von Android

Am 05. November 2007 haben sich 34 Unternehmen zur Open Handset Alliance (OHA) zusammengeschlossen. Die OHA besteht aus Netzbetreibern, Halbleiterfirmen, Geräteherstellern, Softwarefirmen und Vermarktungsfirmen und ist bis heute auf 84 Unternehmen (vgl. open handset alliance 2011) angewachsen. Das Anliegen der OHA ist die Enwicklung der Softwareplattform Android. Treibende Kraft ist dabei die Firma Google. Nach der Ankündigung von Android im November 2007 stellte die Firma HTC 2008 das erste Android-Endgerät vor.

2.3.2 Eigenschaften von Android

Android bildet eine umfassende Softwareplattform für mobile Endgeräte wie z. B. Smartphones und Tablet-PCs. Diese Plattform verfügt über ein auf Linux basierendes Betriebssystem, umfangreiche Bibliotheken, eine Laufzeitumgebung und mobile Schlüsselapplikationen. Durch den Einsatz von Android fallen keine Kosten wie z. B. Lizenzgebühren an.

Da große Teile der Android-Plattform sich unter der Open-Source-Lizenz befinden, können Entwickler nicht nur auf die angebotenen Programmierschnittstellen zurückgreifen, sondern auch den Quellcode einsehen. Sie können das Android-Plugin für Eclipse nutzen, wodurch sie komfortablen Zugriff auf das von Google veröffentlichte Android-SDK haben. Die von ihnen erzeugten Anwendungen sind gleichberechtigt mit den vorinstallierten Anwendungen. Bei Android kommt die weit verbreitete Programmiersprache Java zum Einsatz.

Die Android-Endgeräte sollen typischerweise handlich und leicht sein, daher ergeben sich folgende Einschränkungen:

- Aufgrund der Batterietechnologie steht Energie nur begrenzt zur Verfügung.
- Um Platz, Gewicht und Energie zu sparen, ist auch die Speicher- und Bildschirmgröße begrenzt.
- Da ein mobiles Endgerät nicht standortgebunden ist, kommt es zu schwachen bzw. unterbrochenen Netzwerkverbindungen. Dies muss bei der Auslegung der Antenne sowie der Fehlertoleranz der Software beachtet werden.

2.3.3 Die Architektur von Android

Die Android-Systemarchitektur, veranschaulicht in Abbildung 2.6 auf der nächsten Seite, setzt sich aus fünf Komponenten zusammen. Angefangen bei hardwarenahen Treibern in der Linux-Kernel-Schicht bis hin zu fertigen Applikationen in der Anwendungsschicht.

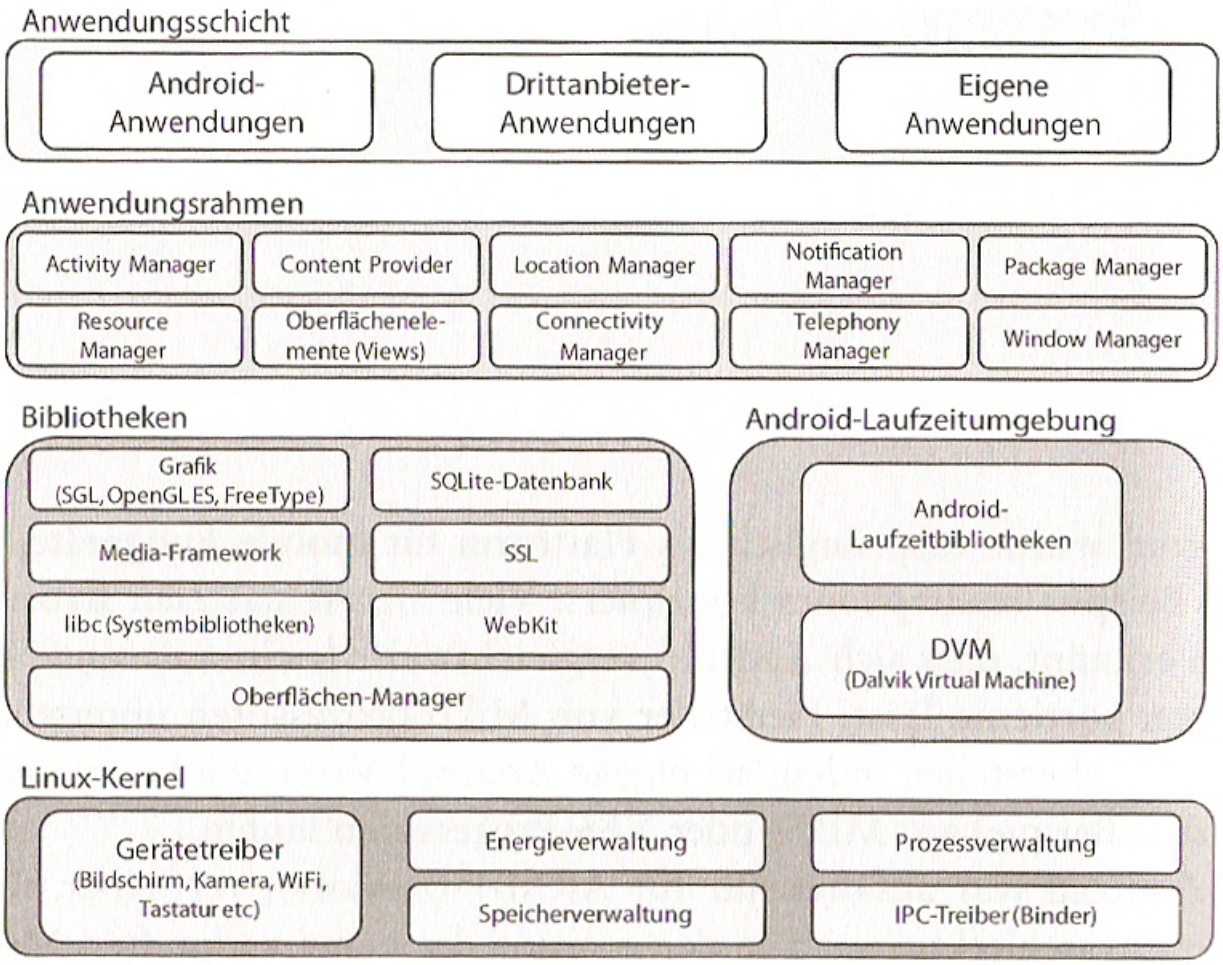

Abbildung 2.6: Die Android-Systemarchitektur (Becker und Pant 2010, S. 20)

Linux-Kernel Android basiert auf einem Linux-Kernel der Version 2.6. Dieser besitzt die benötigten Gerätetreiber und verfügt über einige Optimierungen z. B. in Bezug auf Energieverbrauch und Speichermanagement. Somit wird dem übrigen System eine Hardware-Abstraktionsschicht zur Verfügung gestellt.

Android-Laufzeitumgebung Die Android-Laufzeitumgebung besteht aus der Dalvik Virtual Machine (DVM) sowie den Core Libraries. Der bei der Anwendungsentwicklung erzeugte Java-Bytecode wird durch Cross-Compiling in Dalvik-Bytecode konvertiert. Dadurch werden die Möglichkeiten moderner Mikroprozessoren besser ausgenutzt. Außerdem umgeht man Lizenzgebühren, die durch die Verwendung der JVM und des Java-Bytecodes anfallen würden. Für jede Android-Anwendung wird ein eigener Betriebssystemprozess mit einer Instanz der DVM gestartet. Der Mehrbedarf an Ressourcen wird dabei durch erhöhte Sicherheit und Verfügbarkeit aufgewogen.

Bibliotheken Die Bibliotheken stellen dem Anwendungsrahmen alle zum Ausführen von Android-Anwendungen benötigten Funktionalitäten bereit. Es handelt sich hier um C/C++-Bibliotheken, die über Java-Schnittstellen angesprochen werden können. Sie bestehen zum Teil aus bekannten Open-Source-Produkten. Andere wurden speziell für Android entwickelt.

Anwendungsrahmen Der Anwendungsrahmen ist komplett in Java geschrieben. Er bildet den Unterbau für die Anwendungen und abstrahiert die zugrunde liegende Hardware durch zahlreiche Manager-Klassen. Neben den meisten Klassen und Methoden aus Java SE enthält er zahlreiche android-spezifische Klassen.

Anwendungsschicht In dieser Schicht befinden sich die Android-Anwendungen. Dies können sowohl Eigenentwicklungen als auch die mitgelieferten Standardanwendungen sein. Hier findet die Interaktion zwischen Mensch und Maschine sowie die Kommunikation zwischen Anwendungen statt.

2.3.4 Die Komponenten einer Android-Anwendung

Bei Android handelt es sich um eine Plattform, die komponentenbasierte Anwendungen ermöglichen soll. So muss bei der Softwareentwicklung das Rad nicht jedes mal neu erfunden werden. Eigene Anwendungen können z. B. Teile der Standardanwendungen wiederverwenden. Anwendungen können über ein Berechtigungssystem anderen Anwendungen erlauben, einige ihrer Komponenten zu verwenden. Es wird zwischen den vier folgenden Komponenten unterschieden:

Activity Anwendungen, die mit dem Benutzer interagieren, benötigen mindestens eine Activity. Activities dienen der Darstellung und der Verwaltung von Oberflächen. Sie lesen die Benutzereingaben aus und können miteinander zu komplexeren Anwendungen verknüpft werden.

Service Ein Service führt Hintergrundprozesse aus, die keine Benutzeroberfläche benötigen. Dies kann z. B. die Wiedergabe von Musik sein.

Content Provider Ein Content Provider verwaltet Daten, die von Anwendungen geladen oder gespeichert werden. Er abstrahiert die darunterliegende Persistenzschicht. Über Berechtigungen kann er anderen Anwendungen Daten zur Verfügung stellen. Es findet eine lose Kopplung über eine definierte Schnittstelle statt.

Broadcast Receiver Der Broadcast Receiver empfängt Systemnachrichten, um Anwendungen in die Lage zu versetzen, auf Änderungen des Systemzustandes zu reagieren. Dies kann z. B. ein schwacher Ladezustand des Akkus oder eine schlechte Netzwerkverbindung sein.

Kapitel 3

Die Anwendung zur Bearbeitung der Skizze

Dieses Kapitel befasst sich mit der Anwendung, die eine skizzierte Landkarte transformiert, um die Skizze der Wirklichkeit anzunähern. Die Anwendung soll eine skizzierte Landkarte, die Fehler in der Ausrichtung und Größenproportion der Geoobjekte aufweist, in eine Form überführen, die diese Fehler behebt bzw. der Realität annähert. D. h. die Skizze soll in eine Form überführt werden, in der sie über eine exakte Landkarte gelegt werden kann. Dabei sollen die Abweichungen in Form und Lage der Geoobjekte der Skizze von denen der exakten Karte möglichst gering sein. Es soll erreicht werden, dass für die Punkte der Skizze, die in der Realität die geografischen Koordinaten xy haben, nach der Transformation beim Auflegen auf die exakte Karte diese Punkte auch an bzw. in der Nähe der Position xy der exakten Karte erscheinen.

3.1 Fachliches Konzept

Zunächst muss die handschriftlich erstellte Skizze in einem Format vorliegen, das von der Anwendung geöffnet und weiterverarbeitet werden kann. Es sind zwei Wege denkbar:

1. Das Erstellen der Skizze auf Papier. Anschließend müsste die Skizze eingescannt und in einem Grafikformat abgespeichert werden, das die Anwendung darstellen und transformieren kann.

2. Das Erstellen der Skizze am Computer vom Benutzer unter Anwendung eines Grafikprogramms und dann wiederum Speicherung in einem Format, das die Weiterbearbeitung durch die Anwendung ermöglicht.

Für die erste Variante spricht, dass für viele Menschen das Erstellen einer Skizze mittels Stift und Papier intuitiver ist, als mit einem Grafikprogramm, das vorher erlernt werden muss. Bei der zweiten Variante entfällt dafür der Einsatz eines Scanners.

Nachdem die Skizze dann von der Transformationsanwendung geöffnet wurde, soll deren Annäherung an die Realität erreicht werden, indem Punkte der Skizze als Passpunkte festgelegt werden. Für diese Passpunkte sind neben den Koordinaten auf der Skizze auch ihre geografischen Koordinaten in der Realität anzugeben. Die Passpunkte sollten an markanten Punkten in der Skizze gesetzt werden, deren geografischen Koordinaten möglichst eindeutig zu ermitteln sind. Dies sind z. B. Wegkreuzungen, Häuser, Flussmündungen etc. Die Auswahl der Passpunkte kann auf unterschiedlichen Wegen geschehen:

1. Die Passpunkte könnten mit Hilfe entsprechender Algorithmen von der Anwendung vorgeschlagen werden.
2. Die Passpunkte könnten vom Benutzer mittels Mausklick frei gewählt werden.

Die erste Möglichkeit würde den Benutzer entlasten, da die Auswahl von geeigneten Passpunkten automatisch erfolgt. Verfügt der Benutzer jedoch im voraus von speziellen Punkten auf seiner Skizze über deren geografische Koordinaten, wäre die zweite Möglichkeit benutzerfreundlicher, da diese Punkte dann als Passpunkte definiert werden können. Auch lässt die zweite Möglichkeit ein experimentielleres Vorgehen zu, da man verschiedene Passpunktkombinationen ausprobieren kann.

Um die geografischen Koordinaten der Passpunkte zu erhalten, gibt es wiederum mehrere Möglichkeiten:

1. Der Benutzer ermittelt diese zuvor mittels GPS-Gerät oder mit Hilfe eines Kartendienstes und gibt sie dann in entsprechende Felder in der Anwendung ein.
2. Ein Kartendienst ist in die Anwendung integriert, auf dem man den jeweiligen Passpunkt durch Mausklick auswählt und so seine geografischen Koordinaten der Anwendung übermittelt.

Hier ist der zweite Punkt für den Benutzer komfortabler. Er muss nicht selbst Koordinaten ermitteln und dann eingeben, sondern kann die Passpunkte auf dem Kartendienst anwählen. Vorausgesetzt ist natürlich, dass er die Passpunkte auf dem Kartendienst einwandfrei identifizieren kann.

Bei Betätigung der Schaltfläche zum Auslösen der Transformation könnte dann folgendes ablaufen:

1. Die Anwendung wählt automatisch eine Transformationsart aus, abhängig von der Anzahl der spezifizierten Passpunkte.

2. Der Anwender wählt die gewünschte Transformationsart aus. Sind dann mehr Passpunkte spezifiziert, als für die Transformationsart benötigt, könnte eine Ausgleichsrechnung gemäß Abschnitt 2.2.4 ausgeführt werden, um das Ergebnis zu optimieren.

Die zweite Herangehensweise überlässt dem Anwender mehr Freiheit und Möglichkeit zum Experimentieren.

Die Anwendung führt die Transformation durch, indem mit Hilfe der Passpunte die Transformationsparameter wie Drehwinkel und Maßstabsfaktor gemäß den in Abschnitt 2.2 aufgeführten Verfahren zur Koordinatentransformation errechnet werden. Abschließend wird dann die gesamte Skizze mit den errechneten Transformationsparametern transformiert. Dies ist dann z. B. eine Drehung der Skizze um den Drehwinkel α und die Multiplikation der Koordinaten der Skizze mit dem Maßstabsfaktor m. Die Ausführung der Translation ist für die Darstellung auf dem Bildschirm nicht nötig, da sie das Ergebnis nicht für den Benutzer erkennbar ändert.

Nach erfolgter Transformation soll für die Skizze ein Maßstab eingeblendet werden, wenn der Benutzer dies möchte. Dafür ist es notwendig, die Anwendung über die Bildschirmgröße zu informieren. Die Anwendung kann dann die dargestellte Zeichnung, die in Pixel dimensioniert ist, mit deren Darstellungsgröße auf dem Bildschirm in Relation setzen. Der Benutzer sollte den dargestellten Bereich der Skizze mittels Bildlaufleisten verschieben können und die Skizze vergrößern bzw. verkleinern können, wobei sich dann die Maßstabsangabe anpassen sollte.

Ist der Benutzer mit dem Ergebnis der Transformation nicht zufrieden, sollte er die Möglichkeit haben, eine erneute Transformation mit geänderten Passpunkten und/oder einer anderen Transformationsart durchzuführen. Nach erfolgreicher Transformation soll der Benutzer die Möglichkeit haben, das Resultat abzuspeichern. Das Ergebniss sollte dann in einem Format vorliegen, dass in der Anwendung, die in Kapitel 4 behandelt wird, weiterverarbeitet werden kann.

3.1.1 Anwendungsfall

Aus obigen Überlegungen ergibt sich folgender Anwendungsfall:

Ziel: Transformation einer handschriftlich erstellten Landkarte in eine der Realität angenäherten Skizze.

Vorbed.: Die Skizze liegt in Form einer Grafikdatei vor.

Nachbed.: Die Skizze ist der Realität angenähert.

Erfolgsszenario:

1. Der Anwender öffnet eine Skizze, die daraufhin dargestellt wird.
2. Er hat die Option, die Bildschirmbreite einzugeben, um den Maßstab nach der Transformation zu erhalten.
3. Durch Mausklick wählt er Passpunkte auf der Skizze aus.
4. Für die ausgewählten Passpunkte kann er die geografische Länge und Breite angeben.
5. Durch die Betätigung einer Schaltfläche löst er die Transformation aus.
6. Die transformierte Skizze wird angezeigt und optional mit einer Maßstabsangabe versehen.
7. Der Benutzer kann die Grafik verschieben und vergrößern bzw. verkleinern.
8. Er kann die transformierte Grafik abspeichern.

Fehlerfall: Für die Transformation wurden nicht genügend Angaben in Form von Referenzpunkten gemacht.

Die Durchführung einer Transformation wird durch die Abbildung 3.1 auf der nächsten Seite visualisiert.

3.1.2 Funktionale Anforderungen

Aus dem aufgeführten Anwendungsfall ergeben sich folgende funktionale Anforderungen:

FA-0.1 Die Skizze soll ausgewählt werden können und dann dargestellt werden.

FA-0.2 Optional soll die Bildschirmbreite eingegeben werden können, um einen Maßstab für die Skizze bestimmen zu können.

FA-0.3 Mittels Mausklick sollen Punkte der Skizze ausgewählt werden, dabei werden die x- und y-Koordinaten des jeweiligen Punktes erfasst.

FA-0.4 Für die ausgewählten Punkte soll die geografische Länge und Breite angegeben werden können.

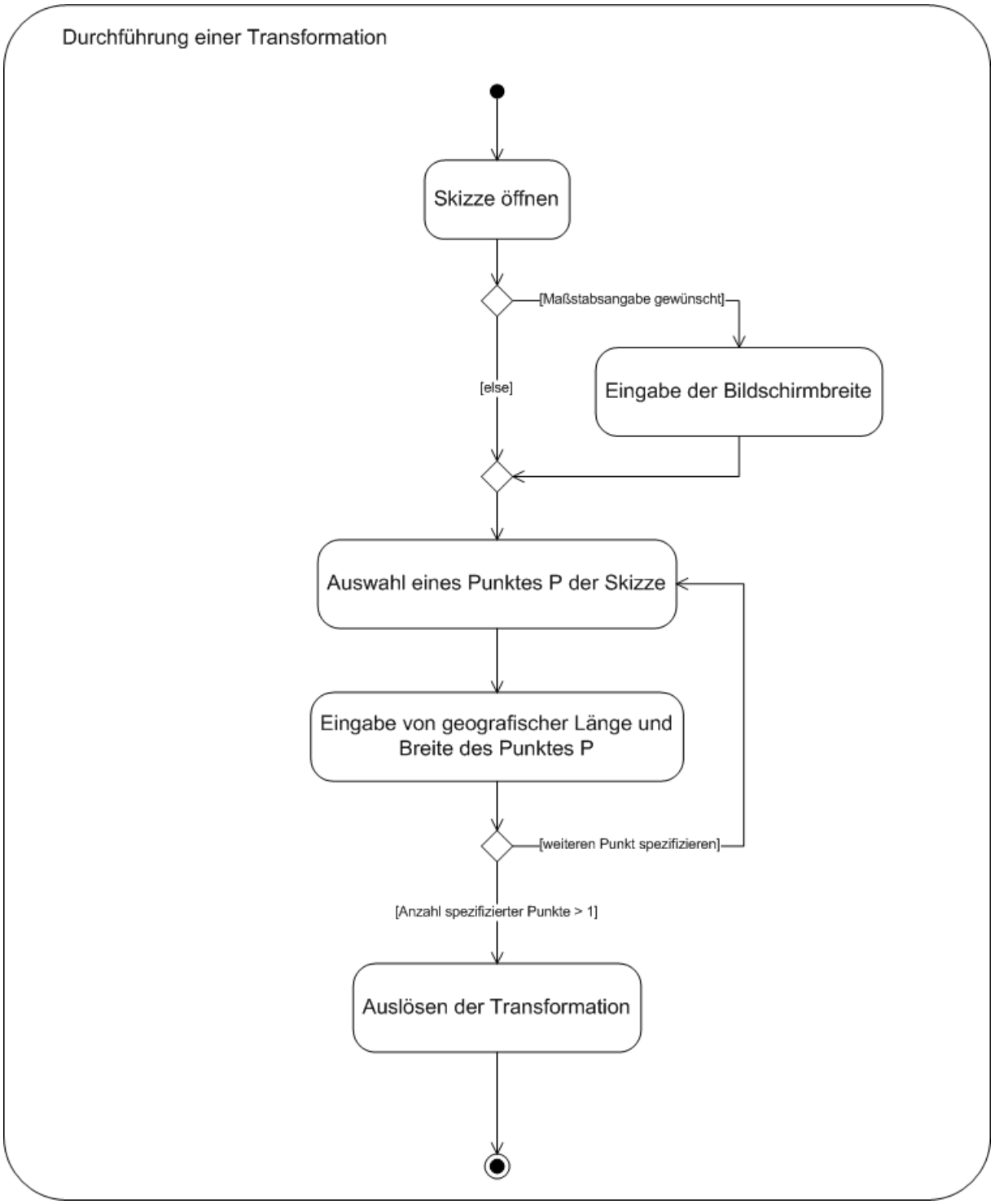

Abbildung 3.1: Aktivitätsdiagramm in UML2-Notation

FA-0.5 Mittels Schaltfläche soll die Transformation ausgelöst werden. Je nachdem wie viele Passpunkte bestimmt wurden, soll eine entsprechende Transformationsart gewählt werden. Die Transformation soll berechnet, und die resultierende Grafik dargestellt werden. Ein alternatives Vorgehen wäre die Festlegung der gewünschten Transformationsart. Sollten mehr Passpunkte, als für die gewählte Transformationsart benötigt, festgelegt worden sein, wird eine Ausgleichsrechnung gemäß Abschnitt 2.2.4 ausgeführt, um das Ergebnis zu optimieren.

FA-0.6 Bildlaufleisten sollen es ermöglichen, den dargestellten Bereich der Skizze zu verschieben.

FA-0.7 Es soll die Möglichkeit geben, die Skizze zu vergrößern bzw. zu verkleinern. Wünschenswert wäre dabei, die Anpassung der Maßstabsangabe.

FA-0.8 Die transformierte Skizze soll abgespeichert werden können.

Zur FA-5 ist der Fehler möglich, dass weniger als zwei bzw. für die gewünschte Transformationsart zu wenige Passpunkte festgelegt wurden. Dieser Fehler soll gemeldet werden, so dass die Informationen vom Anwender ergänzt werden können und dann die Transformation ausgeführt werden kann.

3.1.3 Nichtfunktionale Anforderungen

Folgende nichtfunktionalen Anforderungen sollen umgesetzt werden:

NFA-0.1 Benutzbarkeit - Die Anwendung sollte leicht erlernbar und intuitiv zu benutzen sein.

NFA-0.2 Robustheit - Falsche Eingaben sollten nicht zum Absturz der Anwendung führen.

NFA-0.3 Gute Performanz - Zwischen Auslösen der Transformation und Anzeige des Ergebnisses sollte keine fühlbare Zeit verstreichen.

NFA-0.4 Verfügbarkeit - Die Anwendung sollte jederzeit ausführbar sein.

NFA-0.5 Plattform - Als Plattform soll der Anwendung ein PC mit dem Betriebssystem Windows 7 dienen.

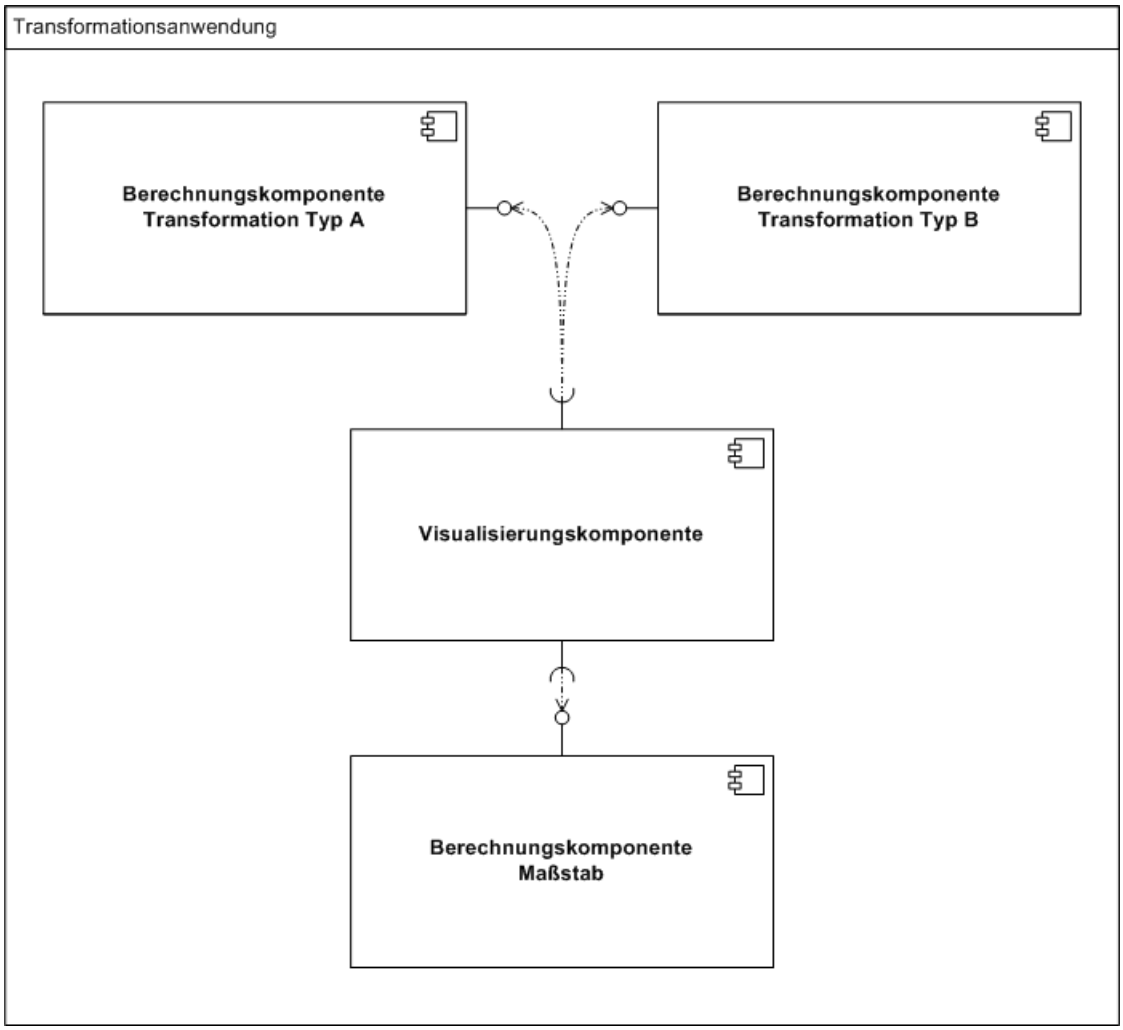

Abbildung 3.2: Fachliche Komponenten in UML2-Notation

3.1.4 Fachliche Architektur

Die fachliche Architektur ist in der Abbildung 3.2 dargestellt. Die Anzeige der Ausgangsgrafik bzw. der transformierten Grafik erfolgt durch die Visualisierungskomponente. In dieser Komponente erfolgen auch die Eingaben durch den Benutzer. Wird die Transformation ausgelöst, wird die erforderliche Berechnung der Transformationsparameter an eine Berechnungskomponente für die gewählte Transformationsart delegiert. In der Abbildung sind zwei Berechnungskomponenten für die Transformation dargestellt. Deren Anzahl richtet sich nach der Anzahl der implementierten Transformationsarten. Die Berechnung des Maßstabes wird an eine entsprechende Berechnungskomponente delegiert.

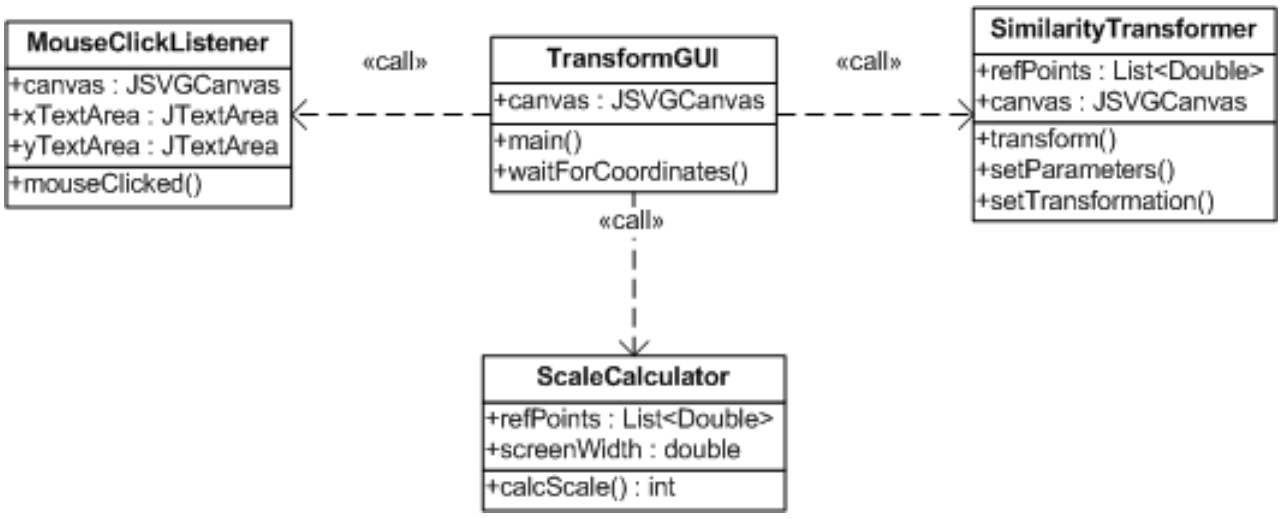

Abbildung 3.3: Klassendiagramm in UML2-Notation

3.2 Technisches Konzept

Die Anwendung zur Bearbeitung der skizzierten Landkarte soll auf der Java-SE-Plattform entwickelt werden. Dabei erfolgt die Darstellung der Skizze im SVG-Format. SVG steht für Scalable Vector Graphics und wurde vom World Wide Web Consortium spezifiziert (siehe World Wide Web Consortium 2011). Dieses Format wurde gewählt, da Vektorgrafiken gegenüber Rastergrafiken mit weniger Rechenaufwand transformiert werden können.

Um SVG-Grafiken mit Java darzustellen, bedarf es externen Bibliotheken. Diese Bibliotheken werden z. B. vom Projekt SVG Salamander (siehe Java.net 2011) und vom batik Java SVG Toolkit (siehe The Apache XML Graphics Project 2010) angeboten. Die Wahl fiel auf das batik Java SVG Toolkit, da es umfangreicher ist und so dem Programmierer mehr Möglichkeiten bietet.

Abbildung 3.3 zeigt das Klassendiagramm der Anwendung zur Transformation, in der die Ähnlichkeitsabbildung (engl.: similarity transformation) realisiert wurde. Die Main-Methode der Anwendung befindet sich in der Klasse `TransformGUI`. Durch ihre Ausführung öffnet sich eine grafische Oberfläche. Sie enthält zum einen eine Leiste mit Elementen, mit denen der Benutzer die nötigen Eingaben tätigen kann, zum anderen eine Fläche, auf der die SVG-Grafik dargestellt wird. Diese Fläche ist ein Swing-Element namens `JSVGCanvas`, das vom Batik-Toolkit bereitgestellt wird.

Nachdem der Benutzer eine Skizze ausgewählt hat, wird diese angezeigt. Er kann jetzt Passpunkte setzen, indem er entsprechende Schaltflächen betätigt. Daraufhin führt die Klasse `TransformGUI` ihre Methode `waitForCoordinates()` aus. Es wird ein `MouseClickListener` instanziiert. Ihm werden das `JSVGCanvas` und die Textfelder

für die ermittelten Bildschirmkoordinaten übergeben. Klickt der Benutzer nun einen Punkt auf der Skizze an, führt der `MouseClickListener` seine Methode `mouseClicked()` aus. Die Koordinaten des angeklickten Punktes erscheinen in den entsprechenden Textfeldern der GUI und der `MouseClickListener` wird beendet. Der Benutzer kann für diesen Punkt nun seine geografischen Koordinaten eingeben. Der ganze Vorgang wird wiederholt, bis genügend Passpunkte spezifiziert wurden.

Bei Betätigung der Schaltfläche zum Transformieren wird eine Instanz der Klasse `SimilarityTransformer` angelegt. Ihr werden die Bildschirmkoordinaten der Referenzpunkte und ihre geografischen Koordinaten mittels der Liste `refPoints` übergeben. Darüberhinaus erhält sie eine Referenz auf das `JSVGCanvas`. Die `TransformGUI` ruft die Methode `transform()` der Klasse `SimilarityTransformer` auf. Daraufhin errechnet diese mit der Methode `setParameters()` die Parameter für die Transformation. Dann transformiert sie die Landkarte auf dem `JSVGCanvas` durch die Methode `setTransformation()`.

Hat der Benutzer die Bildschirmbreite in der GUI angegeben, erfolgt durch das Auslösen der Schaltfläche zum Transformieren darüberhinaus eine Instanziierung der Klasse `ScaleCalculator`. Ihr wird die Liste `refPoints` und die Bildschirmbreite übergeben. Die `TransformGUI` ruft die Methode `calcScale()` der Klasse `ScaleCalculator` auf. Diese berechnet die Kennzahl, die angibt, wie vielen Maßeinheiten in der Realität eine Maßeinheit auf der Karte entspricht, woraufhin in der GUI eine Maßstabsangabe erscheint.

3.3 Implementierung

Die Implementierung erfolgte gemäß dem Klassendiagramm in Abbildung 3.3 auf der vorherigen Seite. Es wurden jedoch zwei Transformationsarten realisiert - Ähnlichkeitstransformation und Affintransformation. So gibt es neben der Klasse `SimilarityTransformer` noch die Klasse `AffineTransformer`, die bezüglich ihrer Datenelemente und ihres Methodenprotokolls analog aufgebaut ist. Die beiden Transformationsklassen sowie die Klasse `ScaleCalculator` bekommen die Liste `refPoints` im Gegensatz zur Abbildung 3.3 lediglich im Konstruktor übergeben. Ihre Elemente werden dann als Datenelemente angelegt.

Die Grundlage der Klasse `TransformGUI` wurde der Site über das batik Java SVG Toolkit (siehe The Apache XML Graphics Project 2010) entnommen. Hier findet sich ein Beispiel zum Öffnen und Anzeigen von SVG-Grafiken (http://xmlgraphics.apache.org/batik/using/swing.html). Dieses Beispiel wurde übernommen und dann modifiziert und erweitert. Dabei kam das Eclipse-Plug-In Visual Editor zum Einsatz.

Die Implementierung lässt den Benutzer die Passpunkte frei wählen. Die geografischen Koordinaten müssen zuvor ermittelt werden und können dann in entsprechende Felder eingetragen werden. Diese Eingabe hat im Format *Gradzahl - Dezimalpunkt - Nachkommastellen* zu erfolgen. Beim Auslösen der Schaltfläche zum Transformieren hängt die Wahl der Transformationsart davon ab, ob der Benutzer zwei oder drei Passpunkte spezifiziert hat. Wurden zwei Passpunkte spezifiziert, erfolgt eine Ähnlichkeitstransformation. Bei drei spezifizierten Passpunkten erfolgt eine Affintransformation. Eine Ausgleichsrechnung gemäß Abschnitt 2.2.4 wurde nicht implementiert. Die Skizze, die transformiert wird, muss einen Bereich abdecken, der nördlich des Äquators und östlich des Nullmeridians liegt, da keine Fallunterscheidung hinsichtlich des Quadranten implementiert wurde.

Werden in den Transformationsklassen die Methoden `setParameters()` aufgerufen, kommen zur Bestimmung der einzelnen Transformationsparameter Unterfunktionen zum Einsatz, deren Algorithmen mit dem Programm MATLAB 7.12 berechnet wurden.

Bei der Ähnlichkeitstransformation findet eine Rotation statt, wodurch die Skizze eingenordet werden soll. Die Translation wird nicht durchgeführt, da es für die Darstellung des Ergebnisses nicht wichtig ist, ob es in x- bzw. y-Richtung verschoben wird oder nicht. Die Koordinaten werden zwar mit dem Maßstabsfaktor multipliziert, anschließend aber nochmal mit dem Kehrwert seines Absolutwertes multipliziert. Er ist für die Darstellung auf dem Bildschirn nur relevant, falls er negativ ausfällt, also eine Spiegelung der Skizze hervorruft. Diese Spiegelung führt im Zusammenspiel mit der Rotation zu einem korrekten Ergebnis.

Bei der Affintransformation ist das Vorgehen identisch. Jedoch werden hier unterschiedliche Maßstabsfaktoren für die x- und für die y-Richtung eingesetzt. Darüberhinaus findet noch eine Scherung der Skizze statt.

Für die Implementierung der Maßstabsangabe wurde das Buch Bronstein u. a. (2008) auf S. 180 zu Rate gezogen. Die Maßstabsangabe wird im Fall einer Ähnlichkeitstransformation korrekt angezeigt - im Fall einer Affintransformation wird die Änderung durch die Scherung und den unterschiedlichen Maßstabsfaktoren für die x- und y-Richtung nicht berücksichtigt. Die Bildlaufleisten wurden realisiert, die Zoomfunktion hingegen nicht.

Abschließend ist eine Speicherung der resultierenden Grafik im PNG-Format möglich. Das Abspeichern in diesem Format hat im Gegensatz zu anderen Formaten das Ergebnis am natürlichsten wiedergegeben. Es kann gut mit den Möglichkeiten, die in Kapitel 4 vorgestellt werden, weiterverarbeitet werden.

Abbildung 3.4 auf der nächsten Seite zeigt die realisierte Anwendung zur Bearbeitung der Skizze nach dem Ausführen einer Affintransformation.

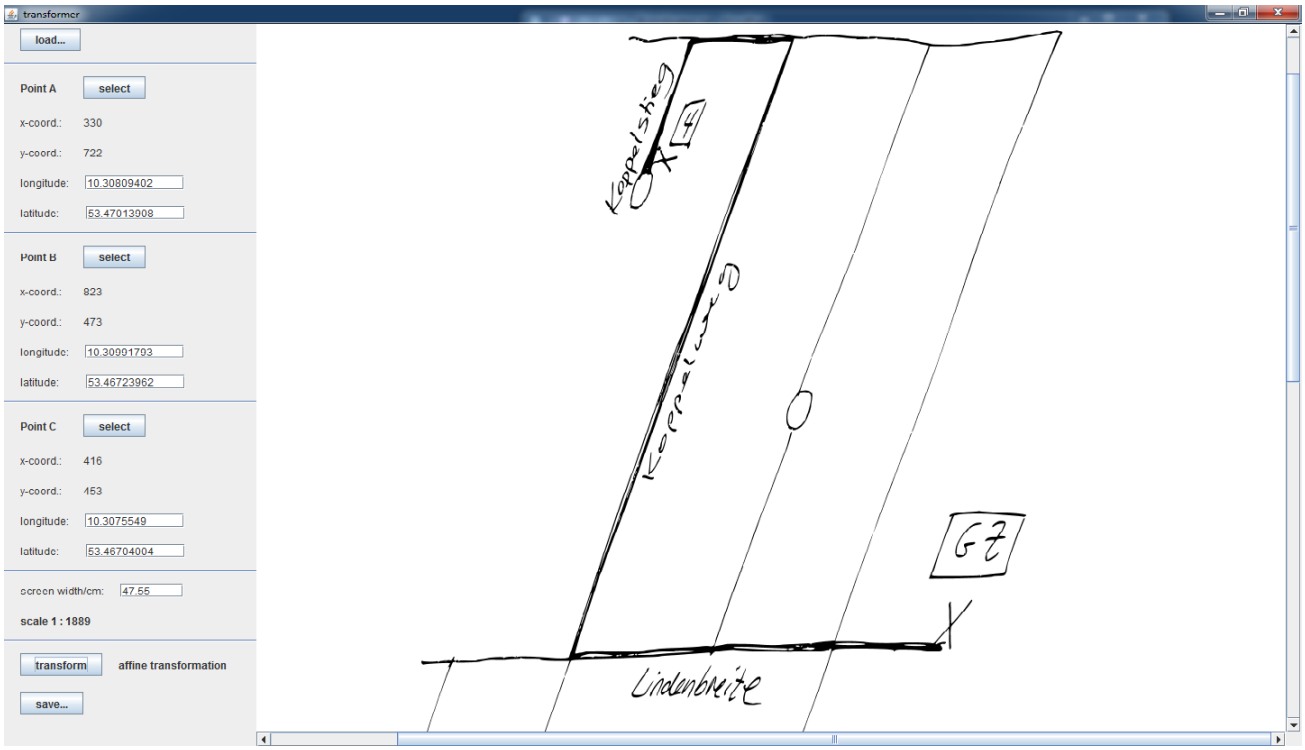

Abbildung 3.4: Die realisierte Anwendung zur Bearbeitung der Skizze

3.4 Test

Zunächst muss erwähnt werden, dass die Vereinigung von Entwickler und Tester in einer Person problematisch ist. Der Tester sollte dem Programm objektiv gegenüberstehen. Daher sollte man möglichst die Entwicklung eines Programms und das Testen dieses Programms von unterschiedlichen Personen ausführen lassen. Folgende Testarten gibt es:

Komponententest „Test einer einzelnen Softwareeinheit [...].” (Spillner und Linz 2005, S. 244)

Integrationstest „Test mit dem Ziel, Fehlerwirkungen in Schnittstellen und im Zusammenspiel zwischen integrierten Komponenten zu finden.” (Spillner und Linz 2005, S. 243)

Systemtest „Test eines integrierten Systems, um sicherzustellen, dass es spezifizierte Anforderungen erfüllt.” (Spillner und Linz 2005, S.252)

Abnahmetest „Formales Testen hinsichtlich der Benutzeranforderungen und -bedürfnisse bzw. der Geschäftsprozesse, das durchgeführt wird, um einen Auftraggeber oder eine bevollmächtigte Instanz in die Lage zu versetzen, entscheiden zu können, ob ein System anzunehmen ist oder nicht.” (Spillner und Linz 2005, S.235)

3.4.1 Konzipierte Tests

Die Anwendung zur Bearbeitung der Skizze könnte einem Komponententest unterzogen werden. Dabei könnte man die verschiedenen Klassen als Komponenten auffassen. Für jede Klasse würde man eine Testklasse schreiben, die jede Methode der zu testenden Klasse auf das erwartete Ergebnis hin überprüft. Hierfür bietet sich das Framework JUnit an.

Auch könnte man den Abnahmetest umsetzen, bei dem man handschriftliche Skizzen mit der Anwendung öffnet, Passpunkte setzt, die Transformation durchführt und das Ergebnis auf seine Plausibilität hin überprüft. Beim berechneten Maßstab könnte überprüft werden, ob die Größenverhältnisse, die auf dem Bildschirm wiedergegeben werden, beim Umrechnen mit dem Maßstab die Größenverhältnisse in der Realität wiederspiegeln.

3.4.2 Realisierte Tests

Der Komponententest wurde mit dem Framework JUnit umgesetzt. Dies jedoch nur für einige Methoden und nicht umfassend.

Der Abnahmetest wurde umgesetzt, indem eine handschriftliche Skizze als Ausgangspunkt genommen wurde. Diese Skizze wurde eingescannt und in vier Versionen als BMP-Datei abgespeichert:

- im Original
- um 90° nach links gekippt
- um 90° nach rechts gekippt
- um 180° gedreht

Mit dem Programm Potrace 1.9 (siehe Selinger 2011) wurden die Skizzen ins SVG-Format konvertiert und konnten somit von der Transformationsanwendung geöffnet werden.

Nun wurden die vier Skizzen einer Ähnlichkeitstransformation unterzogen. Jede der vier Transformationen wurde mit dem gleichen Passpunktepaar ausgeführt. D. h. die geografischen Koordinaten der Passpunkte waren fest und ihre Bildschirmkoordinaten bezogen sich auf die gleichen Bildelemente. Die Transformationen führten zu identischen Ergebnissen, bei denen die Skizze jeweils korrekt eingenordet wurde.

Dieses Vorgehen wurde auch mit der Affintransformation ausgeführt. Wieder wurden die Skizzen korrekt eingenordet. Ihre jeweilige Scherung und Verzerrung aufgrund der Maßstabsfaktoren fiel jedoch unterschiedlich aus.

Der Maßstab, der nach einer Ähnlichkeitstransformation ausgegeben wurde, wurde überprüft, indem der Abstand zweier Punkte auf dem Bilschirm mit einem Lineal erfasst wurde. Die Distanz zwischen diesen beiden Punkten in der Realität wurde mit dem Programm GoogleEarth abgefragt. Es stellte sich heraus, dass diese beiden Abstände durch den Maßstab ins richtige Verhältnis gesetzt wurden.

Kapitel 4

Die Anwendung zur Darstellung der Skizze unter Android

In Kapitel 3 wurde die Anwendung zur Bearbeitung einer handschriftlichen Skizze entwickelt. Als Ergebnis dieser Anwendung liegt nun die bearbeitete Skizze in Form einer Grafikdatei vor. Dieses Kapitel behandelt die Anwendung zur Darstellung dieser Grafikdatei unter Android. Dabei soll die Grafik über das Kartenbild eines Kartendienstes gelegt werden. Die skizzierten Geoobjekte sollen möglichst genau auf den geografischen Koordinaten des Kartendienstes abgebildet werden, die sie auch in der Realität haben.

4.1 Fachliches Konzept

Die in einen Kartendienst einzubettende Datei muss zunächst auf das Android-Endgerät übertragen werden. Die Anwendung auf dem Android-Gerät sollte mit einer Oberfläche starten, die vom Benutzer die für die Darstellung des Ergebnisses erforderlichen Angaben abfragt. Diese sind:

1. Welche Grafikdatei soll in den Kartendienst eingebettet werden?
2. An welchen geografischen Koordinaten auf der Weltkarte des Kartendienstes soll die Grafikdatei dargestellt werden?

Zu Punkt zwei sind mehrere Vorgehensweisen denkbar:

1. Angabe der geografischen Länge und Breite des Mittelpunktes der Grafikdatei sowie ihres Maßstabes. Für diese Variante wäre zu klären, wo der Mittelpunkt der Skizze liegt

und welche geografischen Koordinaten er hat. Außerdem muss der Maßstab des Kartenausschnitts, den der Kartendienst darstellt, mit dem Maßstab der Skizze in Übereinstimmung gebracht werden.

2. Angabe der beiden Längen- und der beiden Breitengrade, die die Skizze einrahmen. Bei dieser Vorgehensweise sind die vier Punkte der Skizze zu bestimmen, die am weitesten in Richtung der jeweiligen Himmelsrichtung liegen. So ergibt sich dann z. B. aus dem Breitengrad des nördlichsten Punktes der Skizze der Breitengrad, der die Skizze an ihrer nördlichen Grenze einrahmt. Für die drei anderen Begrenzungen ist dann analog zu verfahren.

3. Wiederverwendung der in der Anwendung aus Kapitel 3 gesetzten Passpunkte. Ihre Koordinaten könnten als Metadaten in der Grafikdatei hinterlegt sein oder, sollten keine Metadaten unterstützt werden, in einer zweiten Datei mitgeliefert werden, die dann evtl. anzugeben wäre. Diese Angabe könnte entfallen, wenn man durch entsprechende Maßnahmen dafür sorgt, dass diese automatisch geöffnet wird. Eine solche Maßnahme wäre dann, z. B. Grafikdatei und begleitende Datei bis auf die Dateiendung mit dem gleichen Namen zu versehen. Bei der Einbettung der Skizze in den Kartendienst müsste dann dafür gesorgt werden, dass die Passpunkte an den geografischen Koordinaten des Kartendienstes zu liegen kommen, die in der Anwendung aus Kapitel 3 gesetzt wurden. Dies ist für die Passpunkte, mit denen die Transformation durchgeführt wurde, möglich, ohne die Skizze verzerren zu müssen. Die Verzerrung ist bereits in der Anwendung zur Koordinatentransformation durchgeführt worden.

Variante eins benötigt also drei Benutzereingaben und Variante zwei vier Benutzereingaben. Wohingegen Variante drei höchstens eine Benutzereingabe benötigt. Aufgrund dieser Tatsache ist Variante drei am attraktivsten.

Abschließend soll der Benutzer nun eine Schaltfläche betätigen können, um die Enbettung der Skizze angezeigt zu bekommen. Beim Betrachten der Grafik wären folgende Eigenschaften wünschenswert:

1. Der Benutzer kann die angezeigte Karte in alle vier Himmelsrichtungen verschieben. Dabei bewegt sich die Karte des Kartendienstes im gleichen Maße, wie die darübergelegte Skizze.

2. Der Benutzer soll die Möglichkeit haben, in die Karte des Kartendienstes hinein oder aus ihr hinaus zu zoomen. Die aufgelegte Skizze soll sich der Zoombewegung anpassen, indem sie vergrößert oder verkleinert dargestellt wird.

4.1.1 Anwendungsfall

Es ergibt sich folgender Anwendungsfall:

Ziel: Anzeige eines Kartenbildes eines Kartendienstes mit eingebetteter handschriftlicher Skizze.

Vorbed.: Die Skizze liegt als Grafikdatei vor. Evtl. liegt noch eine zusätzliche Datei vor, die die Daten für die geografische Verortung der Skizze auf dem Kartendienst spezifiziert.

Nachbed.: Das Kartenbild eines Kartendienstes mit der darübergelegten handschriftlichen Skizze wird angezeigt.

Erfolgsszenario:

1. Der Anwender wählt eine Grafikdatei aus, die in den Kartendienst eingebettet werden soll.
2. Er bestimmt die Parameter, die die Grafikdatei geografisch auf der Karte des Kartendienstes verorten.
3. Durch Betätigung einer Schaltfläche wird die Karte des Kartendienstes, über die die Grafikdatei gelegt ist, angezeigt.
4. Der Benutzer kann die Karte verschieben und vergrößern bzw. verkleinern, wobei sich die aufgelegte Skizze entsprechend mitbewegt.

4.1.2 Funktionale Anforderungen

Aus obigem Anwendungsfall leiten sich folgende funktionale Anforderungen ab:

FA-1.1 Die einzubettende Grafikdatei lässt sich bestimmen.

FA-1.2 Es lassen sich Parameter zur geografischen Verortung der Skizze eingeben.

FA-1.3 Durch Betätigung einer Schaltfläche wird die resultierende Grafik des Kartendienstes mit der eingebetteten Grafikdatei angezeigt.

FA-1.4 Die resultierende Karte lässt sich verschieben, wobei sich die eingebettete Grafik mit verschiebt.

FA-1.5 Die resultierende Karte lässt sich vergrößern bzw. verkleinern, wobei die eingebettete Grafik sich auch vergrößert bzw. verkleinert.

Bei FA-2 ist ein Fehler möglich, wenn die Verortung der Skizze auf dem Kartendienst mittels einrahmender Längen- und Breitengrade realisiert wurde. Der Benutzer kann z. B. einen begrenzenden nördlichen Breitengrad bestimmen, der südlich des begrenzenden südlichen Breitengrades liegt.

4.1.3 Nichtfunktionale Anforderungen

Folgende nichtfunktionale Anforderungen sollen umgesetzt werden:

NFA-1.1 Benutzbarkeit - Die Anwendung sollte leicht erlernbar und intuitiv zu benutzen sein.

NFA-1.2 Robustheit - Falsche Eingaben sollten nicht zum Absturz der Anwendung führen.

NFA-1.3 Gute Performanz - Zwischen dem Zeitpunkt des Auslösens der Anzeige der Ergebnisgrafik und ihrer Abbildung auf dem Bildschirm sollte keine Verzögerung erfolgen.

NFA-1.4 Verfügbarkeit - Die Anwendung sollte jederzeit ausführbar sein.

NFA-1.5 Plattform - Als Plattform soll der Anwendung ein Android-Endgerät dienen.

4.1.4 Fachliche Architektur

In der Abbildung 4.1 auf der nächsten Seite ist die fachliche Architektur der Anwendung zur Einbettung der Skizze dargestellt. Die zur Darstellung erforderliche Eingabe der Parameter und die Ergebniskarte werden in getrennten Visualisierungskomponenten angezeigt, da der Bildschirm auf Android-Geräten klein ist und sich daher die Zusammenfassung in eine Komponente nicht anbietet.

Der Benutzer gibt die Parameter für die Einbettung der Skizze in eine Visualisierungskomponente ein. Von ihr übernimmt die Visualisierungskomponente für die Ergebniskarte die Parameter und delegiert die Berechnungen zur Einbettung der Skizze an eine Berechnungskomponente.

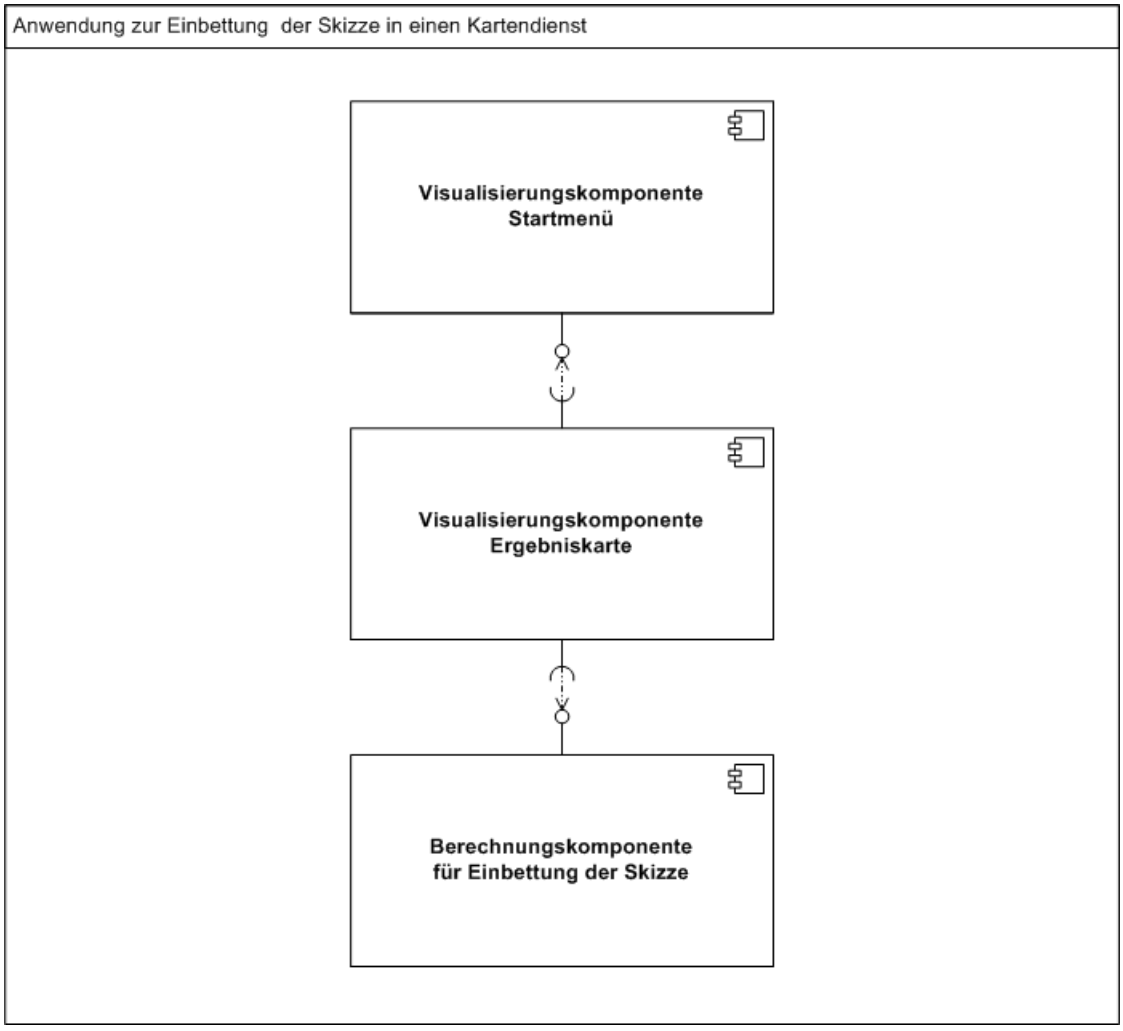

Abbildung 4.1: Fachliche Komponenten in UML2-Notation

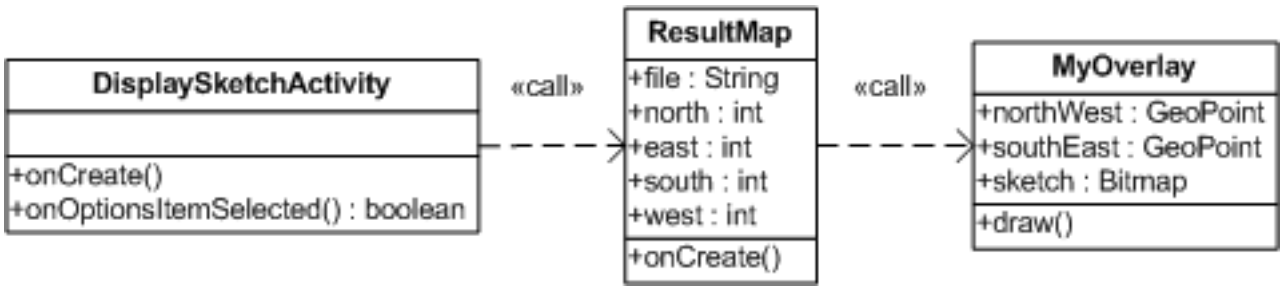

Abbildung 4.2: Klassendiagramm in UML2-Notation

4.2 Technisches Konzept

Bei der Entwicklung der Anwendung zur Darstellung der Skizze wird das Android-SDK eingesetzt. Dabei wird die Programmiersprache Java verwendet. Wie in Abschnitt 2.3 erwähnt, kommen die meisten Klassen und Methoden aus Java SE sowie zahlreiche android-spezifische Klassen zum Einsatz.

Als Kartendienst, in den die Skizze eingebettet werden soll, wurde Google Maps gewählt. Dadurch ist es nötig, die Google APIs in das Android-SDK einzubinden. Die Wahl fiel auf Google Maps, da durch die Tatsache, dass Google sowohl hinter Android als auch hinter Google Maps steht, eine reibungslose Integration des Kartendienstes zu erwarten ist.

In Abbildung 4.2 ist das Klassendiagramm der Anwendung zur Darstellung der Skizze aufgeführt. Die Anwendung startet mit der Methode `onCreate()` der Klasse `DisplaySketchActivity`. Es erscheint daraufhin eine GUI, die dem Benutzer ermöglicht, die Parameter zur Einbettung der Skizze einzugeben. Diese sind bei der hier gewählten Herangehensweise der Name der einzubettenden Datei sowie die beiden Längen- und Breitengrade, die die Position der Skizze auf der Karte des Kartendienstes einrahmen.

Sind die Informationen eingegeben, kann der Benutzer durch Betätigung der Menü-Taste des Android-Gerätes die Anzeige der resultierenden Grafik auswählen. Dadurch wird die Methode `onOptionsItemSelected()` ausgeführt, was zum Aufruf der Methode `onCreate()` der Klasse `ResultMap` führt. Diese Methode sorgt für die Anzeige eines Kartenausschnitts von Google Maps. Sie instanziiert ein Objekt der Klasse `MyOverlay`, das mit seiner Methode `draw()` die Skizze in das Kartenbild von Google Maps einzeichnet.

Es ist sicherzustellen, dass die Skizze zwischen den zuvor gewählten Längen- und Breitengraden eingezeichnet wird. Dies könnte folgendermaßen erreicht werden:

1. Ein Ansatz wäre, wie auf android developers 2011 vorgestellt, die Klasse `MyOverlay` von der Klasse `ItemizedOverlay` abzuleiten. Man kann hier geografische Koordinaten eines Punktes bestimmen, an dem das Overlay zentriert werden soll. Dieser

Punkt ist dann aus den übergebenen Längen- und Breitengraden zu berechnen. Jedoch ist das `ItemizedOverlay` nicht zoombar. Es erscheint in fester Größe auf der Karte, abhängig von seiner Höhe und Breite in Pixeln. Der Benutzer kann jetzt durch hinein- oder hinauszoomen eine Darstellung erreichen, in der der Maßstab der Skizze mit dem Maßstab der Karte des Kartendienstes annähernd übereinstimmt.

2. Ein anderer Ansatz wäre, die Positionierung der Skizze im Kartenbild zu erreichen, indem die Klasse `MyOverlay` ein Rechteck definiert, das von seiner nordwestlichen und südöstlichen Ecke in Form der vom Benutzer eingangs eingegebenen geografischen Koordinaten festgelegt wird. Bei jedem Aufruf der Methode `draw()` werden die Ecken des Rechtecks in Bildschirmkoordinaten umgerechnet und dann die Skizze in dieses Rechteck gezeichnet. Auf diese Weise ist ein Verschieben und Zoomen der Karte möglich, wobei sich die eingebettete Skizze entsprechend mitbewegt.

Durch Betätigung der Zurück-Taste des Android-Gerätes gelangt der Benutzer wieder in das Menü der Klasse `DisplaySketchActivity` und kann die Parameter verändern.

4.3 Implementierung

Die Implementierung der Anwendung zur Darstellung der Skizze entspricht dem Klassendiagramm in Abbildung 4.2 auf der vorherigen Seite. Die Klasse `MyOverlay` wurde jedoch bei der Realisierung in `ZoomableOverlay` umbenannt. Folglich wird die Positionierung der Skizze auf der Karte des Kartendienstes gemäß der zweiten Variante aus Abschnitt 4.2 erreicht, bei der ein Rechteck durch die geografischen Koordinaten seiner Eckpunkte definiert wird, das dann von der Skizze ausgefüllt wird.

Beim Starten der Anwendung wird die Klasse `DisplaySketchActivity` instanziiert. Der Benutzer gibt die Parameter zur Anzeige der Ergebnisgrafik ein. Für die Wahl der geografischen Position der Skizze auf dem Kartenbild des Kartendienstes wurde die Variante gewählt, bei der die beiden Längen- und Breitengrade eingegeben werden, die die Skizze einrahmen. Ihre Eingabe hat im Format *Gradzahl - Dezimalpunkt - Nachkommastellen* zu erfolgen. Breitengraden, die südlich des Äquators liegen, ist ein negatives Vorzeichen voranzustellen. Das Gleiche gilt für Längengrade, die westlich des Nullmeridians liegen.

Wird die Anzeige der resultierenden Grafik ausgelöst, wird ein Intent erzeugt. Ein Intent ist ein konkreter Aufruf einer anderen Activity. Er wird mit den Übergabeparametern versehen. Durch seine anschließende Ausführung wird die Anzeige der Oberfläche der Klasse `ResultMap` ausgelöst, die von der Klasse `MapActivity` abgeleitet ist. Diese Klasse wird von den Google APIs bereitgestellt und realisiert die Anzeige von Google-Maps-Kartenmaterial. Wie beim Kartendienst Google Maps kann man durch Zoomen und Verschieben des Kartenausschnitts navigieren.

Neben den Java-Sourcecode-Dateien, die das Klassendiagramm von Abbildung 4.2 umsetzen, wurden folgende XML-Dateien editiert:

- main.xml / result_map.xml - Diese Dateien befinden sich unter dem Pfad *res/layout* und definieren die grafischen Oberflächen der Klassen `DisplaySketchActivity` bzw. `ResultMap`.
- strings.xml - In dieser Datei erfolgen Definitionen von Texten. Zu finden ist diese Datei unter *res/values*.
- AndroidManifest.xml - Durch diese Datei werden Metadaten der Anwendung definiert. Sie liegt im Wurzelverzeichnis.

Beim Erstellen des Codes wurden drei Quellen als Beispiel genutzt und zum Teil Passagen aus ihnen übernommen:

- Struktur einer Android-Anwendung und Übergabe von Parametern zwischen Activities: Becker und Pant 2010, S. 3 - 17
- Einbindung von Google Maps: android developers 2011
- Implementierung der Methode `draw()` der Klasse `ZoomableOverlay`: Meier 2010, S. 269 und 501

Auf der SD-Karte des Android-Gerätes ist auf oberster Ebene das Verzeichnis *mySketches* anzulegen. In diesem Verzeichnis können dann die Skizzen gespeichert werden, die in den Kartendienst eingebettet werden sollen. Sie können in den gängigen Grafikformaten wie *JPG*, *BMP*, oder *PNG* vorliegen. Es empfiehlt sich jedoch die Verwendung des PNG-Formates, da so die Transparenz des Hintergrundes der Skizze erreicht werden kann.

Abbildung 4.3 zeigt die umgesetzte Anwendung zur Darstellung der Skizze im Android-Emulator.

4.4 Test

In diesem Abschnitt sollen Tests für die Anwendung zur Darstellung der Skizze konzipiert und deren Realisierung dargestellt werden. Die verschiedenen Testarten wurden bereits in Abschnitt 3.4 vorgestellt.

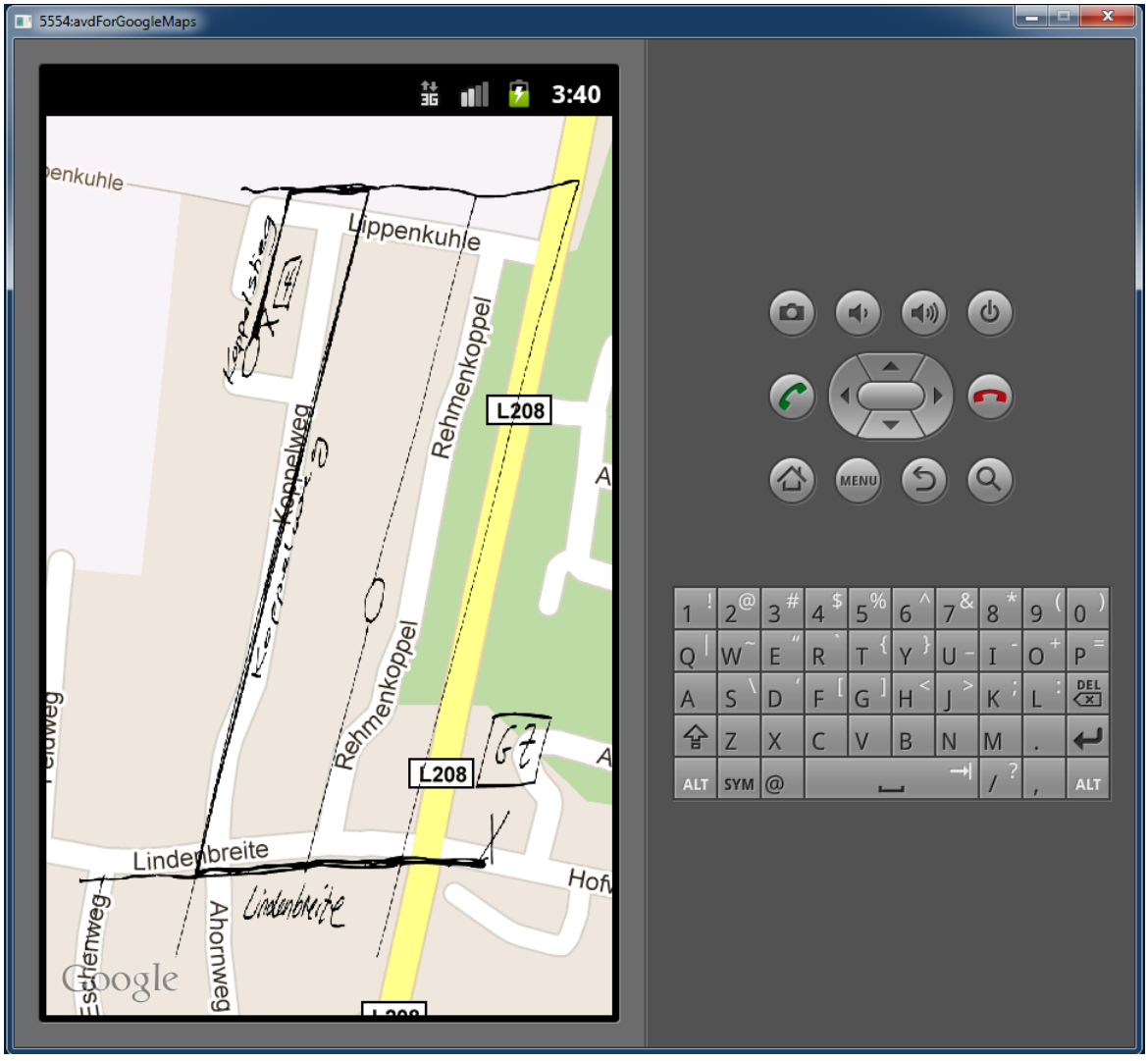

Abbildung 4.3: Die realisierte Anwendung zur Darstellung der Skizze auf dem Android-Emulator

4.4.1 Konzipierte Tests

Wie die Anwendung zur Transformation der Skizze könnte auch diese Anwendung einem Komponententest unterzogen werden, bei dem jede Klasse untersucht wird. Auch hier bietet sich das Framework JUnit an, da es in das Android-SDK integriert ist.

Wiederum könnte ebenfalls der Abnahmetest umgesetzt werden, indem überprüft wird, ob eine Grafikdatei an dem Ort in den Kartendienst eingebettet wird, der zuvor durch die Eingabe der begrenzenden Längen- und Breitengrade spezifiziert wurde. Es sollten alle Kombinationen von Vorzeichen der Längen- und Breitengrade getestet werden.

4.4.2 Realisierte Tests

Während der Komponententest nicht realisiert wurde, wurde der Abnahmetest umgesetzt. Er wurde sowohl mit dem Android-Emulator des Android-SDKs sowie mit dem Smartphone *HTC Desire* ausgeführt. Dabei wurde eine Skizze in Form von zwei Linien, die im rechten Winkel aufeinandertreffen, verwendet. Die eingrenzenden Längen- und Breitengrade wurden so gewählt, dass die Skizze jeweils auf eine Straßenkreuzung projiziert wird. Bei der Anzeige des Ergebnisses wurde kontrolliert, ob die Grafikdatei über die richtige Straßenkreuzung gelegt wurde und ob die beiden Linien durch die zuvor eingegebenen Längen- und Breitengrade begrenzt werden. Abbildung 4.4 zeigt die Ergebnisgrafik einen solchen Tests.

Um alle Vorzeichenkombinationen der Längen- und Breitengrade abzudecken, wurde jeweils eine Kreuzung in Hamburg, Sydney, Rio de Janeiro und New York ausgewählt. Außerdem wurde Greenwich gewählt, um die Kombination von zwei Längengraden mit unterschiedlichen Vorzeichen zu testen. Eine Straße nördlich von Quito wurde gewählt, um die Kombination von zwei Breitengraden mit unterschiedlichen Vorzeichen zu testen. Alle Tests führten zu einer Abbildung, die durch die Wahl von Längen- und Breitengraden zu erwarten war.

Bei der Einbettung von größeren Grafikdateien kommt es zu Programmabstürzen mit der Meldung „Die Anwendung [...] wurde unerwartet beendet.”. Das Logging-System des Android-SDKs meldet einen Out-Of-Memory-Error. Dies lässt auf ein Speicherleck schließen. Bei Verwendung von Grafikdateien mit einer Größe von 54 KB tritt dieser Fehler erst nach längerer Zeit auf. Bei der Verwendung einer Grafikdatei von 225 KB Größe tritt der Fehler sehr rasch auf. Der Android-Emulator ist in dieser Hinsicht robuster als das Smartphone *HTC Desire*.

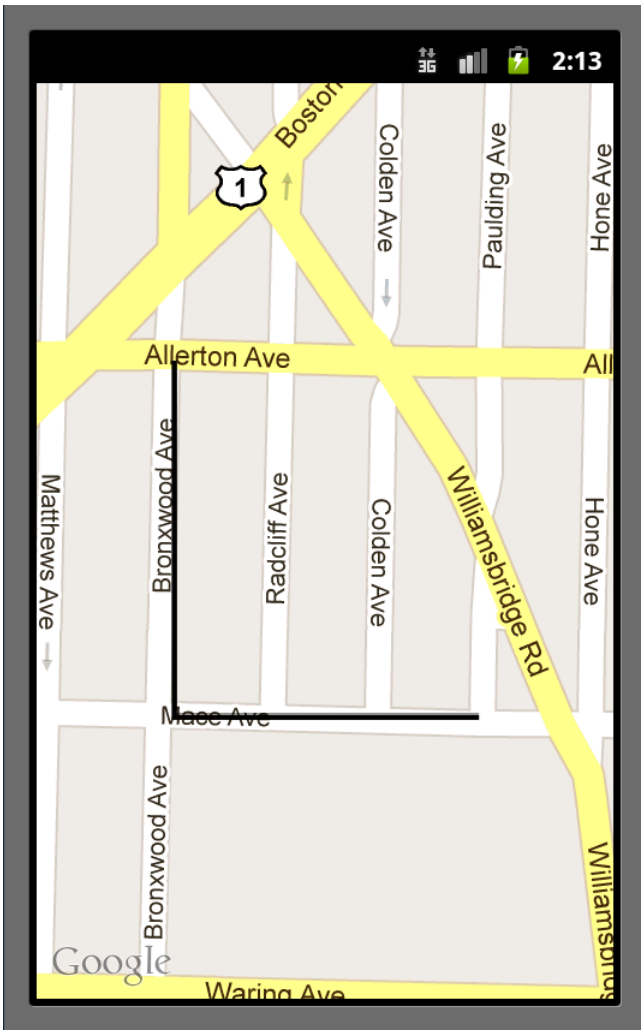

Abbildung 4.4: Straßenkreuzung in New York auf dem Android-Emulator

4.5 Darstellung der Skizze durch eine KML-Datei

In diesem Kapitel wurde eine Anwendung zur Darstellung der Skizze unter Android konzipiert und implementiert. Es gibt darüberhinaus einen alternativen Weg zur Einbettung der Skizze in einen Kartendienst: Die Verwendung der Keyhole Markup Language (KML), die vom Open Geospatial Consortium spezifiziert wurde (siehe Open Geospatial Consortium 2011).

Mit dieser Sprache lassen sich Geodaten definieren, die dann in Google Earth, Google Maps und anderen geografischen Anwendungen angezeigt werden können. In einer KML-Datei lassen sich einfache Elemente wie z. B. Ortsmarken definieren, aber auch Elemente wie ein Overlay. Solch ein Overlay kann dann die handschriftlich skizzierte Landkarte sein, die in der KML-Datei referenziert wird und in Form eines Vierecks gebildet aus Längen- und Breitengraden auf die Karte des Kartendienstes projiziert wird.

4.5.1 KML auf dem PC

Führt man auf dem PC das Programm Google Earth aus, lassen sich KML-Dateien problemlos anzeigen. Ein Overlay-Element, das eine skizzierte Landkarte repräsentiert, wird in das Satellitenbild von Google Earth integriert und passt seine Größe beim Zoomvorgang an.

Bei Google Maps funktioniert die Integration von KML-Dateien nicht ganz so problemlos. Hat man sich mit seinem Google-Nutzerkonto angemeldet, lässt sich auf Google Maps die Schaltfläche „Meine Orte” anwählen. Es lassen sich hier Karten erstellen, indem man die Karte von Google Maps mit z. B. Ortsmarken anreichert. Auch kann man hier eine KML-Datei importieren. Dies funktioniert zur Zeit leider nur begrenzt. Enthält die KML-Datei Ortsmarken funktioniert die Integration problemlos - die Ortsmarken werden angezeigt. Enthält die KML-Datei jedoch ein Overlay, wird dieses leider nicht wiedergeben. Interessant ist das Feature „Meine Orte” vor allem deswegen, weil die Karten, die man erstellt hat, von jedem Computer mit Internetzugang, auf dem man sich mit seinem Google-Nutzerkonto angemeldet hat, abrufbar sind - also auch von einem Android-System aus.

Ein anderer Ansatz, um eine KML-Datei in Google Maps anzuzeigen, ist das Hosten dieser Datei im Internet. Gibt man nun in das Suchfenster von Google Maps die URL dieser Datei ein, wird sie problemlos in Google Maps integriert. Auch das Anzeigen eines Overlays funktioniert auf diese Weise so einwandfrei wie unter Google Earth. Das Ergebnis lässt sich dann jedoch nicht unter „Meine Orte” festhalten.

4.5.2 KML unter Android

Google Earth gibt es auch als App für Android. Hier lässt sich dann jedoch, im Gegensatz zur PC-Version, keine KML-Datei öffnen.

Die Google-Maps-Anwendung für Android hat das Feature „Meine Karten". Hier kann man Karten öffnen, die, wie im Abschnitt 4.5.1 beschrieben, am PC unter „Meine Orte" angelegt wurden. Wie oben bereits erwähnt, funktioniert die Wiedergabe von Overlays hier momentan jedoch nicht. Gibt man in das Suchfenster der Google-Maps-Anwendung die URL einer gehosteten KML-Datei ein, was am PC zum Erfolg geführt hat, bekommt man jedoch mitgeteilt, dass keine Ergebnisse gefunden wurden - hier ist die Suchfunktion anscheinend anders implementiert als in der Google-Maps-Version im Internet. Dieser Versuch führt allerdings zum Erfolg, wenn man Google Maps im Internet-Browser des Android-Systems öffnet. Nun wird das Overlay auch auf diesem System angezeigt. Die Performanz auf einem Android-Smartphone ist allerdings nur mäßig - Zoomen und Verschieben der Karte sind nur beschränkt ausführbar. Geht man diesen Weg mit einem Android-Tablet, ist die Performanz schon besser, reicht aber noch nicht an die eines PCs heran.

Kapitel 5

Bewertung

In diesem Kapitel soll dargestellt werden, inwiefern die funktionalen und nichtfunktionalen Anforderungen durch die Implementierungen umgesetzt wurden. Außerdem soll eine Bewertung der erstellten Anwendungen stattfinden.

5.1 Die Anwendung zur Bearbeitung der Skizze

Tabelle 5.1 fasst die Umsetzung der funktionalen Anforderungen aus Abschnitt 3.1.2 durch die Implementierung zusammen:

Anforderung	Umsetzung	Bemerkung
FA-0.1 (Skizze auswählen und darstellen)	ja	
FA-0.2 (Maßstabsbestimmung)	ein-geschränkt	nur genau bei Ähnlichkeits-transformation
FA-0.3 (Auswählen von Punkten per Mausklick)	ja	Benutzer wählt aus - die Anwendung macht keine Vorschläge
FA-0.4 (Eingabe der geografischen Koordinaten)	ja	Benutzer muss geografische Koordinaten zuvor aus anderer Quelle ermitteln
FA-0.5 (Auslösen der Transformation)	ja	Anwendung bestimmt Transformations-art aus Anzahl der Passpunkte
FA-0.6 (Bildlaufleisten)	ja	
FA-0.7 (Zoomfähigkeit)	nein	
FA-0.8 (Abspeichern des Ergebnisses)	ja	Speicherung der Grafik im PNG-Format

Tabelle 5.1: Umsetzung der funktionalen Anforderungen aus Abschnitt 3.1.2

Die funktionalen Anforderungen aus Abschnitt 3.1.2 wurden fast alle umgesetzt. Die nicht oder nur teilweise implementierten Anforderungen beeinträchtigen jedoch nicht die Grundfunktionalität der Transformation.

Wie bei der FA-0.4 wurde nicht immer die benutzerfreundlichste Variante umgesetzt. Auch hätte man die Information über die Passpunkte so weiterreichen können, dass die Anwendung unter Android darauf Zugriff hat. Die Nutzung der Anwendung zur Bearbeitung der Skizze wird durch die Tatsache eingeschränkt, dass die Implementierung nur für Bereiche nördlich des Äquators und östlich des Nullmeridians ausgelegt ist.

Die Tabelle 5.2 stellt die Umsetzung der nichtfunktionalen Anforderungen aus Abschnitt 3.1.3 dar:

Anforderung	Umsetzung	Bemerkung
NFA-0.1 (Benutzbarkeit)	ja	nach der subjektiven Auffassung des Autors ist diese Anforderung erfüllt
NFA-0.2 (Robustheit)	ja	bei fehlerhaften Eingaben stürzt die Anwendung nicht ab
NFA-0.3 (Performanz)	ja	keine Verzögerung fühlbar
NFA-0.4 (Verfügbarkeit)	ja	
NFA-0.5 (Plattform)	ja	die Ausführbarkeit auf einem Windows-7-PC wurde realisiert

Tabelle 5.2: Umsetzung der nichtfunktionalen Anforderungen aus Abschnitt 3.1.3

Die nichtfunktionalen Anforderungen aus Abschnitt 3.1.3 wurden vollständig umgesetzt. Zu NFA-0.2 ist anzumerken, dass bei unvollständigen Eingaben der Benutzer darauf hingewiesen wird. Bei der Eingabe der geografischen Koordinaten und der Bildschirmbreite werden Zahlen erwartet. Werden hier vom Benutzer z. B. Buchstaben eingegeben, kommt es zu einer Number-Format-Exception, ohne dass dem Benutzer der GUI eine entsprechende Meldung angezeigt wird. Dies könnte man noch verbessern.

Zu Abschnitt 3.4 ist zu bermerken, dass das Testen noch hätte intensiviert werden können, um nicht nur den Abnahmetest sondern auch die anderen Testarten ausführlich umzusetzen.

5.2 Die Anwendung zur Darstellung der Skizze unter Android

In Tabelle 5.3 wird die Umsetzung der funktionalen Anforderungen aus Abschnitt 4.1.2 durch die Implementierung dargestellt:

Anforderung	Umsetzung	Bemerkung
FA-1.1 (Auswahl einzubettende Grafikdatei)	ja	
FA-1.2 (Parametereingabe zur geografischen Verortung)	ja	Eingabe der beiden begrenzenden Längen- und Breitengrade
FA-1.3 (Anzeige des Ergebnisses auslösen)	ja	
FA-1.4 (Verschieben der Karte - Skizze bewegt sich analog)	ja	
FA-1.5 (Zoomen der Karte - Skizze zoomt analog)	ja	

Tabelle 5.3: Umsetzung der funktionalen Anforderungen aus Abschnitt 4.1.2

Die in Abschnitt 4.1.2 aufgestellten funktionalen Anforderungen wurden alle realisiert. Die FA-1.2 wurde in einer Variante umgesetzt, die vom Benutzer die Eingabe von zwei Längen- und Breitengraden verlangt. Dieser Aspekt könnte noch vereinfacht und somit benutzerfreundlicher werden. Die Existenz eines Speicherlecks ist unschön und sollte von den Entwicklern der Android-Plattform ausgeräumt werden.

Die Tabelle 5.4 führt die Umsetzung der nichtfunktionalen Anforderungen aus Abschnitt 4.1.3 auf:

Anforderung	Umsetzung	Bemerkung
NFA-1.1 (Benutzbarkeit)	ja	
NFA-1.2 (Robustheit)	ja	bei fehlerhaften Eingaben stürzt das Programm nicht ab
NFA-1.3 (Performanz)	eingeschränkt	Verzögerung durch Internetzugriff des Kartendienstes
NFA-1.4 (Verfügbarkeit)	eingeschränkt	Kartendienst benötigt Internetzugang
NFA-1.5 (Plattform)	ja	lauffähig auf Endgeräten mit Android in der Version 2.2 oder höher

Tabelle 5.4: Umsetzung der nichtfunktionalen Anforderungen aus Abschnitt 4.1.3

Bis auf die Einschränkungen, die die Nutzung des Internets mit sich bringt, konnten die nichtfunktionalen Anforderungen aus Abschnitt 4.1.3 umgesetzt werden. Zu NFA-1.2 ist anzumerken, dass fehlerhafte Eingaben nicht möglich sind bzw. durch eine Fehlermeldung abgefangen werden. Falls der Benutzer ein fehlerhaftes Rechteck definiert, dessen östliche Begrenzung z. B. westlich der westlichen Begrenzung liegt, erscheint zwar die Karte des

Kartendienstes. Die Skizze wird aber nicht eingeblendet. In diesem Fall könnte man noch eine entsprechende Fehlermeldung implementieren.

Wie bei der Anwendung zur Bearbeitung der Skizze hätte auch für diese Anwendung das Testen intensiviert werden können.

Kapitel 6

Zusammenfassung und Ausblick

Abschließend soll jetzt die Arbeit zusammengefasst und ein Ausblick auf Ansatzpunkte gegeben werden, an denen die Arbeit weitergeführt und verbessert werden könnte.

6.1 Zusammenfassung

Zunächst wurden die Grundlagen für diese Arbeit vorgestellt. Es wurde auf den Begriff *Landkarte* eingegangen und auf ihre Darstellung im Vektor- bzw. Rastermodell. Anschließend wurden Verfahren zur Koordinatentransformation vorgestellt, mit denen sich die Annährerung einer skizzierten Landkarte an die Realität erreichen lässt. Außerdem wurde die Softwareplattform Android erläutert.

Eine Anwendung zur Bearbeitung einer skizzierten Landkarte wurde konzipiert und implementiert. In der Implementierung wurden zwar nicht alle funktionalen Anforderungen vollständig realisiert, die Grundfunktionalität der Anwendung wird jedoch nicht beeinträchtigt. Allerdings lassen sich nur Landkarten korrekt transformieren, die nördlich des Äquators und östlich des Nullmeridians liegen.

Auch die Konzipierung und Implementierung einer Anwendung zur Darstellung der Skizze unter Android brachte eine funktionierende Anwendung hervor. Die funktionalen und nichtfunktionalen Anforderungen konnten bis auf die Beschränkung, die eine erforderliche Internetanbindung mit sich bringt, vollständig umgesetzt werden.

Der angestrebte Schritt von einer skizzierten Landkarte hin zu ihrer Einbettung in einen Kartendienst eines mobilen Android-Gerätes ist erreicht worden.

6.2 Ausblick

Für die Anwendung zur Bearbeitung der Skizze könnten noch weitere Arten der Koordinatentransformation implementiert werden. Denkbar ist auch, einen anderen Ansatz zu entwickeln, mit dem sich die Skizze transformieren lässt.

Die Benutzerfreundlichkeit der Anwendung könnte noch verbessert werden. Beispielsweise ist das Vorgehen zur Ermittlung und Eingabe der geografischen Koordinaten der Passpunkte in der realisierten Form aufwändig. In Abschnitt 3.1 wurde eine besser handhabbare Variante vorgestellt.

Auch im Hinblick auf das Zusammenspiel beider Anwendungen, könnte das Vorgehen vereinfacht werden. Die geografischen Koordinaten, die in der Anwendung zur Bearbeitung der Skizze vom Benutzer angegeben wurden, könnten an die andere Anwendung weitergegeben werden, indem sie z. B. durch die Metadaten der Grafikdatei transportiert werden. In der Anwendung zur Darstellung der Skizze könnte dann auf eine Eingabe geografischer Koordinaten verzichtet werden.

Für die Darstellung der Skizze unter Android wurde der Kartendienst Google Maps benutzt. An seiner Stelle könnte man mit weiteren Kartendiensten, wie z. B. OpenStreetMap oder Bing Maps, experimentieren. Zu begrüßen wäre die Behebung des Speicherlecks durch die Entwickler der Android-Plattform in zukünftigen Android-Versionen.

Die in Abschnitt 4.5 beschriebene Möglichkeit zur Darstellung der Skizze mittels KML-Datei ist dann eine gute Alternative, wenn die Funktionalität von Google Maps auf dem PC sowie auf der Android-Plattform in dieser Hinsicht überarbeitet wird. So könnte dann mit dem Features „Meine Orte” auf dem PC eine Karte mit einer eigebetteten Skizze erstellt werden und auf dem Android-Gerät mit dem Feature „Meine Karten” geöffnet werden.

Anhang A

Inhalt der CD-ROM

Die dieser Arbeit beiliegende CD-ROM hat folgende Verzeichnisstruktur:

- **Anwendung 01**
  - **Transformer**: Der Sourcecode der Anwendung zur Bearbeitung der Skizze inklusive der Bibliotheken des batik Java SVG Toolkits
  - **Beispieldaten**: Beispieldaten zur Nutzung mit der Anwendung
- **Anwendung 02**
  - **DisplaySketch**: Der Sourcecode der Anwendung zur Darstellung der Skizze unter Android
  - **Beispieldaten**: Beispieldaten zur Nutzung mit der Anwendung
  - **SD-Karte**: IMG-Datei einer SD-Karte, die die Datei *sample.png* enthält
- **Dokumente**: diese Bachelorarbeit sowie eine Anleitung zur Installation und Bedienung der erstellten Anwendungen im PDF-Format

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[android developers 2011] ANDROID DEVELOPERS: *Google Map View*. 2011. – URL `http://developer.android.com/resources/tutorials/views/hello-mapview.html`. – Zugriffsdatum: 03.11.2011

[Becker und Pant 2010] BECKER, Arno ; PANT, Marcus: *Android 2 : Grundlagen und Programmierung*. 2. akt. u. erw. Aufl. dpunkt.verlag GmbH, 2010. – ISBN 9783898646772

[Bronstein u. a. 2008] BRONSTEIN, Ilja N. ; SEMENDJAJEW, Konstantin A. ; MUSIOL, Gerhard ; MÜHLIG, Heiner: *Taschenbuch der Mathematik*. 7. vollst. überarb. u. erg. Aufl. Verlag Harri Deutsch, 2008. – ISBN 9783817120178

[Diercke 2002] DIERCKE, Carl: *Diercke Weltatlas*. 5. akt. Aufl. Westermann, 2002. – ISBN 3141006008

[Hill 2006] HILL, Linda L.: *Georeferencing : The Geographic Associations of Information*. The MIT Press, 2006. – ISBN 9780262083546

[Java.net 2011] JAVA.NET: *SVG Salamander*. 2011. – URL `http://svgsalamander.java.net/`. – Zugriffsdatum: 07.08.2011

[de Lange 2005] LANGE, Norbert de: *Geoinformatik : in Theorie und Praxis*. 2. akt. u. erw. Aufl. Springer, 2005. – ISBN 9783540282914

[Luhmann 2010] LUHMANN, Thomas: *Nahbereichsphotogrammetrie : Grundlagen, Methoden und Anwendungen*. 3. völlig neu bearb. u. erw. Aufl. Wichmann, 2010. – ISBN 9783879074792

[Meier 2010] MEIER, Reto: *Professional Android 2 Application Development*. Wrox, 2010. – ISBN 9780470565520

[Mosemann und Kose 2009] MOSEMANN, Heiko ; KOSE, Matthias: *Android : Anwendungen für das Handy-Betriebssystem erfolgreich programmieren*. Carl Hanser Verlag, 2009. – ISBN 9783446417281

[Open Geospatial Consortium 2011] OPEN GEOSPATIAL CONSORTIUM: *KML*. 2011. – URL `http://www.opengeospatial.org/standards/kml/`. – Zugriffsdatum: 18.10.2011

[open handset alliance 2011] OPEN HANDSET ALLIANCE: *Homepage*. 2011. – URL `http://www.openhandsetalliance.com/`. – Zugriffsdatum: 20.10.2011

[Resnik und Bill 2003] RESNIK, Boris ; BILL, Ralf: *Vermessungskunde für den Planungs-, Bau- und Umweltbereich*. 2. völlig neu bearb. u. erw. Aufl. Wichmann, 2003. – ISBN 3879073996

[Selinger 2011] SELINGER, Peter: *Potrace*. 2011. – URL `http://potrace.sourceforge.net/`. – Zugriffsdatum: 13.09.2011

[Spillner und Linz 2005] SPILLNER, Andreas ; LINZ, Tilo: *Basiswissen Softwaretest*. 3., überarb. u. aktualisierte Aufl. dpunkt Verlag, 2005. – ISBN 3898643581

[The Apache XML Graphics Project 2010] THE APACHE XML GRAPHICS PROJECT: *batik Java SVG Toolkit*. 2010. – URL `http://xmlgraphics.apache.org/batik/index.html`. – Zugriffsdatum: 06.08.2011

[World Wide Web Consortium 2011] WORLD WIDE WEB CONSORTIUM: *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. 2011. – URL `http://www.w3.org/TR/SVG11/`. – Zugriffsdatum: 06.08.2011

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw.§24(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Halstenbek, 07.12.2011 Marcus Rohwer