Institut für Rechnergestützte Ingenieursysteme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3500

Entwicklung einer Schnittstelle zur multidisziplinären Entwurfsoptimierung im Bereich Städteplanung

Oleg Martin

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Dieter Roller
Betreuer/in:	M. Sc. Julian Eichhoff
Beginn am:	19. Juni 2013
Beendet am:	19. Dezember 2013
CR-Nummer:	J.6, I.2.4, F.4.2, G.1.6

Kurzfassung

Multidisziplinäre Optimierung von komplexen Systemen sowie die Formulierung eines Optimierungsproblems und deren nahtlose Einbindung in den Bereich der Städteplanung ist Gegenstand der aktuellen Forschung. Architekten im Bereich des Städtebaus befassen sich typischerweise mit dem räumlichen Entwurf von großflächigen Stadtstrukturen sowie deren sozialen und technologischen Implikationen, wodurch die Notwendigkeit einer optimalen Verwendung von Bauflächen in den Vordergrund rückt. In der vorliegenden Diplomarbeit werden Lösungswege für den Einsatz multidisziplinärer Entwurfsoptimierung im Bereich der Städteplanung mit bereits existierenden Softwarewerkzeugen erarbeitet und erprobt. Das Ziel dieser Arbeit ist es, Architekten und Städteplaner mit Hilfe einer einheitlichen Schnittstelle zwischen den existierenden Softwarewerkzeugen bei der Entwurfsfindung zu unterstützen und damit eine nachhaltige städtebauliche Entwicklung eines Stadtgebietes zu ermöglichen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Abkürzungsverzeichnis

AAA-Modell	AFIS-ALKIS-ATKIS-Modell
AdV	Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland[1]
AFIS	Amtliches Festpunktinformationssystem
ALKIS	Amtliches Liegenschaftskatasterinformationssystem
ATKIS	Amtliches Topographisch-Kartographisches Informationssystem
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CGA	Computer Generated Architecture
DLM	Digitales Landschaftsmodell
DXF	Drawing Interchange File Format by Autodesk[2]

[1] siehe `http://www.adv-online.de`
[2] siehe `http://www.autodesk.de`

EPSG ..... European Petroleum Survey Group Geodesy[3]
GIS ..... Geoinformationssystem
GML ..... Geography Markup Language
GPS ..... Global Positioning System
GUI ..... Graphical User Interface
ISO ..... International Organization for Standardization
LGL ..... Landesamt für Geoinformation und Landentwicklung BW[4]
MVC ..... Model-View-Controller
NAS ..... Normbasierte Austauschschnittstelle
OGC ..... Open Geospatial Consortium[5]
OSM ..... OpenStreetMap[6]
PSO ..... Particle Swarm Optimization
UI ..... User Interface
UML ..... Unified Modeling Language
WFS ..... Web Feature Service
WGS ..... World Geodetic System
WMS ..... Web Map Service
WSDL ..... Web Services Description Language
XML ..... Extensible Markup Language

[3]siehe `http://www.epsg.org`
[4]siehe `https://www.lgl-bw.de/lgl-internet/opencms/de/index.html`
[5]siehe`http://www.opengeospatial.org`
[6]siehe `http://www.openstreetmap.org`

1. Einleitung

1.1. Einführung

Durch den technologischen Fortschritt der letzten Jahre nahm das computerbasierte Entwerfen in der Architektur wie auch in der Stadtplanung eine der bedeutendsten Rollen ein. Die Nutzung von computergestützten Werkzeugen und generativen Verfahren wurde nicht nur beim Design und Entwurf von Gebäuden sondern auch bei der Entwicklung von neuen Stadtformen immer wichtiger.

Neben CAD-Werkzeugen zur Darstellung eines städtebaulichen Entwurfs, Simulationswerkzeugen zur Stadtplanung und Software zur Optimierung haben sich beim Entwurf sowie der Optimierung und Analyse von Städten Geoinformationssysteme, kurz GIS, zu einem der wichtigsten Hilfsmittel entwickelt.

Aufgrund der immer komplexer werdenden integrativen Planung einer Stadt besteht daher eine Notwendigkeit diesen Entwurfsprozess für den Stadtplaner zu vereinfachen und dabei möglichst ein optimales Ergebnis hinsichtlich städtebaulicher Kriterien zu erzielen. Um dies zu erreichen, müssen zahlreiche Bereiche der Stadtplanung und Informatik sowie die verfügbaren Werkzeuge betrachtet und verknüpft werden. Die Erforschung eines kombinierten Einsatzes dieser Fachbereiche und Werkzeuge blieb lange aus, was in der hohen Komplexität der Integration aller benötigten Softwaresysteme und der zahlreichen problemspezifischen Ansätze seine Gründe hat.

1.2. Stadtplanung in der Architektur

Stadtplanung ist ein weiter Teilbereich der Architektur und beschäftigt sich überwiegend mit dem Entwurf einer Stadt sowie mit den räumlichen und sozialen Strukturen darin. In der Stadtplanung werden hinsichtlich sozialer und technologisch relevanter Strukturen Planungskonzepte unter Abwägung von öffentlichen und privaten Belangen mit dem Ziel der Konfliktminimierung erarbeitet. Dabei steuert die Stadtplanung im Wesentlichen die Bodennutzung im urbanen Gebiet mit dem Ziel einer nachhaltigen städtebaulichen Entwicklung der Stadt und deren Teilgebiete.

Der Begriff „Stadtplanung“ deutet dabei an, dass sich eine Stadt in ihrer Gesamtentwicklung planerisch erfassen ließe. In der Geschichte wurden viele Versuche unternommen unter der Anwendung von naturwissenschaftlichen Erkenntnissen eine städtische Entwicklung vorauszubestimmen. Noch zu Beginn des 20. Jahrhunderts verlangte die Industrialisierung

unter dem explosionsartigen urbanen Wachstum eine funktionale Neuordnung der Städte nach rationalen Gesichtspunkten und neuen Ordnungsprinzipien. Mit dem Aufkommen des Computers in 1960er Jahren schien die Realisierung einer genaueren Vorausplanung der Stadtentwicklung möglich.

Die zeitliche Entstehung einer Stadt und die Nutzung der Stadtfläche ist demnach Forschungsgegenstand des Wissenschaftsgebietes der Stadtmorphologie. Dieses Wissenschaftsgebiet besteht seit einigen Jahrzehnten und beschreibt dabei die Formprinzipien, nach denen Stadtgrundrisse aufgebaut werden und nach denen Städte entstehen bzw. entstanden sind. Wichtige Aspekte stellen dabei die Entstehungsbedingungen und die räumlichen Eigenarten der Stadtfläche dar. Die Gestalt einer Stadt wird durch politische, soziale, wirtschaftliche und technische Bedingungen der jeweiligen Zeit bestimmt. Ein Ziel der Stadtmorphologie ist dabei die optimale Nutzung der zur Verfügung stehende Bodenfläche (vgl. [MCF93] und [Cur96]). Als Folge daraus werden städtebauliche Maßnahmen als Hilfsmittel für den Entwurf eines optimalen Stadtplans ergriffen.

1.2.1. Optimale Stadtplanung

Unter optimaler Stadtplanung versteht man in der Architektur den Versuch eine integrative Planung von großflächigen Stadtstrukturen unter gleichzeitiger Berücksichtigung möglichst vieler städtebaulicher, technologischer, sozialer, politischer und wirtschaftlicher Aspekte zu finden. Unter den gegebenen Bedingungen stellt die Erstellung bzw. das Auffinden eines solchen Entwurfs für den Städtebauarchitekten eine besondere Herausforderung dar.

1.2.2. Grundlegende Probleme

Ausgehend von der geschichtlichen Entwicklung der Stadtplanung sind alle Versuche einer detailgenauen Stadtplanung bis heute gescheitert. So wird im Allgemeinen angenommen, dass städtische Entwicklungsprozesse in ihrer Gesamtheit viel zu komplex und zum Teil auch widersprüchlich sind, als dass sie zeit- und praxisnah erfasst und fortgeschrieben werden könnten (vgl. [MCF93]).

Es gibt demnach keine Stadt mit „exklusiven“ Rahmenbedingungen, sodass jede Stadtentwicklung nicht vorhersehbaren Einflüssen ausgesetzt sowie in überregionale Austauschprozesse eingebunden ist. Trotz dieser grundlegenden Problematik ist Stadtplanung bis heute unverzichtbar. Sie bildet unter Anderem die Grundlage zum Verständnis von allgemeinen Stadtformen und stellt Perspektiven zur Integration vieler Einzelentwicklungen im urbanen Bereich bereit.

Das grundlegende Problem der Komplexität einer integrativen Stadtplanung wird durch die steigende Anzahl neuer Technologien, die bei der Stadtplanung mitberücksichtigt werden müssen, weiter verstärkt. Dadurch wird die Stadtplanung zunehmend komplexer und unüberschaubarer. Der Versuch einen optimalen Stadtplan zu finden, der möglichst alle Aspekte bzw. Kriterien einer Stadt auf einmal einbezieht, führt zwangsläufig zu einem multidisziplinären

Optimierungsproblem. Aus wissenschaftlicher Sicht ist die Formulierung dieses Optimierungsproblems, der Lösungsansatz und die in Frage kommenden Lösungsalgorithmen in Bezug auf die Städteplanung ein interessantes Feld, welches bei einem stetig steigenden Urbanisierungsgrad der Ballungsgebiete seine Daseinsberechtigung besitzt.

1.3. Anmerkungen

Ein Teil der verwendeten Literatur liegt ausschließlich in englischer Sprache vor. Für die Nutzung und Referenzierung solcher englischen Originaltexte verwendet der Verfasser dieses Dokuments eigenhändig angefertigte Übersetzungen. Diese wurden nach bestem Wissen und Gewissen vorgenommen. Alle vorkommenden Fachbegriffe werden, sofern dies möglich ist und es sich nicht um Eigennamen handelt, durch entsprechende deutsche Begriffe ersetzt.

Anerkannte Technologien und Standards, aber auch bekannte Markennamen, werden standardmäßig durch die Angabe ihrer offiziellen Web-Präsenz in einer Fußnote referenziert.

Das erste Vorkommen neu eingeführter Fachbegriffe und Akronyme wird vom restlichen Text durch eine kursive *Schriftauszeichnungsart* hervorgehoben.

Alle besprochenen Grundlagen und Hintergründe erheben keinen Anspruch auf Vollständigkeit. Sie beschränken sich lediglich auf die zum Verständnis des vorliegenden Sachverhalts erforderlichen Themen.

1.4. Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Problemstellung: Kapitel 2 beschreibt detailliert die Problemstellung hinsichtlich der Entwicklung einer Schnittstelle zur multidisziplinären Entwurfsoptimierung im Bereich der Stadtplanung. Das Problem der multidisziplinären Entwurfsoptimierung wird hierbei in zwei Teilbereichen betrachtet. Es wird im einzelnen auf das praktische Problem der Städtebauarchitekten in der Stadtplanung sowie auf das implizierte technische Problem aus Sicht der Informatik zur Realisierung einer Problemlösung detailliert eingegangen.

Kapitel 3 – Forschungsstand: In Kapitel 3 wird der aktuelle Stand der Wissenschaft und der Technik betrachtet. Eine detaillierte Übersicht existierender Ansätze in den Bereichen Optimierung von komplexen Systemen, multidisziplinären Entwurfsoptimierung in der Stadtplanung und Schnittstellen zwischen GIS und CAD-Systemen bildet die Grundlage dieser Arbeit.

Kapitel 4 – Architektur und Konzeption: Die Konzeptions- und Entwurfssphase der Schnittstelle zur multidisziplinären Entwurfsoptimierung in der Stadtplanung sind die zentralen Themen dieses Kapitels. Neben einer Anforderungsanalyse findet eine detaillierte Beschreibung und kritische Beurteilung der Architektur, des logischen Datenmodells, der einzelnen Applikationskomponenten sowie diverser Anwendungsfälle und Verhaltensbeschreibungen statt.

Kapitel 5 – Optimierungsframework: Mittelpunkt von Kapitel 5 ist die Definition und Erklärung der für diese Arbeit essentiellen formalen Designsprache und des Optimierungsmodells. Im Einzelnen soll eine grundlegende Designsprache vorgestellt werden, die die Entwurfsmuster, Formgrammatiken und Erzeugungsregeln zur prozeduralen Erzeugung eines Stadtplans beschreibt. Weiterhin wird das verwendete Optimierungsmodell sowie das Optimierungsframework im Ganzen beschrieben und diskutiert.

Kapitel 6 – Implementierung: Die bisherigen theoretischen Erkenntnisse werden in diesem Kapitel von ihrer praktischen Seite beleuchtet. Die eingesetzten externen Frameworks zur Handhabung der festgelegten Anforderungen des Projekts werden vorgestellt und ihre Funktionsweise anhand ausgewählter Schlüsselkomponenten aufgezeigt.

Kapitel 7 – Evaluation: In Kapitel 7 wird die Evaluation der implementierten Schnittstelle beschrieben. Dabei wird zuerst im Kontext der Evaluierung das verwendete Vorgehensmodell diskutiert und die Evaluationsergebnisse dargestellt. Anschließend werden detaillierte Schlussfolgerungen aus den Ergebnissen gezogen.

Kapitel 8 – Zusammenfassung und Ausblick fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor. Unter anderem werden in diesem Kapitel die erreichten Resultate mit Blick auf die Problemstellung interpretiert, die Vor- und Nachteile der gefundenen Lösung herausgearbeitet sowie die Auswirkungen des gefundenen Lösungsansatzes auf das praktische Problem vorgestellt.

2. Problemstellung

Im Bereich des Städtebaus bzw. der Städteplanung befassen sich Architekten mit dem Entwurf und der Planung von großflächigen Stadtstrukturen. Gegenwärtig werden immer mehr neue Technologien entwickelt, die bei der Städteplanung mitberücksichtigt werden müssen. Dadurch wird die Städteplanung zunehmend komplexer und unüberschaubarer. Unter anderem müssen verschiedene Technologien, wie zum Beispiel Elektromobilität, moderne Gebäudetechnik, Wasserversorgung, Informations- und Kommunikationstechnik schon früh mit all ihren unterschiedlichen Aspekten beim städtebaulichen Entwurf berücksichtigt werden. Da diese verschiedenen Aspekte zusätzlich vernetzt und voneinander abhängig sind, müssen sie im Sinne der integrativen Planung gleichzeitig betrachtet werden. Unter diesen Bedingungen ist es für Städtebauarchitekten eine besondere Herausforderung einen optimalen Entwurf zu finden, welcher die Erwartungen der Menschen an die Technologien in der Stadt am besten erfüllt und Infrastrukturen effizient und möglichst optimal nutzbar macht.

Die zentrale Frage, auf die in dieser Diplomarbeit eine Antwort gefunden werden soll, lautet daher: Ist es möglich alle für die Stadtplanung relevanten Aspekte einer Stadt formal so abzubilden, dass ein optimaler Masterplan in Bezug auf diese Aspekte in annehmbarer Zeit und mit vertretbarem Aufwand aus einer vorgegebenen Menge möglicher Pläne gefunden werden kann? Anhand dieser Fragestellung lässt sich das multidisziplinäre Optimierungsproblem in der Städteplanung in zwei Teilprobleme aufteilen: die eigentliche Stadtplanfindung als praktisches Problem des Stadtplaners und den Entwurf einer Schnittstelle zwischen den verwendeten Softwaresystemen als technisches Informatikproblem.

2.1. Praktisches Problem aus Sicht der Stadtplanung

Das praktische Problem umfasst die multidisziplinäre Entwurfsoptimierung in der Stadtplanung. Dabei soll ein näherungsweise optimaler Masterplan gefunden werden, welcher bestimmten Zielen der Stadtplanung gerecht wird. Die Ziele können dabei politischer, sozialer wie auch wirtschaftlicher Natur sein und zum Beispiel die Verminderung der Luftverschmutzung, die Vermeidung von Verkehrsstaus, die wirtschaftlich sinnvolle Nutzflächenverteilung im Stadtbereich oder die Verbesserung des Nahverkehrs darstellen.

2.1.1. Multidisziplinäre Entwurfsoptimierung

Für bestimmte multidisziplinäre Optimierungsprobleme im Bereich der Stadtplanung wurden bisher nur für den Problemfall angepasste Lösungen entwickelt. Dies ist zum einen

auf die allgemeine Stadtstruktur einer jeden Stadt, welche keine festen Rahmenbedingungen für den Optimierungsansatz aufweist, zurückzuführen. Zum anderem ist dies der hohe Anzahl der verschiedenen Softwaresysteme und computergestützten Verfahren, die heutzutage bei der Stadtplanung eingesetzt werden und deren komplexer Integration in den Entwurfsprozess der Stadt geschuldet. So werden unter anderem Geoinformationssysteme, kurz GIS, zur Informationserfassung, Bearbeitung, Organisation, Analyse und Präsentation geografischer bzw. räumlicher Daten verwendet. Erwähnenswert ist hier das kommerzielle GIS des US-amerikanischen Softwareherstellers Esri Inc.[1] mit dem Namen ArcGIS[2], welches eines der meistbenutzten GIS in der Stadtplanung ist. Zusätzlich werden verschiedene Geodatenmodelle, wie das AFIS-ALKIS-ATKIS-Modell der Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV[3]) zur Beschreibung der geografischen und amtlichen Informationen in einem digitalen Landschaftsmodell mit der Abkürzung DLM verwendet. Außerdem werden Verfahren des computerbasierten Entwerfens eingesetzt, um Stadtstrukturen prozedural erzeugen sowie Designoptimierungen auf diesen anwenden zu können. Ebenfalls werden gängige CAD-Systeme wie AutoCAD[4] zur Darstellung von Stadtplänen verwendet und angepasste Simulationsframeworks in der Stadtplanung eingesetzt.

Um nun eine einheitliche Lösung für multidisziplinäre Optimierungsprobleme im Bereich der Stadtplanung zu erhalten, müssen diese existierenden Softwarewerkzeuge und Verfahren im kombinierten Einsatz in den Entwurfsprozess integriert werden. Um dies zu erreichen, wird dieses technische Problem im Folgenden aus Sicht der Informatik betrachtet.

2.2. Technisches Problem aus Sicht der Informatik

Das praktische Problem der Stadtplaner impliziert direkt das technische Problem, welches die kombinierte Verwendung der verschiedenen Softwaresysteme zur Lösung des praktischen Problems umfasst. Aus der kombinierten Verwendung der einzelnen Softwaresysteme und computergestützten Entwurfsverfahren folgt wiederum die Notwendigkeit einer einheitlichen Schnittstelle zwischen den Softwaresystemen. Im Folgenden wird die Problemstellung der Schnittstellenentwicklung betrachtet.

2.2.1. Schnittstellenentwicklung

Die zu entwickelnde einheitliche Schnittstelle soll für die verwendeten Geoinformationssysteme, CAD-Systeme, Optimierungs- und Simulationsframeworks zur Lösung von multidisziplinären Optimierungsproblemen in der Stadtplanung entworfen und prototypisch mit der Programmiersprache Java realisiert werden. Damit wäre eine formal korrekte Definition

[1] siehe `http://www.esri.com`
[2] siehe `http://www.esri.com/software/arcgis`
[3] siehe `http://www.adv-online.de`
[4] siehe `http://www.autodesk.de`

der Kommunikation zwischen den einzelnen Softwaresystemen und damit die automatische Abbildung typischer Planungsprobleme auf verfügbare Verfahren zur multidisziplinären Entwurfsoptimierung möglich. Die Realisierung einer solchen Schnittstelle wäre für den Städtebauarchitekten eine bedeutende Hilfe im Entwurfsprozess und ein wichtiger Schritt hin zur Lösung des oben aufgezeigten Problems.

3. Forschungsstand

Im Bereich der multidisziplinären Entwurfsoptimierung existieren bereits Ansätze zur Lösung planerischer Probleme. Durch eine entsprechende Formulierung der Teilprobleme aus den verschiedenen Planungsfeldern und deren Verknüpfung zu einem Gesamtproblem lassen sich optimale Lösungen, z. B. durch evolutionäre Algorithmen, automatisch ermitteln. Dabei wird aber schnell deutlich, dass bevor das Optimierungsproblem an einen Lösungsalgorithmus übergeben werden kann, es relativ aufwändig formal korrekt definiert werden muss.

3.1. Übersicht existierender Ansätze

Der Bereich der Entwurfsoptimierung in der Städteplanung lässt sich im Groben in drei Teilbereiche einteilen:

- **Optimierung komplexer Systeme** basierend auf Verfahren der angewandten und diskreten Mathematik zur Lösung von Optimierungsproblemen
- **Multidisziplinäre Entwurfsoptimierung bei der Stadtplanung** bzw. Planung der Bodennutzung für die vorhandenen Teilflächen einer Stadtfläche im GIS
- **Schnittstelle zwischen GIS und CAD-System** als Verknüpfung der multidisziplinären Entwurfsoptimierung bei der Planung der Bodennutzung im GIS und der Darstellung eines Stadtplans mit gängigen CAD-Systemen

3.1.1. Optimierung komplexer Systeme

Die Grundlage der multidisziplinären Entwurfsoptimierung in der Stadtplanung bildet das umfassende Gebiet der Optimierung aus der angewandten und diskreten Mathematik. Das Gebiet beschäftigt sich damit möglichst optimale Parameter eines meist komplexen Systems zu finden. Solche Problemstellungen werden als Optimierungsprobleme bezeichnet und stellen sich in vielen wissenschaftlichen Bereichen wie der Stadtplanung, Meteorologie, Operations Research oder Wirtschaftsmathematik. Grundsätzlich bestehen Optimierungsprobleme immer dann, wenn mit unbekannten Parametern gearbeitet wird und eine möglichst optimale Lösung in Abhängigkeit von den Parametern gefunden werden soll.

Ein bekanntes Beispiel eines Optimierungsproblems wäre die Vorhersage des Klimas in der Klimaforschung. Dabei stellen Klimamodelle vereinfachte numerische Systeme der eigentlichen Prozesse in der Atmosphäre dar. Es wird nun versucht diese Prozesse durch Gleichungen,

welche aus grundlegenden physikalischen Gesetzen oder statistischen Modellen abgeleitet werden, zu approximieren. Dazu müssen die Parameter der Gleichungen so optimiert werden, dass die Klimamodelle die tatsächlichen Prozesse möglichst gut abbilden.

Das Gebiet der Optimierung beinhaltet demnach neben den Optimierungsverfahren zur Lösung von skalaren Optimierungsproblemen die für die Stadtplanung wichtigen numerischen Optimierungsverfahren zur Lösung von Vektoroptimierungsproblemen bzw. multikriteriellen Optimierungsproblemen und den wichtigen Spezialfall der linearen Optimierung. Im folgenden werden diese beiden Optimierungsprobleme näher betrachtet, wobei die Bezeichnung „Programm“ als Synonym zu „Optimierungsproblem“ zu verstehen ist. Diese Tatsache ist historisch begründet indem die ersten Anwendungen der Optimierung vorwiegend militärische Probleme waren und hierbei die englische Übersetzung eines Aktionsplans „program of actions“[1] zu finden war.

Lineare und ganzzahlige lineare Optimierungsprobleme

Bei dem Spezialfall der linearen Optimierung bzw. der linearen Programmierung wird eine lineare Zielfunktion über einer Menge von Nebenbedingungen, die durch lineare Gleichungen und Ungleichungen dargestellt sind, optimiert (vgl. [JS04] und [AEHS03]). Die lineare Optimierung ist eines der Hauptverfahren im Bereich des Operations Research und dient hierbei unter anderem als Hilfsmittel zur Entwicklung und dem Einsatz quantitativer Modelle und Methoden zur Entscheidungsunterstützung bei der Ablauf- und Planungsforschung. Die am häufigsten eingesetzten Lösungsverfahren von linearen Optimierungsproblemen sind zum einen Simplex-Verfahren, welche in der Lage sind, das globale Optimum im Prinzip exakt berechnen zu können, und zum anderen die Innere-Punkt-Verfahren, die seit den 1990er Jahren bei bestimmten Arten von linearen Optimierungsproblemen in Konkurrenz zu dem Simplex-Verfahren stehen (vgl. [JS04]).

Die lineare Optimierung ist unter anderem die Grundlage der ganzzahligen linearen Optimierung, die häufig bei der Planung von komplexen Systemen, wie Verkehrs- oder Telekommunikationsnetzen, der Produktionsplanung und vor allem der Städteplanung verwendet wird. Die Beschränkung auf ganzzahlige Variablen macht das ganzzahlige lineare Optimierungsproblem deutlich komplexer erweitert aber gleichzeitig die Möglichkeiten seiner Anwendung. Es gibt verschiedene Lösungsverfahren für ganzzahlige lineare Optimierungsprobleme. Unter anderem werden exakte Lösungsverfahren, wie zum Beispiel das sogenannte Meta-Verfahren Branch-and-Bound, welches mit Hilfe von Entscheidungsbäumen die Lösung ermittelt, verwendet. Andererseits werden sogenannte Schnittebenenverfahren, die auf der Lösung vieler ähnlicher linearer Optimierungsprobleme basieren, verwendet. Zusätzlich werden neben diesen Lösungsverfahren eine Vielzahl von Heuristiken eingesetzt, welche Bewertungsfunktionen und bereits erworbene Erfahrung zur Lösungsfindung hinzuziehen. Trotz dieser Möglichkeiten ist die Lösung ganzzahliger linearer Optimierungsprobleme immer noch eine komplexe Aufgabe, die abhängig vom Problemfall speziell entwickelte und besser

[1] siehe `https://www2.informs.org/History/dantzig/in_interview_irv5.htm`

angepasste Algorithmen erfordert. Dieser Tatsache geschuldet werden daher auch mehrere Lösungsverfahren kombiniert eingesetzt.

Multikriterielle Optimierungsprobleme

Die multikriterielle Optimierung bzw. Pareto-Optimierung beschäftigt sich mit der Lösung von Optimierungsproblemen, bei denen mehrere Zielfunktionen gleichzeitig optimiert werden sollen. Das multikriterielle Optimierungsproblem beschreibt dabei ein komplexes System, dass von mehreren Zielen und Einschränkungen abhängig ist. Einzelne Lösungen, die das Optimum aller Komponenten der Zielfunktion finden, gibt es nicht. Vielmehr ist das Ergebnis einer multikriteriellen Optimierung eine Lösungsmenge aus der eine einzelne, durch Gewichtung der einzelnen Komponenten der Zielfunktion, optimale Lösung ausgewählt werden kann. Demnach werden wichtige Ziele als Teilziele aufgefasst und mittels Gewichtungsfaktoren zu einer gemeinsamen Zielfunktion zusammengefasst. Die einzelnen Teilziele bilden hierbei lineare oder nichtlineare Optimierungsprobleme, die mit den passenden Lösungsverfahren gelöst werden, um anschließend eine optimale Lösung bezüglich der gemeinsamen Zielfunktion zu bestimmen. Da die einzelnen Zielkriterien der gemeinsamen Zielfunktion meist im Konflikt zueinander stehen und eine separate Optimierung aller Kombinationsmöglichkeiten der Gewichtung der einzelnen Komponenten der Zielfunktion keine eindeutig beste Lösung hervorbringt, bestimmt man eine Menge von Lösungen des Optimierungsproblems, bei der eine Verbesserung eines Zielfunktionswertes nur noch durch Verschlechterung eines anderen erreicht werden kann. Man bestimmt also die Menge der optimalen Kompromisse. Diese Lösungsmenge, das sogenannte Pareto-Optimum, beinhaltet dann pareto-optimale Lösungen. Aus dieser Lösungsmenge kann je nach subjektiver Wichtigkeit der einzelnen Zielfunktionskriterien jeweils mindestens eine Lösung ausgewählt werden, die für diese einzelnen Gewichtungen optimal ist (vgl. [Ehr00]).

Exakte und heuristische Lösungsverfahren

Bei der linearen und ganzzahlig linearen Optimierung werden die verwendeten Lösungsverfahren in die Klassen der exakten und der heuristischen Lösungsverfahren eingeteilt. Für die exakten Lösungsverfahren ist stets beweisbar, dass ein optimales Ergebnis gefunden oder nicht gefunden werden kann bei der Voraussetzung, dass der Lösungsalgorithmus beliebig lange ausgeführt wird. Zu den exakten Lösungsverfahren gehören unter anderem die Branch-and-Bound- sowie die Schnittebenenverfahren. Die heuristischen Lösungsverfahren dagegen liefern schnell eine zulässige Lösung ohne eine Information über die Qualität der Lösung in Hinsicht auf die Optimallösung. Findet ein heuristisches Lösungsverfahren keine Lösung, so kann nicht bestimmt werden ob das Lösungsverfahren unzureichend oder das betrachtete Optimierungsproblem unlösbar ist. Zu den heuristischen Lösungsverfahren gehören unter anderem Verfahren, die minimal aufspannende Bäume nutzen sowie sogenannte Metaheuristiken einsetzen. Dabei können zum Beispiel die lokale Suche, die Tabu-Suche sowie evolutionäre Algorithmen als Metaheuristiken eingesetzt werden.

In der multidisziplinären Entwurfsoptimierung werden unter anderem metaheuristische Algorithmen zur näherungsweisen Lösung von multikriteriellen Problemen verwendet (vgl. [JMT02]). Diese Algorithmen, die allgemein auch Metaheuristiken genannt werden, werden meistens bei Problemen eingesetzt, für die kein anderer effizienter Lösungsalgorithmus bekannt ist. Im Gegensatz zu problemspezifischen Heuristiken, definieren Metaheuristiken eine abstrakte Folge von Schritten zur Lösung von Problemen. Diese können auf abstrakter Ebene auf beliebige Problemstellungen angewendet werden. Obwohl in der Regel bei der Anwendung von Metaheuristiken nicht garantiert ist, dass eine optimale Lösung gefunden wird, ist es grundsätzliche möglich eine näherungsweise optimale Lösung in annehmbarer Laufzeit zu berechnen. Dies hängt aber generell von der Definition und Implementierung der einzelnen Lösungsschritte ab.

Viele Metaheuristiken basieren unter anderem auf dem Prinzip der lokalen Suche (vgl. [RN09]). Da diese, wie ihr Name schon sagt, nur lokale Optima in einem Problemraum findet, werden verschiedene Algorithmen zur Exploration des globalen Problemraumes verwendet. Zum einen wird der Hill-Climbing-Algorithmus verwendet, bei dem mehrere Suchversuche gestartet werden, um lokalen Optima zu entkommen. Zum anderen wird das sogenannte Simulated Annealing verwendet, das bei der Suche nach dem globalen Optimum zwischenzeitlich erst größere, im später Verlauf dann nur noch kleine Verschlechterungen des Suchergebnisses akzeptiert. Weiterhin werden auch evolutionäre bzw. genetische Algorithmen verwendet, die mehrere bereits gefundene lokale Lösungen zu einer neuen Lösung kombinieren oder diese zufällig verändern. Zudem wird auch die sogenannte Tabu-Suche mit Tabu-Listen verwendet, die bereits gefundene Lösungen von der erneuten Betrachtung ausschließt. Die erwähnten Algorithmen sind nur einige aus einer großen Menge, die die lokale Suche verbessern können (vgl. [RN09]).

Ebenfalls basieren viele Metaheuristiken auf der Klasse naturinspirierter Verfahren, die eine lernende Komponente in der Suche besitzen und zum Beispiel Schwarmintelligenzen verwenden. Dazu zählen unter anderem sogenannte „Ant Colony Optimization“-Algorithmen, bei denen das natürliche Verhalten von Ameisen auf der Wegsuche modelliert und zur Lösungsfindung herangezogen wird (vgl. [JMT02]). Ein weiterer Algorithmus dieser Klasse ist der „Particle Swarm Optimization“-Algorithmus, welcher das Verhalten von Vogel- oder Fischschwärmen als Vorbild bei der Lösungsfindung hat (vgl. [MMH12]). Diese Form der Optimierung wird auch Partikelschwarmoptimierung genannt. Die weiter oben im Text erwähnten evolutionären bzw. genetischen Algorithmen mit dem Vorbild der biologischen Evolution gehören ebenfalls in die Klasse naturinspirierter Verfahren. Zusätzlich wird auch mit sogenannten künstlichen neuronalen Netzen sowie hybriden und parallelen Anwendungen der erwähnten Algorithmen versucht, komplexe Systeme zu optimieren.

Anwendung auf multidisziplinäre Entwurfsoptimierung

Um die multikriterielle Optimierung bzw. Pareto-Optimierung auf die multidisziplinäre Entwurfsoptimierung in der Städteplanung übertragen zu können, muss das Optimierungsproblem, welches hier das komplexe System der Stadt und alle ihre infrastrukturellen Abhängigkeiten umfasst, formal definiert werden. Dazu muss das komplexe System der Stadt und

damit das Optimierungsproblem in Teilprobleme zerlegt und formal beschrieben werden, damit der eigentliche Optimierungsprozess, wie er auf der Abbildung 3.1 zu sehen ist, auf dem formalen Optimierungsmodell ablaufen kann. Diese Zerlegung des Optimierungsproblems bezeichnet man als Dekomposition (vgl. [Gey09] und [GWF11]). Die Dekomposition kann entsprechend „top-down“ oder „bottom-up“ erfolgen. Eine „Top-down“-Dekomposition bedeutet dabei, dass ein vorhandenes Gesamtproblem in Teilprobleme zerlegt und anschließend optimiert wird. Im Gegensatz dazu spricht man von einer „Bottom-up“-Dekomposition, wenn aus vorhandenen Teilproblemen und deren Verknüpfung ein Gesamtproblem entsteht und dieses anschließend optimiert wird. Bei beiden Ansätzen ist die entsprechende Formulierung des Optimierungsproblems und das daraus resultierende Optimierungsmodell von großer Bedeutung für die Effizienz und Laufzeit der Optimierung.

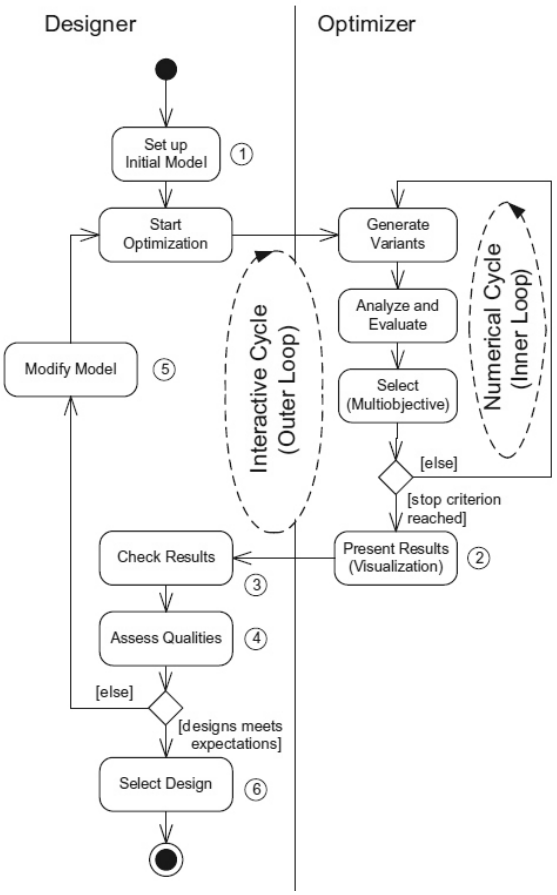

Abbildung 3.1.: Optimierungsprozess für multikriterielles Optimierungsproblem, wie er zum Beispiel in der Stadtplanung eingesetzt werden kann. Dabei zeigt die Abbildung den Zusammenhang zwischen interaktivem Designprozess und Optimierungsprozess (vgl. [Gey09])

3.1.2. Multidisziplinäre Entwurfsoptimierung in der Stadtplanung

Die multidisziplinäre Entwurfsoptimierung bei der Planung der Bodennutzung für die vorhandenen Teilflächen einer Stadtfläche entstand in den 1960er Jahren als die erste Stadtplanungssoftware, sogenannte Geoinformationssysteme bzw. GIS, entwickelt wurden. Die multidisziplinäre Entwurfsoptimierung nutzte dabei unter anderem Optimierungsmodelle auf Basis der linearen Optimierung (vgl. [AEHS03] und [Chu93]). Auf Grundlage dieser hat sich die die ganzzahlige lineare Optimierung seit ihren Anfängen in den 1950er Jahren noch stärker zu einem Modellierungs- und Optimierungswerkzeug für viele praktische Probleme, für die keine speziellen Algorithmen bekannt sind, entwickelt. Mit Hilfe der ganzzahligen linearen Optimierung bzw. der ganzzahligen linearen Programmierung wird dabei zum Beispiel die Zusammensetzung der Bodennutzung im Stadtbereich abhängig von verschiedenen strukturellen Aspekten einer Stadt optimiert. Konkret werden hierbei lineare Zielfunktionen über einer Menge Nebenbedingungen, die durch lineare Gleichungen und Ungleichungen beschrieben sind, optimiert. Dabei werden die städtebaulichen Ziele als lineare Zielfunktionen, die gegebenen Einschränkungen der Städteplanung als Menge von Nebenbedingungen und die Anzahl der bebaubaren Teilbereiche der Stadtfläche als Problemraum beschrieben. Da es sich hierbei um ein multikriterielles Optimierungsproblem handelt, wird die Menge der möglichen optimalen Lösungen als Pareto-Optimum bezeichnet und umfasst die optimierten Zuteilungen der verschiedenen Arten der Bodennutzung im Stadtbereich hinsichtlich der gewichteten Zielkriterien.

Weiterhin existieren für die multidisziplinäre Entwurfsoptimierung bei der Planung der Bodennutzung heuristische Optimierungsverfahren, die unter anderem mit genetischen Algorithmen arbeiten (vgl. [BTBD99], [CHX09] und [CBH+11]). Diese Form der Entwurfsoptimierung wurde, in Anlehnung an die biologische Evolution, ursprünglich im Kontext der Städteplanung für die Optimierung von Transportwegen in der Großstadt entwickelt und auf die Planung der Bodennutzung übertragen (vgl. [BTBD99]). Dabei wird bei dieser Form der Optimierung die Stadtfläche in gleich große Zellen mit bestimmten Bodennutzungstypen unterteilt. Anschließend werden mögliche Kombinationen der Zellen und damit Varianten des Stadtplans als Startpopulation erzeugt. Diese werden dann anhand einer sogenannten Fitnessfunktion bewertet, die das zu lösende Optimierungsproblem beschreibt. Die Fitnessfunktion setzt sich aus verschieden gewichteten Zielkriterien zusammen und beschreibt dabei die Ziele der Entwurfsoptimierung im Kontext der Stadtplanung, welche unter anderem die Minimierung von Verkehrsstaus, Luftverschmutzung, Kriminalität und die Maximierung von Wohnqualität und wirtschaftlicher Entwicklung der Stadt sind. Diejenigen Lösungen, die die besten Fitness-Werte aufweisen, werden zufällig durch bestimmte Operationen, wie Mutation und Rekombination verändert, während die restlichen verworfen werden. Die neu gewonnenen Entwurfslösungen, welche neue Kombinationen der einzelnen Zellen des Stadtplans darstellen, werden dann wiederum bewertet und der Kreislauf setzt sich fort, bis eine optimale Lösung gefunden wird. Dieses Vorgehen wird nun für die einzelnen gewichteten Zielkriterien, welche für sich Teilprobleme bilden, wiederholt bis eine Lösungsmenge, das sogenannte Pareto-Optimum, gefunden wird. Der Stadtplaner entwirft nun anhand dieser Menge einen möglichst optimalen Stadtplan hinsichtlich der gewichteten Zielkriterien.

Neben dem Einsatz von genetischen Algorithmen werden ebenfalls Optimierungsverfahren auf Basis anderer metaheuristischer Algorithmen wie zum Beispiel Simulated Annealing, Greedy Growing, Tabu Search und weitere für die multidisziplinäre Entwurfsoptimierung bei der Planung der Bodennutzung eingesetzt (vgl. [AEHS03]). Diese Verfahren sind in der Lage näherungsweise optimale Lösungen für große kombinatorische Probleme zu liefern.

Auf Basis der Automatentheorie der theoretischen Informatik existieren weitere heuristische Optimierungsverfahren für die multidisziplinäre Entwurfsoptimierung bei der Planung der Bodennutzung (vgl. [WMP03] und [Wu98]). Hierbei werden ausgehend von der Vorhersage bzw. Simulation einer Stadtentwicklung optimale Stadtentwürfe abgeleitet. Anhand von Turing-Automaten und bestimmten Regeln, die im Kontext der Stadtplanung die Knotenübergänge des Automaten definieren, können generische Topologien einer Stadt erzeugt und damit ganze Stadtbereiche automatisch entworfen werden. Dies führt zur der Möglichkeit realistische Wachstumsmuster einer Stadt zu simulieren und daraus einen möglichst optimalen Entwurf abzuleiten.

Eine weitere Menge von Optimierungsmodellen für die multidisziplinäre Entwurfsoptimierung bei der Planung von Bodennutzung nutzt Algorithmen, die in der verteilten künstlichen Intelligenz Verwendung finden. Bei dem sogenannten „Particle Swarm Optimization"-Algorithmus, kurz PSO, wird das Prinzip der Schwarmintelligenz aufgegriffen und auf die multidisziplinäre Entwurfsoptimierung im Bereich der Städteplanung übertragen (vgl. [MMH12] und [SJFY11]). Zuerst wird dabei der Stadtbereich in gleich große Zellen mit einem bestimmten Bodennutzungstyp unterteilt. Anschließend organisieren sich die Zellen anhand bestimmter, im Kontext der Stadtplanung gegebener Regeln und gewichteter Zielkriterien der Zielfunktion, neu und bilden einen generisch erzeugten Stadtentwurf. Dabei repräsentieren die definierten Zellen die einzelnen Agenten. Als Folge bewegt sich der „Schwarm" von Flächenzellen hin zum Optimum der Ziel- bzw. Fitnessfunktion. Dieses Vorgehen wird für die einzelnen gewichteten Zielkriterien, welche für sich Teilprobleme bilden, wiederholt bis eine Lösungsmenge, das sogenannte Pareto-Optimum, gefunden wird. Diese Lösungsmenge beinhaltet alle optimalen Stadtentwürfe bezüglich der gewichteten Ziele. Der Stadtplaner entwirft nun anhand dieser Menge einen möglichst optimalen Stadtplan. Der hauptsächliche Vorteil dieses Optimierungsverfahrens gegenüber einem, das zum Beispiel genetische Algorithmen verwendet, ist in Bezug auf die Umsetzung und Anwendung die höhere Flexibilität und Einfachheit.

3.1.3. Schnittstelle zwischen GIS und CAD

Ausgehend von einem Geoinformationssystem zur Erfassung, Bearbeitung, Organisation, Analyse und Präsentation räumlicher Daten ist es für Stadtplaner von Vorteil, nicht nur die Metadaten einer Stadt, sondern auch die Geometrie und Topologie der Stadtstruktur visuell bewerten zu können. Da das GIS die Funktionalität einer interaktiven Karte mit sich bringt, sind die Funktionen zum Design eines Stadtentwurfs begrenzt. Als sehr gute Alternative eignen sich hier gängige CAD-Systeme, die sehr umfangreiche und mächtige Entwicklungswerkzeuge zur Generierung von Geometrie und Topologie mit sich bringen. Damit sind CAD-Systeme im weiten Sinne sehr gute Designwerkzeuge für die Stadtplanung.

Im Bereich der Stadtplanung ist die Verbindung von Geoinformationssystemen mit gängigen CAD-Systemen ein wichtiges Ziel, das dem Architekten erlaubt den Designprozess einer Stadtplanung ohne Umwege direkt auf den Metadaten des GIS auszuführen. Um dieses Ziel zu erreichen, werden unter anderem sogenannte „urban and design patterns“, also Designmuster für den Entwurf einer Stadt, verwendet (vgl. [BDS08], [DB11] und [DRS07]). Mit diesen Mustern ist es möglich ganze Stadtstrukturen auf den Daten eines GIS zu erzeugen. Bei der Erzeugung der Stadtstrukturen werden „shape grammars“, also sogenannte Formgrammatiken benutzt um die Erzeugungsregeln eines Designs zu beschreiben. Aufbauend darauf wird ein Erzeugungsmodell für das Design einer Stadt entworfen, das die Stadtstrukturen beschreibt und einer Stadtontologie zu Grunde liegt. Diese Ontologie stellt im Kontext der Stadtplanung ein Netzwerk von Informationen mit logischen Relationen dar und beschreibt den Zusammenhang zwischen GIS-Metadaten und den Stadtstrukturen im Designprozess der Stadtplanung.

Um nun die Schnittstelle zwischen GIS und CAD-System entwerfen zu können, muss die Ontologie der Stadtstruktur innerhalb eines GIS hinsichtlich ihrer Designfunktionen, anstelle der vorhandenen Analyseeingenschaften, verfeinert werden (vgl. [BDS08]). Dadurch wird es möglich komplexe Designs bzw. Stadtentwürfe oder sogenannte Masterpläne mit Hilfe eines definierten Designmusters generisch zu erstellen. Wie in Abbildung 5.2 zu sehen, sind dabei die Entwürfe die Ergebnisse der Anwendung einer urbanen Designsprache, welche durch Formgrammatiken definiert wird und von einer Menge von Entwurfsmustern abhängt. Die Entwurfsmuster wiederum erhalten vorhandene GIS-Metadaten in bestimmter Formatierung als Eingabedaten. Der Stadtplaner hat nun eine Menge von GIS-Metadaten, die den Kontext zur Entwicklung eines Entwurfs darstellen. Wie Abbildung 3.3 zeigt, bilden dann generisch erzeugte Stadtplanentwürfe durch die Anwendung der Erzeugungsregeln das Resultat in einem CAD-System. Die Erzeugungsregeln werden dabei auf einer Ontologie von Stadtstrukturelementen mit bestimmten Attributen angewandt.

Mit Hilfe von bestimmten Verfahren, die diese formale Designsprache zur generischen Erzeugung von Stadtstrukturen mit samt ihrer Meta-Eigenschaften verwenden können, ist es dann möglich mit gängigen Optimierungsframeworks multidisziplinäre Entwurfsoptimierung in der Städteplanung durchzuführen. Eines dieser Verfahren ist das sogenannte *Shape-Rewriting*, welches an Graphersetzungssysteme angelehnt ist und in Kapitel 5 erläutert wird.

3.2. Verwandte Arbeiten und Projekte

Die kombinierte Verwendung und Integration gängiger Technologien und computergestützter Werkzeuge innerhalb eines Softwaresystems im Bereich der Stadtplanung ist keine neuartige Idee. Die theoretischen und praktischen Grundlagen hierfür wurden bereits in früheren Arbeiten gelegt. Das Spektrum in diesem Bereich reicht von reinen Spezifikationen bis hin zu voll einsatzfähigen Lösungen. Bekannte, als auch weniger bekannte Ansätze und Arbeiten, sollen im Folgenden vorgestellt werden.

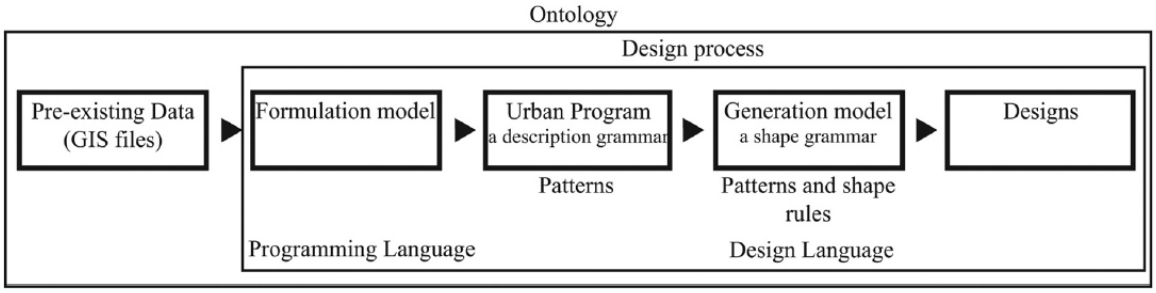

Abbildung 3.2.: Zusammenhang von Ontologie, Designsprache, Entwurfsmuster, Formgrammatiken und Erzeugungsregeln (vgl. [BDS08])

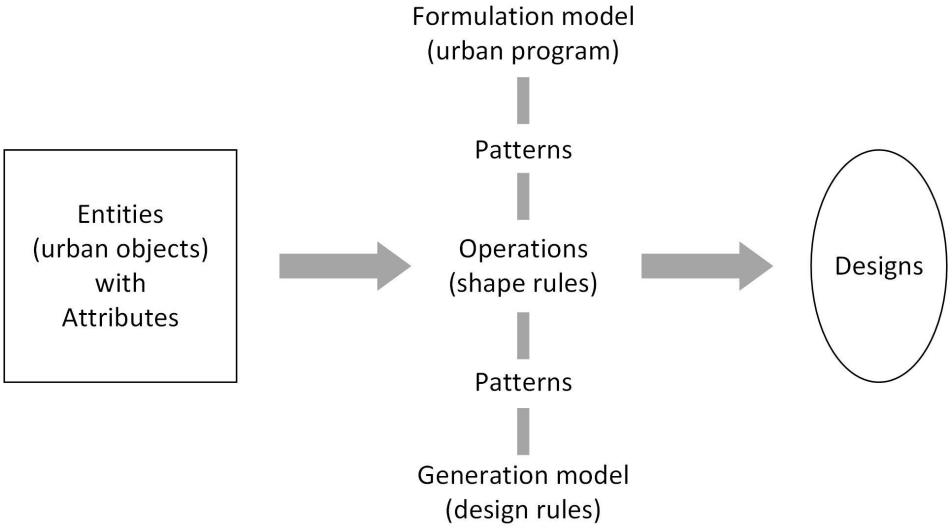

Abbildung 3.3.: Entwürfe resultieren aus der Anwendung der Erzeugungsregeln auf einer Ontologie von Stadtstrukturelementen mit bestimmten Attributen (vgl. [BDS08])

3.2.1. Esri CityEngine

CityEngine[2] ist eine spezialisierte kommerzielle Erweiterung des GIS ArcGIS[3] der US-amerikanischen Firma Esri[4]. Die Funktionalität der Software reicht von 3D-Modellierung urbaner Gebiete bis hin zu prozeduraler Modellierung und Erzeugung von Stadtplänen bzw. Stadtgebieten. CityEngine ist in die Klasse der 3D-GIS einzuordnen.

[2]siehe `http://www.esri.com/software/cityengine`
[3]siehe `http://www.esri.com/software/arcgis`
[4]siehe `http://www.esri.com`

CityEngine biete einige der fortschrittlichsten Techniken zur prozeduralen Erzeugung von dreidimensionalen Inhalten für die Architektur und Stadtplanung auf der Grundlage von erweiterten L-Systemen (vgl. [MP01]). Zur Modellierung der Inhalte werde hierbei sogenannte „Computer Generated Architecture"-Regeln, kurz CGA-Regeln, verwendet. Diese Regeln unterliegen einer CGA-Formgrammatik und erzeugen bzw. verfeinern 3D-Modelle einer Stadtstruktur iterativ. Das verwendete Regelsystem kann vom Nutzer nach bestimmten Kriterien abgeändert und erweitert werden, sodass Raum für neue Designmöglichkeiten entsteht. Weiterhin können mit CityEngine Straßennetze prozedural generiert und Fassaden der 3D-Gebäude assistentengestützt entworfen werden. Ebenfalls ist es mit CityEngine möglich 2D-Daten eines GIS in 3D-Modelle zu transformieren. Es werden unter anderem das Industriestandartformat für vektorbasierte Geodaten, das sogenannte Shape-Format, sowie der Zugriff auf OpenStreetMap unterstützt. Zudem bietet CityEngine eine umfassende Unterstützung für 3D-Formate aus dem Bereich 3D-Modellierung und -Konstruktion. Eine Optimierung der erzeugten Inhalte auf Basis von Geodaten bietet CityEngine zum Zeitpunkt der Erstellung dieser Diplomarbeit nicht an.

3.2.2. CityGML

Die City Geography Markup Language (CityGML[5]) ist ein häufig verwendetes Beispiel einer Ontologie, die die Geometrie, Semantik und Topologie zwischen einzelnen Stadtstrukturen in einem 3D-Stadtmodell speichert. CityGML basiert auf der Auszeichnungssprache GML, welche wiederum auf der Sprache XML aufbaut, und ist ein Anwendungsschema zur Speicherung und zum Austausch von 3D-Stadtmodellen. Bei der Modellierung von Stadtobjekten setzt CityGML unter anderem auf Standards des Open Geospatial Consortium, das mit OGC[6] abgekürtzt wird.

Der Verwendungszweck von CityGML geht weit über die Speicherung und den Austausch von 3D-Stadtmodellen hinaus. So macht es CityGML möglich 3D-Stadtmodelle für eine detaillierte Analyse und Simulation zu verwenden. Außerdem wird durch die Anwendung der Sprache das sogenannte Urban Data Mining, bei dem die Operationalisierung der Strukturerkennung und Strukturbildung von Ähnlichkeitsmustern in einem urbanen Gebiet im Vordergrund steht, möglich. Dementsprechend kommt CityGML auch häufig innerhalb von Geoinformationssystemen zum Einsatz (vgl. [BDS08]).

[5] siehe `http://www.citygml.org`
[6] siehe `http://www.opengeospatial.org`

4. Architektur und Konzeption

Dieses Kapitel befasst sich mit den Details des Entwurfs und der Beschreibung der konzipierten Architektur der geplanten Applikation *CADtoGIS Schnittstelle*. Zuerst werden die genauen Anforderungen dieser Aufgabe definiert, bevor im nächsten Schritt das entwickelte Lösungskonzept und die einzelnen Komponenten der Architektur untersucht werden können. Diese umfassen neben dem logischen Datenmodell und der Optimierungskomponente verschiedene Komponenten zur Beschaffung und Verarbeitung von raumbezogenen Geodaten sowie Komponenten zur Erzeugung und Darstellung von CAD-Stadtmodellen.

4.1. Konzeption des Lösungsansatzes

Um die Entscheidungen bezüglich der getroffenen Modellierung detailliert begründen zu können, ist eine geeignete Einordnung der Arbeit in den Gesamtzusammenhang von Vorteil. Zu diesem Zweck werden im Folgenden alle Kopplungen der *CADtoGIS Schnittstelle* zum bestehenden Umfeld beschrieben. Zudem muss eine bestimmte Menge aller gegebenen Anforderungen diskutiert werden.

4.1.1. Anforderungen

Eine Software bzw. Applikation muss im Allgemeinen gewisse Mindestvoraussetzungen erfüllen, um sinnvoll eingesetzt werden zu können. Diese umfassen neben der Änderbarkeit und Robustheit unter anderem auch die Belastbarkeit und Zuverlässigkeit des Systems. Der im Umfang dieser Diplomarbeit zu implementierende Softwareprototyp, welcher eine Schnittstelle zwischen bereits existierenden Softwaresystemen realisiert, muss zudem die Voraussetzung der Austauschbarkeit von Programmkomponenten erfüllen.

Die folgende Liste von Anforderungen ist von allgemeiner Natur und bezieht sich unter anderem auf die Mindestvoraussetzungen von Software im Allgemeinen.

- **Änderbarkeit:** Damit die entwickelte Software zu einem späteren Zeitpunkt erweitert werden kann, soll schon bei der Planung und Implementierung der Software auf diesen Aspekt geachtet werden. Der Quellcode soll richtig und sinnvoll strukturiert, sowie einfach gestaltet sein. Außerdem soll der Quellcode keine schwer verständlichen Konstrukte beinhalten. Besonderer Wert soll auf die Dokumentation des Codes gelegt werden, damit spätere Erweiterung und Wartung der Software keine Probleme mit nicht dokumentierten Schnittstellen bringt.

- **Robustheit:** Da die Software in der Praxis von Menschen eingesetzt wird, die diese Software nicht selbst entwickelt haben, ist es unvermeidlich, dass ein Benutzer die Applikation falsch verwendet und fehlerhafte Eingaben durchführt. Daher soll die Software fehlerhafte Eingaben abfangen und entsprechend darauf reagieren. Die Applikation sollte in so einem Fall nicht abstürzen.
- **Portabilität:** Aufgrund der eingesetzten Programmiersprache Java ist ein plattformunabhängiger Betrieb der Software möglich. Das Betriebssystem Microsoft Windows 7 soll auf jeden Fall unterstützt werden.
- **Zuverlässigkeit:** Die Software soll für den Benutzer eine zuverlässige Unterstützung seiner Arbeit sein. Daher soll die Software korrekt und genau funktionieren. Zu beachten ist das es sich bei der Applikation um eine Form der Machbarkeitsstudie handelt. Es soll auf eine gewisse Ausfallsicherheit geachtet werden.
- **Bedienbarkeit (Usability):** Bei dieser Software soll auf eine gute Bedienbarkeit geachtet werden. Dies bezieht sich nicht nur auf die direkte Benutzungsoberfläche, sondern auch auf funktionale Möglichkeiten bestimmte Prozesse auszuführen. Diese Möglichkeiten werden durch Usability Patterns, die im Kapitel 4.7 in die Use-Cases mit eingebunden werden, beschrieben und sollen in dieser Software schon in der frühen Planungsphase ihren Platz finden.

Auf Grundlage der Anforderungen an Software im Allgemeinen werden in der folgenden Liste die wichtigsten Anforderungen an den Prototypen *CADtoGIS Schnittstelle* detaillierter beschrieben.

- **Austauschbarkeit der Programmkomponenten:** Bei der Applikation *CADtoGIS Schnittstelle* sollen die externen Programmkomponenten zur Anzeige, Bezug, Verarbeitung und Optimierung von Geodaten und CAD-Daten optional ausgetauscht bzw. ersetzt werden können. Dazu ist ein modularer Aufbau der Software vorgesehen.
- **Bezug und Anzeige von Geoinformationen:** Die Applikation soll Geoinformationen aus bestimmten Quellen beziehen, verarbeiten und darstellen können. Die Darstellung soll auf einer Kartenansicht erfolgen.
- **Erzeugung eines CAD-Stadtmodells aus den Geodaten:** Die Applikation soll eine automatische Erzeugung eines CAD-Stadtmodells anhand der bezogenen Geodaten bzw. Geoinformationen bereitstellen.
- **Verknüpfung von Geodaten und CAD-Stadtmodell:** Die Applikation soll eine konsistente Kopplung der vorliegenden Geodaten mit dem erzeugten CAD-Stadtmodell ermöglichen.
- **Optimierung von CAD-Modell und Geodaten:** Die Applikation soll unter anderem die Einspeisung der erzeugten Daten in ein Optimierungsframework gewährleisten. Außerdem soll die Optimierung des CAD-Modells anhand der bezogenen Geodaten und anhand von festgelegten Zielfunktionen erfolgen. Insofern sollen beide Daten aufgrund der Kopplung von Geodaten und CAD-Modell optimiert werden.

Die hier diskutierte Anforderungsmenge stellt in erster Linie einen konzeptionellen Rahmen dar, auf dem im Verlauf der Arbeit aufgebaut werden soll. Die genauen Details der Anforderungsumsetzung werden in Kapitel 6 besprochen.

4.1.2. Makroskopische Einordnung

Um die multidisziplinäre Entwurfsoptimierung bei der Planung eines städtischen Teilgebiets mit dem interaktiven Entwurfs- bzw. Designprozess eines CAD-Systems unter Hinzunahme von raumbezogenen Metainformationen zu verbinden, wird eine Schnittstelle für die Kommunikation zwischen CAD-System, Optimierungsframework und einer Geoinformationsquelle entworfen. Diese Geoinformationsquelle kann unter anderem ein GIS oder auch ein Geoserver sein, welcher Geodaten bereitstellt und verarbeitet. Der Fokus liegt dabei unter anderem auf der Entwicklung eines Datenmodells zur Abbildung von CAD-Geometrie unter Verwendung von raumbezogenen Metainformationen, wie zum Beispiel der Bodennutzung, sowie auf der Erzeugung neuer CAD-Stadtgeometrie in Abhängigkeit der Zielvorgaben des Städtebauarchitekten. Die Erzeugung einer neuen CAD-Stadtgeometrie und die multidisziplinäre Entwurfsoptimierung hängen dabei stark von einander ab. Dabei muss zuerst eine formale Designsprache mit Entwurfsmustern, Erzeugungsregeln und Formgrammatiken für die Geometriedaten eines CAD-Stadtmodells und den raumbezogenen Meta-Daten der Stadt und ihrer Umgebung entworfen werden.

Die Abbildung 4.1 zeigt das Lösungsmodell, welches dem konzipierten Lösungsansatz unterliegt. Ausgehend von einem initialen CAD-Modell, das raumbezogene Meta-Eigenschaften aus einem GIS oder einer anderen Geoinformationsquelle besitzt und durch den Einsatz einer Designsprache entworfen worden ist, wird eine multidisziplinäre Entwurfsoptimierung angestoßen. Die multidisziplinäre Entwurfsoptimierung findet dabei innerhalb eines Optimierungsframeworks wie zum Beispiel Opt4J[1] statt. Dabei werden alle vorhandenen Daten des Stadtmodells dem Optimierungsframework übergeben und anhand eines Optimierungsmodells, das sich direkt auf die Designsprache bezieht, Varianten des initialen CAD-Stadtmodells gebildet. Die einzelnen Varianten können daraufhin optional ein Simulationssystem durchlaufen und werden anschließend auf Grundlage der Simulationsergebnisse, interaktiv vom Stadtplaner oder automatisch vom Optimierungsframework hinsichtlich der Planungsziele bewertet. Bei einem nicht zufriedenstellenden Ergebnis der Bewertung wird das Optimierungsmodell hinsichtlich der Erzeugungsregeln abgeändert, sodass bei einem erneuten Optimierungsdurchlauf neue Varianten des initialen CAD-Entwurfsmodells entstehen. Die Änderung bzw. Anpassung des Optimierungsmodells hinsichtlich der Erzeugungsregeln geschieht dabei mit Hilfe eines sogenannten Graphersetzungssystems (engl. *Shape-Rewriting*). Der Optimierungsprozess läuft so lange bis die festgelegten Planungsziele zufriedenstellend erfüllt sind und ein optimiertes Ergebnis gefunden wird. Das optimierte Ergebnis besteht dabei aus einem CAD-Modell, das in einem gängigen CAD-System visuell bewertet werden kann und einer Menge von raumbezogener Metadaten, die in Abhängigkeit des CAD-Modells in einem GIS bewertet und analysiert werden können.

[1] siehe `http://opt4j.sourceforge.net/index.html`

Außerdem zeigt Abbildung 4.1, dass das Lösungsmodell sich in drei Bereiche einteilen lässt. Im oberen und unteren Teil der Abbildung befinden sich die für den Städtebauarchitekten interessanten Daten und Abläufe. Der mittlere Teil stellt dagegen den technischen Teil dar, welcher zur Lösung des praktischen Problems der Stadtplanung beiträgt. Ein ausführliche Beschreibung der Konzeption des technischen Teils ist in Kapitel 5 vorzufinden.

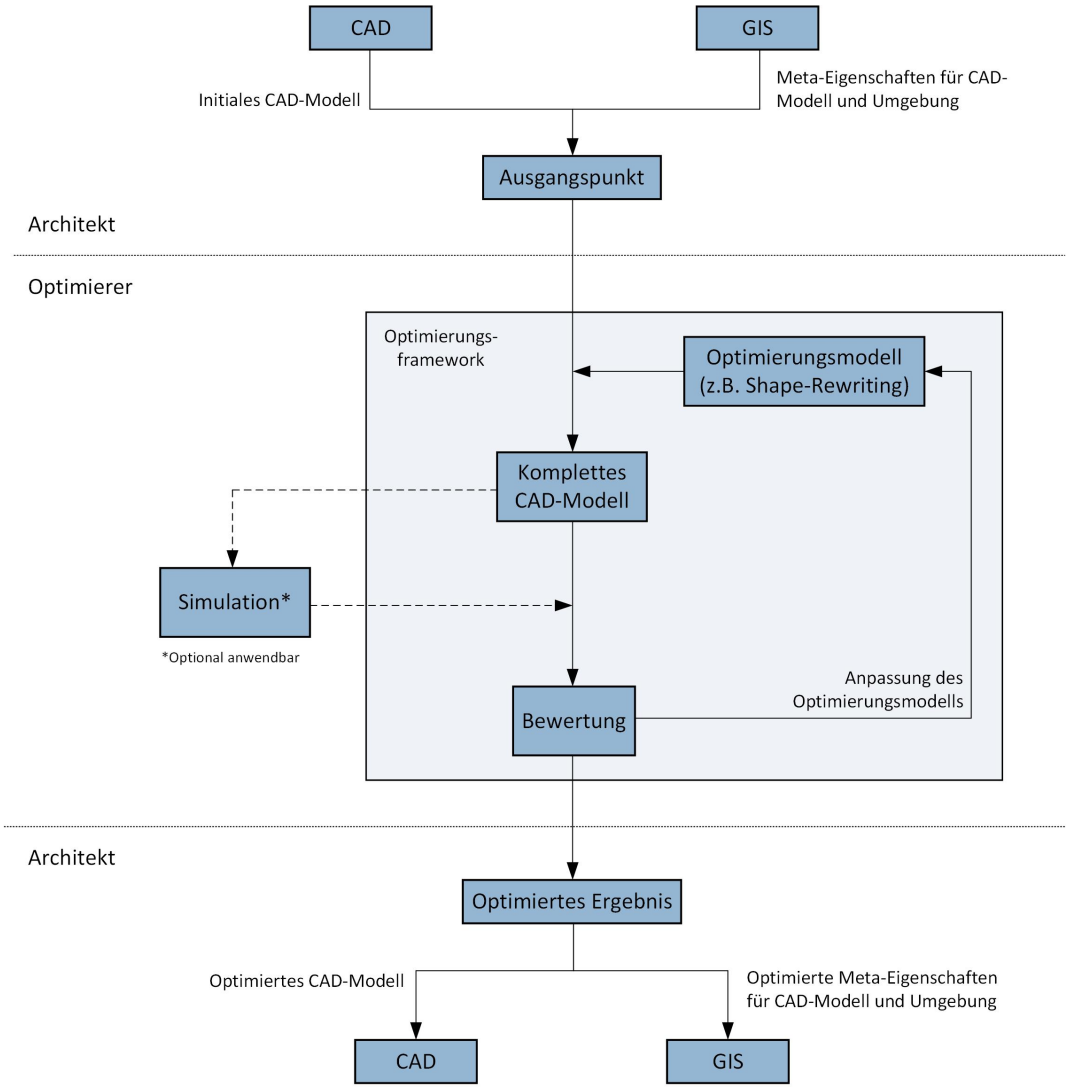

Abbildung 4.1.: Das Lösungsmodell der Applikation *CADtoGIS Schnittstelle*

4.2. Architektur

Der vorherige Abschnitt 4.1 bot einen Überblick über die Umgebung der Applikation *CADtoGIS Schnittstelle* sowie eine detaillierte Betrachtung des Lösungsansatzes. Als nächstes wird die eigentliche Architektur der *CADtoGIS Schnittstelle* betrachtet. Dazu sieht man auf Abbildung 4.2 die konzeptionelle Architektur der Applikation.

CADtoGIS Schnittstelle ist eine eigenständige Java Applikation, welche die Grafikbibliothek Swing zur Erstellung der Benutzeroberfläche nutzt. *CADtoGIS Schnittstelle* ist die Hauptkomponente der Applikation und bietet die geforderte Funktionalität hinsichtlich der Anforderungen. Die Entwurfsrichtung wurde „Bottom-Up“ gewählt, da dadurch zeitnah mit der Programmierung eines Prototyps begonnen werden kann und dies den Anforderungen entspricht. Bei diesem Basis-Ansatz sollen zunächst einzelne Softwarebestandteile z. B. Funktionen, Klassen, Module definiert und eventuell sogar direkt geschrieben werden. Aus diesen Teilen werden dann größere Elemente der Software zusammengesetzt, bis das vollständige System erstellt ist.

Als Architekturmuster wird das Model-View-Controller-Architekturmuster, das ein bewährtes und oft eingesetztes Muster ist, verwendet. Logisch wird die Applikation daher in das *Datenmodell* (engl. Model), die *Oberfläche* (engl. View) und die *Steuereinheit* (engl. Controller) unterteilt. Dabei wird der Großteil aller ausführbaren Aktionen an die Steuereinheit weitergeleitet und von dieser ausgeführt. Wie der Name schon sagt, stellt die Komponente *Model* das benötigte Datenmodell zur Abbildung und Speicherung der verwendeten Daten bereit. Die Komponente *View* dient dazu die Benutzeroberfläche der Applikation zu realisieren sowie alle vom Benutzer kommenden Eingaben entgegenzunehmen und an die Komponente *Controller* zu leiten. Die Komponente *Controller* implementiert die Logik für die Umsetzung des Lösungsansatzes.

Model

Die Komponente Model, welche im Projekt *CADtoGIS Schnittstelle* als *DataModel* bezeichnet wird, definiert das nötige urbane Objekt, um auf Stadtformen und -strukturen, welche durch Geometrien in Verbindung mit Geodaten beschrieben werden, zugreifen zu können. Insofern beschreibt diese Komponente die Dekomposition von einfachen Stadtstrukturen, wie Straßen, Gebäuden und Gebäudeblöcken und ist objektorientiert entworfen. Der Abschnitt 4.4 behandelt dieses Thema ausführlich.

View

Die Komponente View, welche im Projekt *CADtoGIS Schnittstelle* als *UserInterface* bezeichnet wird, dient zur Realisierung der Benutzeroberfläche. Nach dem MVC-Architekturmuster besteht eine strikte Trennung zwischen Benutzereingaben und Ausführung der programminternen Logik. Die Komponente *UserInterface* nimmt nur Benutzereingaben entgegen und führt keine Logik aus. Im Abschnitt 4.5 wird dieses Thema ausführlicher betrachtet.

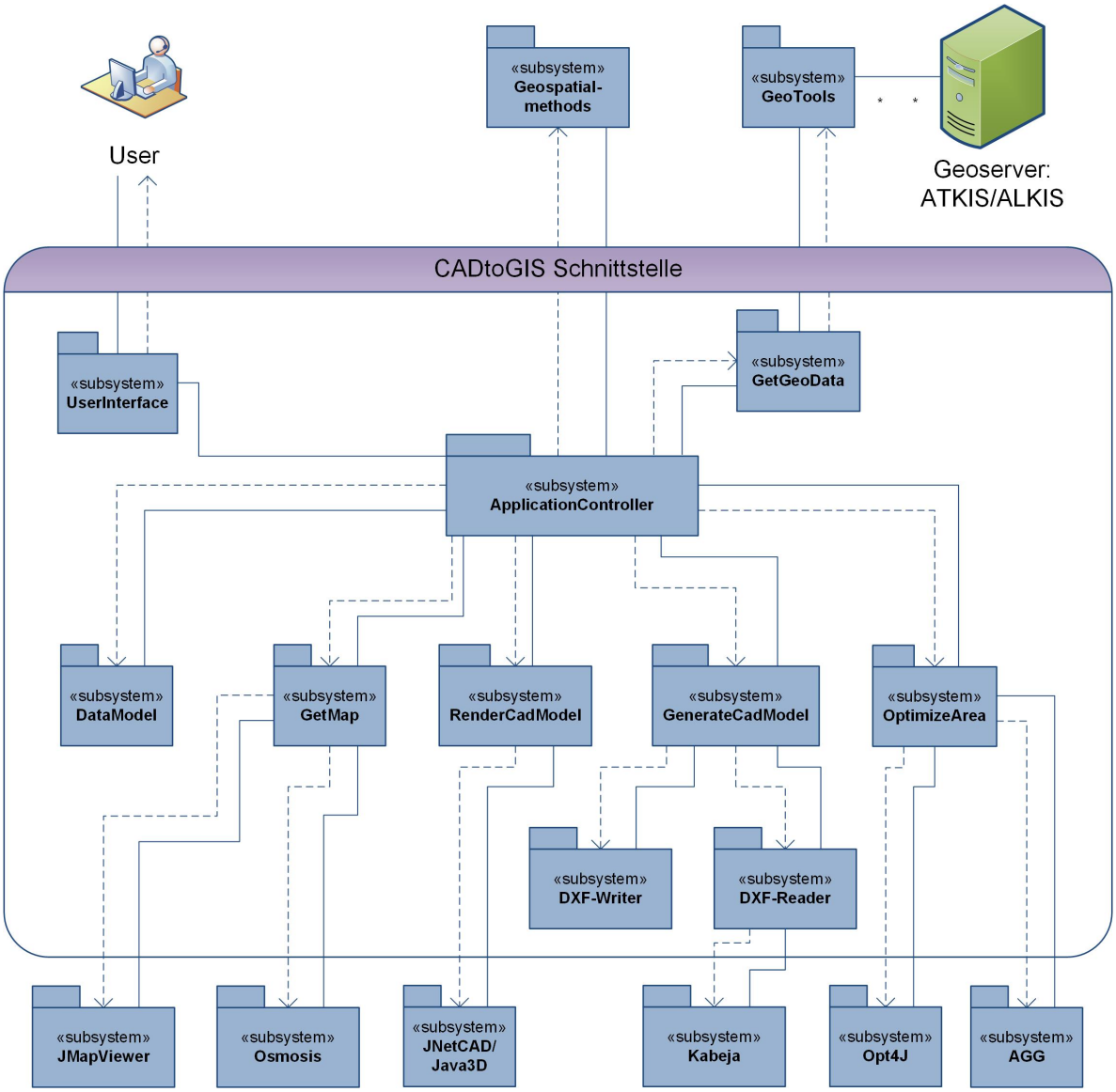

Abbildung 4.2.: Die konzeptionelle Architektur der *CADtoGIS Schnittstelle*

Controller

Die Komponente Controller, welche im Projekt *CADtoGIS Schnittstelle* als *ApplicationController* bezeichnet wird, dient als allgemeine Steuerung der Applikation und stellt die Verbindung zwischen User-Interface, Datenmodell und externen Komponenten her, sodass diese voneinander getrennt sind. Die Komponente *ApplicationController* dient der Erzeugung der benötigten urbanen Objekte, welche aus dem Datenmodell hervorgehen. Außerdem werden hier die Er-

zeugung des CAD-Stadtmodells sowie die Optimierung gestartet. Der Abschnitt 4.6 diskutiert die Komponente *ApplicationController* detaillierter.

Externe Komponenten

Die Applikation bzw. die Hauptkomponente *CADtoGIS Schnittstelle* verwendet zahlreiche externe Komponenten. Diese nutzen freie externe Bibliotheken wie OpenStreetMap-API/JMapViewer, Osmosis, Kabeja, JNetCAD/Java3D, GeoTools, Geospatialmethods, Opt4J und AGG. Alle externen Komponenten werden an die interne Komponente *Application-Controller* angebunden und von dieser verwendet. Die allgemeine Kopplung der externen Komponenten mit der Applikation *CADtoGIS Schnittstelle* sowie die Interaktion mit der Komponente internen *ApplicationController* werden in den Abschnitten 4.3 sowie 4.6 detailliert beschrieben.

4.2.1. Integration der Komponenten

Alle internen Komponenten sind in der Hauptkomponente *CADtoGIS Schnittstelle* und somit in der eigentlichen Applikation integriert. Externe Komponenten für den Bezug und die Verarbeitung von Geodaten, zur Erzeugung und Darstellung eines CAD-Stadtmodells sowie zur Durchführung der multidisziplinäre Entwurfsoptimierung werden an die interne Komponente *ApplicationController* und damit an die Applikation angebunden. Für die Bearbeitung eines erzeugten CAD-Stadtmodells kann optional eine externe gängige CAD-Anwendung verwendet werden. Die genauen Details der Integration werden in Kapitel 4.6 sowie in Kapitel 6 diskutiert.

4.3. Hauptkomponente der Applikation

Die Hauptkomponente *CADtoGIS Schnittstelle* stellt die eigentliche Java-Applikation *CADtoGIS Schnittstelle* dar und bietet eine Benutzeroberfläche zur Auswahl eines urbanen Zielbereichs mit einem bestimmtem Optimierungspotential innerhalb einer Kartenansicht. Zu dem gewählten Zielbereich, kann innerhalb der Hauptkomponente ein CAD-Modell des Stadtbereichs erstellt werden. Außerdem können abhängig vom gewählten Bereich die verfügbaren Geodaten eingeblendet sowie die eigentliche Entwurfsoptimierung gestartet werden.

Im Detail werden alle Eingaben vom User-Interface der Applikation an die interne Komponente *ApplicationController* übergeben und dort verarbeitet. Mit Hilfe der externen Bibliothek JMapViewer[2], die die externe Komponente *JMapViewer* darstellt, wird die OpenStreetMap[3]-API angesprochen und entsprechend der gewünschte Zielbereich als Kartenansicht dargestellt.

[2]siehe `http://wiki.openstreetmap.org/wiki/JMapViewer`
[3]siehe `http://www.openstreetmap.de`

Die externe Komponente *Osmosis*[4] verwendet die gleichnamige freie Bibliothek und dient der Extrahierung von geografischen Daten aus lokalen OpentStreetMap-Datenbank-Dateien. Diese Vorgehensweise bei der Beschaffung raumbezogener Daten genügt dem Charakter eines Prototypen. Die externe Komponente *Geospatialmethods* nutzt die freie Bibliothek Geospatialmethods[5], welche eine Menge von Methoden für geografische Berechnungen bietet. Diese werden für die Auswertung raumbezogener Daten in der internen Komponente *ApplicationController* verwendet. Die externe Bibliothek GeoTools[6] wird dementsprechend in der externen Komponente *GeoTools* verwendet, um geografische Daten aus den sogenannten AFIS-ALKIS-ATKIS[7]-Katalogen zu beziehen. Diese Daten werden lokal verarbeitet, was ebenfalls dem Charakter eines Prototypen genügen soll. Die Kopplung der Hauptkomponente an einen Web-Service eines Geoservers, welcher die selben geografischen Daten bereitstellt, ist ebenfalls ohne weiteres mit Hilfe der externen Komponente *GeoTools* möglich. In Hinblick auf das zentrale Ziel der Schnittstellenentwicklung wurde diese Funktionalität aber zunächst zurückgestellt. Die geografischen Daten aus OpenStreetMap und AFIS-ALKIS-ATKIS werden im gemeinsamen Datenmodell der Hauptkomponente *CADtoGIS Schnittstelle* zusammengefügt (siehe dazu Kapitel 4.4). Durch Verwendung der externen Komponente *JNetCAD*, die die freie Bibliothek JNetCAD[8] beinhaltet, ist es anschließend möglich das aus dem Datenmodell generierte CAD-Stadtmodell darzustellen. Das entsprechende CAD-Stadtmodell wird mit Hilfe der internen Komponente *DXF-Writer* sowie der externen Komponente *Kabeja*, die die freie Bibliothek Kabeja[9] als DXF-Reader verwendet, erzeugt.

Die externen Komponenten *Opt4J* und *AGG* verwenden die entsprechenden gleichnamigen freien Bibliotheken und werden zur multikriteriellen Entwurfsoptimierung des urbanen Zielbereichs eingesetzt. Dabei wird mit Hilfe von Opt4J[10] ein Optimierungsframework entworfen. Dieses Optimierungsframework arbeitet auf Erzeugungsgraphen für urbane Stadtstrukturen, welche mit Hilfe der Bibliothek AGG[11] implementiert werden. Eine ausführliche Erklärung der theoretischen Konzeption des Optimierungsframeworks sowie dessen Umsetzung ist in Kapitel 5 und Kapitel 6 vorzufinden.

4.3.1. Speicherung von Daten

Die meisten Projektdaten werden anwendungsintern in urbanen Datenobjekten, die durch das logische Datenmodell definiert werden, gespeichert. Von externen Quellen bezogenen und in der Applikation erzeugte Geodaten und CAD-Daten werden in lokalen Esri-Shape- bzw. DXF-Dateien gespeichert und können anschließend weiterverarbeitet werden.

[4] siehe `http://wiki.openstreetmap.org/wiki/DE:Osmosis`
[5] siehe `http://geospatialmethods.org`
[6] siehe `http://geotools.org`
[7] siehe `http://www.adv-online.de/AAA-Modell`
[8] siehe `http://www.johannes-raida.de/jnetcad.htm`
[9] siehe `http://kabeja.sourceforge.net`
[10] siehe `http://opt4j.sourceforge.net`
[11] siehe `http://user.cs.tu-berlin.de/gragra/agg`

4.4. Logisches Datenmodell

In der internen Komponente *DataModel* werden die nötigen Objekte definiert, um mit Geometrien in Verbindung mit Geodaten arbeiten zu können. Diese Objekte werden von dem logischen Datenmodell der *CADtoGIS-Schnittstelle*, welches zum einem Teil auf der Arbeit von [BDS08] basiert, definiert. Der Ansatz für den Entwurf des Datenmodells beruht auf der Grundlage einer Dekomposition von vorhandenen Stadtstrukturen. Dabei wird zuerst versucht einen Stadtplan auf möglichst einfache Stadtstrukturen, wie zum Beispiel Straßen, Gebäude, Nutzflächen und Gebäudeblöcke herunter zu brechen, um diese so genau wie möglich beschreiben und definieren zu können. Anschließend wird durch das erneute Kombinieren der einfachen Stadtstrukturen die Erzeugung eines neuen Stadtplans mit Hilfe einer Designsprache und bestimmten Erzeugungsregeln ermöglicht (siehe dazu Kapitel 5).

Das zentrale Datenobjekt des logischen Datenmodells ist das Stadt-Objekt *UrbanObj*. Wie in Abbildung 4.3 zu sehen, beinhaltet dieses den Stadt-Objekt-Typ *ObjType* sowie das Nutzflächen-Objekt *Zone*, in dem sich das Stadt-Objekt befindet. Die Stadt-Objekt-Typen *Street* und *Building* stellen die wichtigsten Elemente des Datenmodells dar und erben die Eigenschaften des Stadt-Objekts *UrbanObj*. Sie beschreiben jeweils Straßen und Gebäude in ihrer geografischen Lage und Geometrie. Unter anderem definiert dabei das Stadt-Objekt *Street* die geografischen Punkte des Straßenverlaufs, alle anschließenden Seitenstraßen, die zugrunde liegende CAD-Geometrie sowie eine Reihe von Metainformationen wie zum Beispiel den Straßennamen und -typ. Das Stadt-Objekt *Building* beschreibt hingegen über eine Liste geografischer Punkte die Lage und Fläche des Gebäudes, die zugrunde liegende CAD-Geometrie wie auch eine Reihe von Metainformationen wie zum Beispiel den Gebäudenamen und -typ. Die Abbildung A.5 im Anhang A illustriert einen weiteren Stadt-Objekt-Typ mit der Bezeichnung *ObjRelation*, der zusätzliche Beziehungen zwischen den einzelnen Stadt-Objekten definiert.

Ein weiteres zentrales Element des Datenmodells bildet das Objekt *UrbanArea*. Wie die Abbildung A.6 im Anhang A darstellt, beinhaltet dieses Objekt alle anderen Objekte und definiert die Beschreibung des gesamten gewählten Stadtbereichs. Das Objekt *UrbanArea* beinhaltet dabei die Objekte *Bound*, *Block*, *StreetNetwork*. Wobei das Objekt *Bound* die Grenze des gewählten Stadtbereichs definiert. Das Objekt *Block* beschreibt einen Gebäudeblock innerhalb des gewählten Stadtbereichs und beinhaltet eine bestimmte Anzahl an Gebäudeobjekten. Der Gebäudeblock wird dabei von einer bestimmten Anzahl Straßen umschlossen. Das Objekt *StreetNetwork* beschreibt innerhalb des gewählten Stadtbereichs das gesamte Straßennetz.

Die geografische Lage und Fläche eines jeden Objekts wird durch die Klassen *LatLonPoint* und *LatLonLine* definiert. Die CAD-Geometrie eines jeden Objekts wird durch die Klassen *ObjGeometry* und *Coords2D* beschrieben. Wobei das Objekt *ObjGeometry* entweder den Geometrietyp „Polyline“ und „Polygon“ als Eigenschaft besitzen kann. Die Beziehung zwischen dem geografischen Punkt der raumbezogenen Daten und dem geometrischen Punkt der CAD-Daten wird über die Klasse *PointRelation* realisiert.

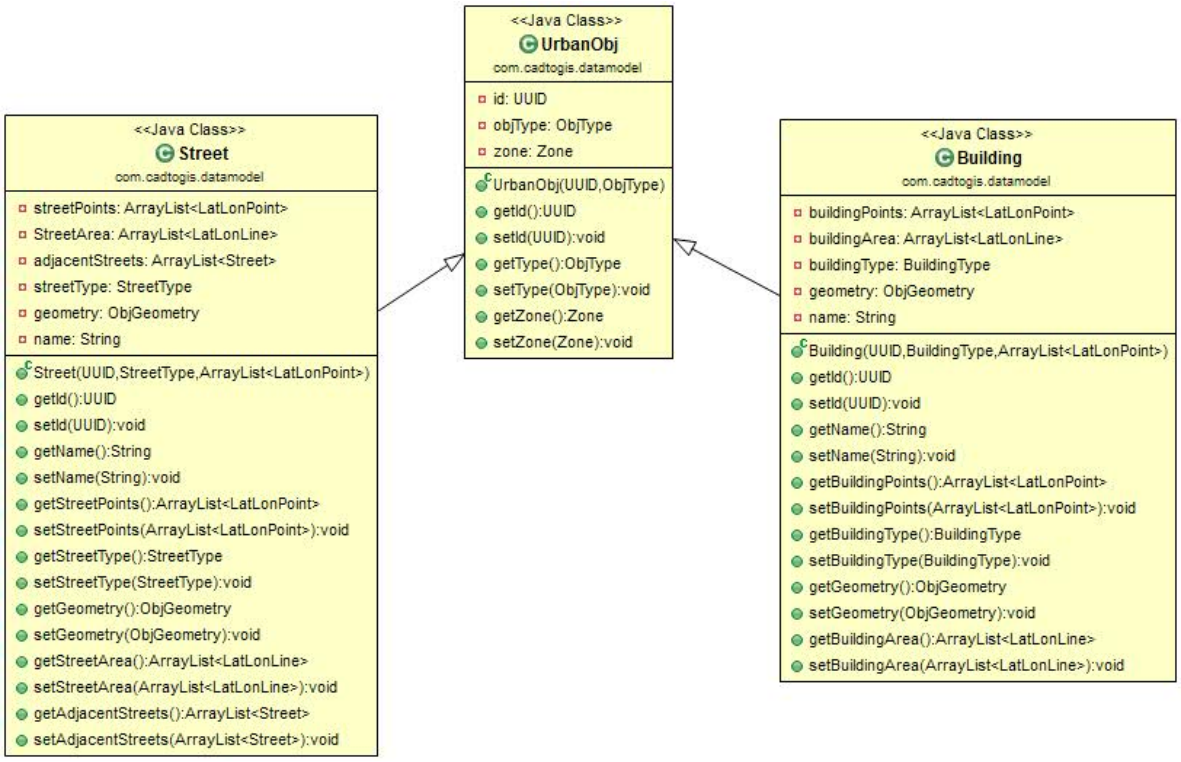

Abbildung 4.3.: Übersicht der drei wichtigsten Klassen des logischen Datenmodells

4.5. Grafische Benutzeroberfläche

Dieses Kapitel beschreibt die einzelnen Bestandteile der grafischen Benutzeroberfläche der Applikation *CADtoGIS Schnittstelle*. Die Benutzeroberfläche wird in der internen Komponente *UserInterface* realisiert und unterliegt dem MVC-Architekturmuster. Insofern besteht eine klare Trennung zwischen Benutzereingaben und Ausführung der programminternen Logik. In der Folge werden Eingaben in der grafische Benutzeroberfläche entgegengenommen und in der Komponente *ApplicationController* verarbeitet.

Die grafische Benutzeroberfläche lässt sich im Groben in drei Bereiche unterteilen. Der erste Bereich befasst sich mit der Anzeige und der Auswahl von raumbezogenen Daten innerhalb einer Kartenansicht. Dieser Bereich ist der Hauptbereich und dient als Einstieg in die Applikation. Aus diesem heraus werden die beiden anderen Bereiche der grafischen Benutzeroberfläche erreicht. Einer dieser Bereiche umfasst die Anzeige eines erzeugten CAD-Stadtmodells mit der Möglichkeit der lokalen Speicherung des angezeigten CAD-Stadtmodells. Eine interaktive Bearbeitung bzw. Anpassung des CAD-Stadtmodells ist optional mit Hilfe einer externen CAD-Anwendung möglich. Der dritte Bereich der grafischen Benutzeroberfläche ermöglicht die Durchführung der multidisziplinären Entwurfsoptimierung auf den raumbezogenen Geodaten sowie den Daten des CAD-Stadtmodells. Hinsichtlich der Optimierung werden hier bestimmte Zielfunktionen, nach welchen der Entwurf optimiert werden soll, ausgewählt

und der eigentliche Optimierungsprozess gestartet. Nach einem erfolgreichen Optimierungsdurchlauf kann das Ergebnis im Bereich der Anzeige von CAD-Modellen visuell bewertet werden. Ebenfalls gibt es nach erfolgreichem Durchlauf der Optimierung die Möglichkeit das Ergebnis als CAD-Modell sowie als raumbezogene Geodaten zu speichern. Eine detaillierte Umsetzung des erläuterten allgemeinen Ansatzes für die grafische Benutzeroberfläche sowie den einzelnen Bereichen dieser ist in Kapitel 6.4 vorzufinden.

4.6. Steuereinheit der Anwendung

Die interne Komponente *ApplicationController* bildet die Steuereinheit der Applikation *CADtoGIS Schnittstelle*. Hier werden sämtliche Funktionen der Anwendung und die Verbindung zwischen User-Interface, Datenmodell und externen Komponenten realisiert. Dementsprechend umfasst die Komponente unter anderem den Bezug von raumbezogenen Geodaten, die Erzeugung der urbanen Stadt-Objekte sowie CAD-Stadtmodelle und die multidisziplinäre Entwurfsoptimierung der erzeugten Daten. Im Folgenden werden diese Themen ausführlich erläutert und das Zusammenspiel sowie die Kommunikation zwischen den einzelnen Komponenten aufgezeigt.

4.6.1. Zusammenspiel der Komponenten der Anwendung

Die Komponenten der Applikation *CADtoGIS Schnittstelle* aus Abbildung 4.2 markieren keine eindeutigen Ablauffolgen. Aus diesem Grund wird das Konzept in diesem Abschnitt erneut aufgegriffen und mit Hilfe eines Aktivitätsdiagramms (engl. Activity Diagram), das auf der grafischen Modellierungssprache UML basiert, das Zusammenspiel aller Komponenten veranschaulicht. Das Aktivitätsdiagramm ist dabei eine objektorientierte Adaption des Programmablaufplans, welcher häufig auch als Programmflussdiagramm bezeichnet wird.

Hierbei können Aktivitäten (engl. activities) komplexe Abläufe darstellen, die durch unterschiedliche Komponenten der Architektur ausgeführt werden bzw. für die unterschiedliche Komponenten verantwortlich sind. Zur Zuordnung der Aktionen (engl. actions) einer Aktivität zu den verantwortlichen Komponenten können Aktivitätsdiagramme in Partitionen (engl. Activity Partition) unterteilt werden. Da ein entsprechend aufgeteiltes Diagramm wie ein in Schwimmbahnen eingeteiltes Schwimmbecken aussieht, werden die Partitionen auch Schwimmbahnen (engl. swimlanes) genannt.

Das Aktivitätsdiagramm in Abbildung 4.4 zeigt dabei einen allgemeinen Durchlauf der Anwendung ohne sich speziell auf einen Anwendungsfall zu beziehen. Diese „allgemeine“ Aktivität enthält Aktionen, die Eingaben des Benutzers in Ausgaben transformieren und von den einzelnen Anwendungsfällen abgeleitet worden sind. Insofern werden in Abbildung 4.4 nicht alle Anwendungsfälle aufgegriffen sowie die verwendeten Anwendungsfälle nicht vollständig abgebildet. Das Ziel dieses allgemeinen Aktivitätsdiagramms ist die Darstellung des Zusammenspiels bzw. der einzelnen Verantwortungsbereiche der Komponenten im Programmablauf.

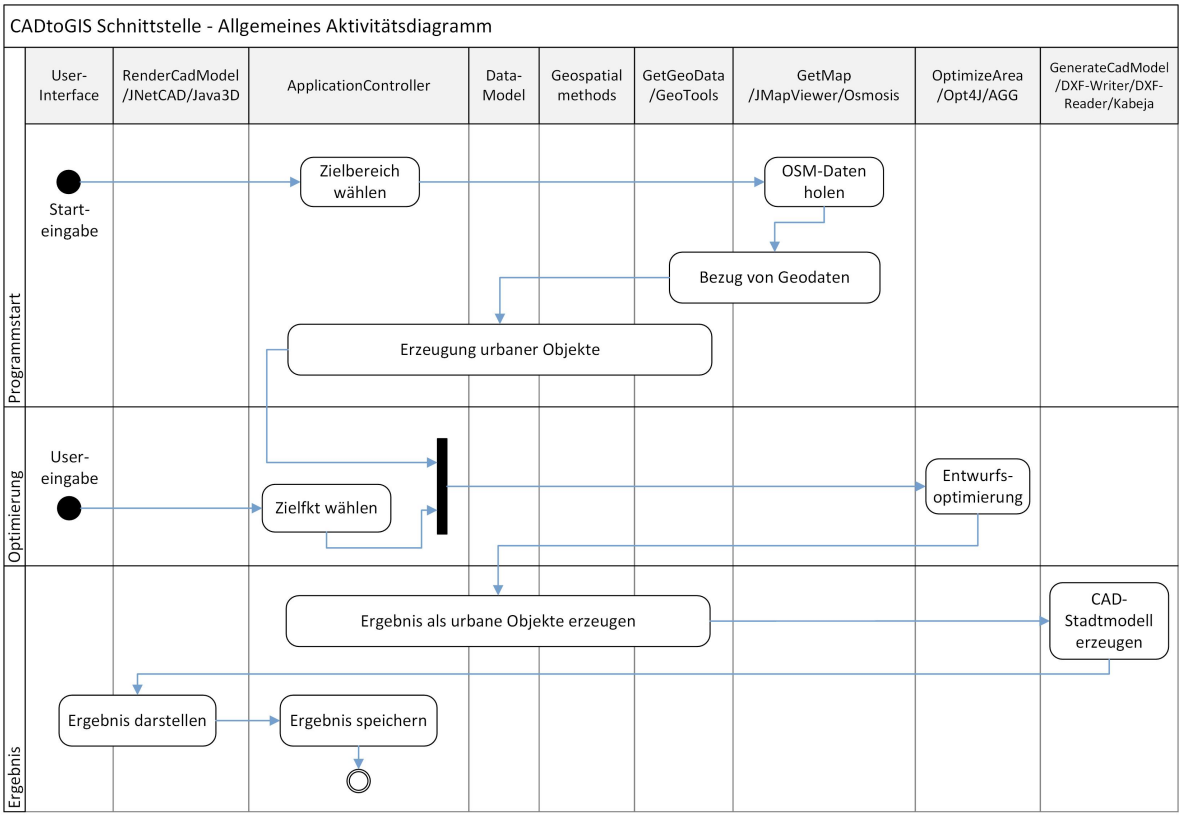

Abbildung 4.4.: Allgemeine Darstellung des Programmablaufs in einem Aktivitätsdiagramm mit den einzelnen Verantwortungsbereichen der Komponenten

Kommunikation zwischen den Komponenten

Die Kommunikation zwischen den einzelnen programminternen und externen Komponenten erfolgt zum einen auf der Grundlage des logischen Datenmodells und der aus diesem entworfenen Stadt-Objekten *UrbanObj*. Das Objekt *UrbanArea*, welches alle erzeugten Stadt-Objekte vereint, ist zentrales Kommunikationsobjekt zwischen den einzelnen Komponenten. Dieses Objekt vereint alle notwendigen Informationen des initialen Stadtbereichs und ist Basis für die multidisziplinäre Entwurfsoptimierung und damit für die Erzeugung neuer Stadtformen. Zum anderen werden für das initiale Einlesen von raumbezogenen Daten und erzeugten CAD-Daten sowie für das finale Speichern der optimierten Ergebnisse Datenaustauschformate, wie zum Beispiel das Esri-Shape- und OSM-Format verwendet. Zusätzlich werden Datenaustauschformate für CAD-Daten, wie das DXF-Format verwendet. Das Thema der verwendeten Datenformate wird im Kapitel 6.1 ausführlich behandelt.

4.6.2. Bezug von Geodaten

Der Bezug von Geodaten in der Applikation *CADtoGIS Schnittstelle* erfolgt über zwei Quellen für raumbezogenen Daten. Zum einen werden Geodaten für die Anzeige und Manipulation der Kartenansicht in der grafischen Benutzeroberfläche sowie zur geografischen Geometriebestimmung und Ermittlung von Metainformationen der einzelnen Stadt-Objekte von OpenStreetMap bezogen. Zum anderen werden AFIS-ALKIS-ATKIS-Geodaten des Landesamtes für Geoinformation und Landentwicklung Baden-Württemberg für den entsprechenden geografischen Zielbereich bezogen. Beide Datenarten werden in Hinsicht auf das logische Datenmodell in der internen Komponente *ApplicationController* im Stadt-Objekt kombiniert und bilden die Informationsgrundlage für die Entwurfsoptimierung. Im Folgenden wird der Bezug der notwendigen Geodaten innerhalb der Komponenten *GetMap* und *GetGeoData* detaillierter dargestellt.

Komponente GetMap

Die interne Komponente *GetMap* dient zur Anzeige und Manipulation einer OpenStreetMap-Kartendarstellung, welche in der grafischen Benutzeroberfläche auf dem Startbildschirm bereitgestellt wird. Mit Hilfe dieser Komponente wählt der Benutzer einen bestimmten Zielbereich mit Optimierungspotenzial aus. Für diesen Bereich werden dann die geografischen Informationen aus lokalen OSM-Dateien geladen und in die Stadt-Objekte eingelesen. Entsprechend können diese Informationen in der grafischen Benutzeroberfläche eingeblendet werden.

Im Detail greift die Komponente *GetMap* auf die externe Komponente *JMapViewer*, welche die OpenStreetMap-API benutzt, zu um ein Kartenobjekt zu erstellen. Dieses Kartenobjekt wird in der Komponente *ApplicationController* verarbeitet und an die Komponente *UserInterface* weitergegeben. Ebenfalls verwendet die Komponente *GetMap* die externe Komponente *Osmosis*, welche die gleichnamige Bibliothek benutzt, um OSM-Dateien zu laden und in Java-Objekte einzulesen. Hierbei wird eine lokal vorhandene OSM-Planet-Datei[12], die den Raum Stuttgart umfasst, verarbeitet. Dabei wird die OSM-Planet-Datei durch Hinzunahme der Grenzen des Zielbereiches verkleinert und eine wesentlich kleinere OSM-Datei lokal geschrieben. Diese OSM-Datei wird darauf eingelesen und anschließend in der Komponente *ApplicationController* in *UrbanObj*-Stadt-Objekte geschrieben und überführt. Hierbei werden die Inhalte der OSM-Datei, welche aus geografischen Knoten (engl. nodes) und Wegen (engl. ways) bestehen (siehe dazu Kapitel 6.1.1), in zwei getrennte Listen geschrieben. Diese Listen mit den raumbezogenen Daten bilden anschließende die Grundlage für die Erzeugung der urbanen Objekte mit der Bezeichnung *UrbanObj*. Eine detaillierte technische Umsetzung der internen Komponente *GetMap* ist in Kapitel 6 zu finden.

[12] siehe `http://www.geofabrik.de`

Komponente GetGeoData

Die interne Komponente *GetGeoData* dient dem Bezug von Geodaten über einen Geoserver oder aus lokalen Geodaten-Dateien. Werden die Geodaten optional von einem Geoserver bezogen, dient diese Komponente der Herstellung einer Verbindung zum Geoserver und dem anschließenden Bezug der Geodaten zu einem bestimmten geografischen Zielbereich. Da es sich bei dieser Arbeit um eine prototypische Entwicklung handelt und das eigentliche Ziel die Schnittstellenentwicklung ist, wurde eine Anbindung an einen Geoserver nicht in Betracht gezogen. Dementsprechend werden die Geodaten aus lokal gespeicherten Geodaten-Dateien bezogen, wobei die Komponente *GetGeoData* dem Einlesen der Geodaten dient.

Im Detail gibt es zwei Möglichkeiten mit dieser Komponente die für den geografischen Zielbereich notwendigen Geodaten zu erhalten. Die erste Möglichkeit ist der Zugriff auf einen Web Feature Service, kurz WFS, eines Geoservers. Dieser WFS greift dabei auf Geodaten, die im einheitlich Datenmodell für alle amtlichen Geodaten Deutschlands, dem AFIS-ALKIS-ATKIS-Datenmodell (AAA-Datenmodell) gespeichert sind, zu. Diese Geodaten werden in einem digitalen Landschaftsmodell, dem DLM, erfasst. Dabei ist das digitale Landschaftsmodell mit dem höchsten Detailgrad das maßstabslose Basis-DLM in dem die Landschaft systematisch nach Objektarten und zugehörigen beschreibenden Informationen strukturiert wird.

Die zweite Möglichkeit ist das direkte Auslesen der Geodaten im AAA-Datenmodell aus lokal gespeicherten Esri[13]-Shape-Dateien, die das Basis-DLM beschreiben. Hierbei gibt es zwei Arten wie die Geodaten gespeichert werden. Einmal als AAA-Daten für das Basis-DLM mit dem Inhalt als Inhaltsebenen und einmal als AAA-Daten für das Basis-DLM mit dem Inhalt als kompakte Beschreibung im NAS-Format. Das NAS-Format basiert dabei auf der „Normbasierten Austauschschnittstelle“. Beide Auslieferungsarten finden in einem Esri-Shape-Format statt. Dabei beinhalten mehrere Zusatzdateien Metainformationen zur Geometrie, welche in der Shape-Datei gespeichert wird. Unter anderem werden in einer Projektions-Datei die Informationen zur verwendeten Projektion gespeichert (siehe dazu das Kapitel 6.2). DBF-Dateien beinhalten die Metadaten der Geometrie in Form von Attributen und SHX-Dateien speichern den Attributindex und bilden dabei die Verknüpfung zwischen Geometrie und Attributen. Eine ausführliche Erklärung des DLM, des AFIS-ALKIS-ATKIS-Datenmodells sowie des Esri-Shape-Formats ist im Kapitel 6.1.2 vorzufinden.

Die interne Komponente *GetGeoData* ließt demnach lokale Geodaten im Esri-Shape-Format mit kompakter Beschreibung im NAS-Format ein und überführt diese in Java-Objekte. Die dadurch erhaltenen Geodaten werden in der internen Komponente *ApplicationController* als Grundlage für die Erzeugung der urbanen Objekte mit der Bezeichnung *UrbanObj* sowie der Entwurfsoptimierung verwendet.

[13] siehe `http://www.esri.com`

4.6.3. Erzeugung urbaner Objekte

Die Erzeugung der urbanen Objekte mit der Bezeichnung *UrbanObj* erfolgt auf Grundlage des logischen Datenmodells in der internen Komponente *ApplicationController* und bildet neben der multidisziplinären Entwurfsoptimierung einen zentralen Teil der Applikation.

Im Detail wird ausgehend von den bezogenen Geodaten der internen Komponenten *GetMap* und *GetGeoData* das zentrale Datenobjekt *UrbanArea* bzw. die enthaltenen Stadt-Objekte *Street* und *Building* in der Komponente *ApplicationController* erzeugt. Hierbei bildet die Dekomposition der vorhandenen Stadtstrukturen die theoretische Grundlage.

Die Erzeugung der urbanen Objekte *Street* und *Building* läuft in drei Schritten ab. Wobei im ersten Schritt die raumbezogenen OpenStreetMap-Daten, welche die Komponente *GetMap* liefert, verwendet werden. Dabei werden die enthaltenen geografischen Punkte, welche im OSM-Format als Knoten bezeichnet werden, den enthaltenen Gebäude- und Straßeninformationen zugeordnet und somit die jeweilige Geometrie auf Basis geografischer Koordinaten in den Objekten *Street* und *Building* erstellt. Zu beachten ist dabei, dass nur Stadt-Objekte aus dem gewählten Zielbereich erstellt und verarbeitet werden. In diesem Schritt werden nicht nur die Geometriedaten der urbanen Objekte sondern auch bereits raumbezogenen Metainformationen aus den OpenStreetMap-Daten in die urbanen Objekte überführt. Unter anderem werden hier bereits Gebäude- und Straßennamen sowie der jeweilige Gebäude- oder Straßentyp festgelegt. Anschließend werden im zweiten Schritt die raumbezogenen AFIS-ALKIS-ATKIS-Daten, welche die Komponente *GetGeoData* liefert, verwendet. Hierbei werden konkret die Nutzflächendaten des amtlichen topographisch-kartographischen Informationssystems, kurz ATKIS, zur Erzeugung weiterer raumbezogener Metainformationen in den urbanen Objekten benutzt. Insofern bieten diese AAA-Daten eine Vielzahl weiterer Daten, die im Umfang dieser Diplomarbeit nicht alle betrachtet und einbezogen werden konnten. Mit Hilfe der Nutzflächendaten, welche im Esri-Shape-Format vorliegen, werden anschließend die Nutzflächentypen ermittelt und allen erzeugten Stadt-Objekten der Nutzflächentyp, in dem sie sich befinden, zugewiesen. Auf Grundlage dieser raumbezogenen Metainformationen in den urbanen Objekten lassen sich im Folgenden Zielfunktionen für die multidisziplinäre Entwurfsoptimierung definieren. Anschließend lassen sich optimierte Stadtstrukturen hinsichtlich dieser Zielfunktionen erzeugen. Im dritten Schritt der Erzeugung der urbanen Objekte wird die Geometrie der Objekte, die bisher durch geografische Koordinaten definiert wird, in eine CAD-konforme Geometrie überführt und mit den geografischen Daten verknüpft. Die Koordinatentransformation bzw. das Mapping der geografischen Geometrie auf CAD-Geometrie erfolgt mit Hilfe der Mercator-Projektion und der internen Komponente *GetGeoData* sowie der externen Komponente *GeoTools*. Eine ausführliche Beschreibung der Umsetzung der vollständigen Erstellung der urbanen Objekte ist in Kapitel 6 vorzufinden.

4.6.4. Erzeugung eines CAD-Stadtmodells

Die Erzeugung eines CAD-Stadtmodells erfolgt in der internen Komponente *GenerateCadModel*. Dabei wird das gängige DXF-Format, welches der kleinste gemeinsame Nenner der meisten CAD-Systeme ist und einen Industriestandard darstellt, verwendet.

Im Detail greift die Komponente *GenerateCadModel* auf die internen Komponenten *DXFWriter* und *DXFReader* zu, um ein CAD-Modell zu schreiben oder zu lesen. Dabei erzeugt die Komponente *DXFWriter* ein neues, lokal gespeichertes, CAD-Modell im DXF-Format. Dieses CAD-Modell wird aus einem *UrbanArea*-Objekt erstellt, indem alle geometrischen Punkte der *Building*-Objekte und *Street*-Objekte ausgelesen und in Polylinien umgewandelt werden. Dabei werden die *Building*-Objekt-Polylinien und *Street*-Objekt-Polylinien in der DXF-Datei in verschiedenen Layern gespeichert. Anschließend wird das erzeugte CAD-Modell in die DXF-Datei gespeichert und kann von einer externen CAD-Anwendung ausgegeben oder intern mit Hilfe der Komponente *RenderCadModel* in der Applikation angezeigt werden.

Die interne Komponente *DXFReader* verwendet die externen Bibliothek Kabeja und dient dem Auslesen von lokal gespeicherte CAD-Modelle im DXF-Format. Nach dem Auslesen eines CAD-Modells wird dieses in *UrbanObj*-Objekte gespeichert. Dabei sind *Building*-Objekte und *Street*-Objekte in der DXF-Datei in verschiedenen Layern enthalten, sodass eine Überführung der gelesenen Polylinien der jeweiligen Layer in die einzelnen Stadt-Objekttypen möglich ist. Ein detaillierte Umsetzung dieser Komponenten wird im Kapitel 6 gezeigt.

4.6.5. Optimierung

Die multidisziplinäre Entwurfsoptimierung wird mit Hilfe der internen Komponente *OptimizeArea*, welche die externen Komponenten *Opt4J* sowie *AGG* verwendet, realisiert. Die externen Komponenten verwenden dabei jeweils die gleichnamigen externen freien Bibliotheken.

Ausgehend vom Lösungsmodell des entworfenen Lösungsansatzes, welches in Abbildung 4.1 illustriert ist, realisiert dabei die Komponente *Opt4J* das Optimierungsframework und die Komponente *AGG* das Optimierungsmodell, welches ein Graphersetzungssystem (engl. *Shape-Rewriting*) darstellt. Das Optimierungsmodell arbeitet hierbei mit einer entworfenen Stadtontologie, welche eine Designsprache für die Erzeugung neuer Stadtformen beinhaltet. Eine ausführliche Erläuterung der theoretischen Konzeption des Optimierungsframeworks ist in Kapitel 5 vorzufinden. Ebenfalls wird die technische Umsetzung des Optimierungsframeworks in Kapitel 6 dargestellt.

4.7. Anwendungsfälle und Verhaltensbeschreibungen

In diesem Unterabschnitt werden die Anwendungsfälle der Applikation *CADtoGIS Schnittstelle* analysiert und erklärt. Das Anwendungsfalldiagramm (engl. use case diagram) aus Abbildung 4.5 stellt alle Nutzungsmöglichkeiten der Applikation dar. Wichtig ist hierbei die logische Abgrenzung zwischen der eigentlichen Applikation *CADtoGIS Schnittstelle* und der externen CAD-Komponente, welche optional zur Darstellung und Manipulation des erzeugten CAD-Modells eingesetzt werden kann. Die Applikations-spezifischen Fälle beschreiben jegliche Funktionalität, die ausschließlich durch die *CADtoGIS Schnittstelle* zur Verfügung gestellt wird. Diese umfassen unter anderem die Auswahl eines geografischen Zielbereichs,

die Auswahl und Anzeige von Geodaten sowie den Start der Entwurfsoptimierung. Die externen Anwendungsfälle hingegen beschreiben jene Aktionen, die Gebrauch von externen CAD-Anwendungen machen.

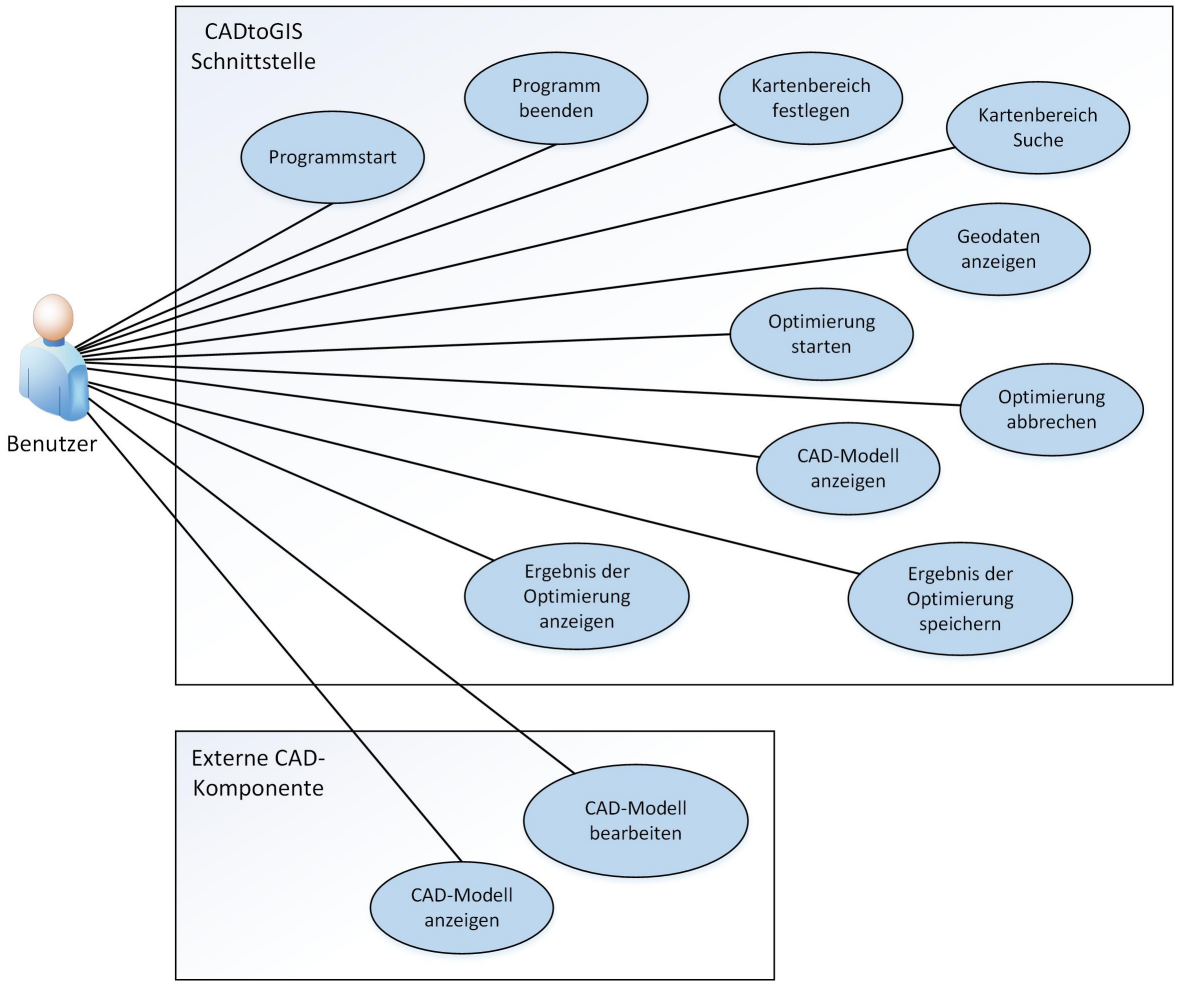

Abbildung 4.5.: Anwendungsfälle der Applikation *CADtoGIS Schnittstelle*

4.7.1. Anwendungsfälle

Die identifizierten Akteure bestehen hier ausschließlich aus dem *Benutzer*. Die Benutzerrolle bezeichnet einen Stadtplaner oder Architekten. Der Benutzer kann alle Funktionen der Applikation verwenden. Außerdem kann er einen bestimmten Stadtbereich auswählen und anschließend für diesen Bereich eine Entwurfsoptimierung anstoßen.

Der Abbildung 4.5 nach bestehen die identifizierten Systeme aus der Applikation *CADtoGIS Schnittstelle*, welche im Folgenden mit „System1“ bezeichnet wird, sowie einer externen CAD-Komponente, die die Bezeichnung „System2“ besitzt.

Anwendungsfall 1: Programmstart		
Ziel:		Applikation startet und zeigt eine Standardansicht der Karte an. Zielbereich ist z. B. der Bereich der Universität Stuttgart.
Akteure:		Benutzer
Beschreibung:		Die Applikation startet und zeigt einen standardmäßig ausgewählten Kartenbereich an. Der Benutzer muss zu diesem Zeitpunkt nichts eingeben.
Ebene:		Benutzersicht
Priorität:		Mittel
Normalablauf		
Vorbedingung:		Applikation wurde gestartet.
1	Benutzer	Benutzer muss keine weiteren Eingaben tätigen.
2	System1	System1 zeigt die Standardkartenansicht an.
Nachbedingung:		Startbildschirm wird angezeigt.

Anwendungsfall 2: Programm beenden		
Ziel:		Applikation wird beendet und alle Fenster der Applikation werden geschlossen.
Akteure:		Benutzer
Beschreibung:		Die Applikation wird nach dem Klicken auf den Button „Beenden“ oder auf den Button „Schließen“ rechts oben im Hauptfenster beendet und das Hauptfenster geschlossen.
Ebene:		Benutzersicht
Priorität:		Mittel
Normalablauf		
Vorbedingung:		Applikation wurde gestartet und kann benutzt werden.
1	Benutzer	Benutzer klickt auf den Button „Beenden“ oder auf den Button „Schließen“ rechts oben im Hauptfenster.
2	System1	System1 wird beendet und das Hauptfenster geschlossen.
Nachbedingung:		Applikation wurde beendet.

Anwendungsfall 3: Kartenbereich festlegen		
Ziel:		Kartenbereich bzw. Zielbereich auf der angezeigten Karte im Hauptfenster auswählen.
Akteure:		Benutzer
Beschreibung:		Der Benutzer wählt einen Zielbereich auf der angezeigten Karte aus, um für diesen Bereich ein Entwurfsoptimierung durchführen zu können.
Ebene:		Benutzersicht
Priorität:		Mittel
Normalablauf		
Vorbedingung:		Applikation wurde gestartet, Kartenansicht geladen und Applikation ist für Eingaben bereit.
1	Benutzer	Benutzer wählt einen bestimmten Bereich in der Kartenansicht durch Setzen von mindestens zwei Punkten auf der Karte durch Klicken auf die Karte.
2	System1	System1 markiert den ausgewählten Bereich in der Kartenansicht farblich und zeigt diesen an.
Nachbedingung:		Hauptfenster zeigt Kartenansicht mit ausgewähltem Zielbereich an.

Anwendungsfall 4: Kartenbereich Suche		
Ziel:		Kartenbereich bzw. Zielbereich auf der angezeigten Karte im Hauptfenster suchen.
Akteure:		Benutzer
Beschreibung:		Der Benutzer sucht einen möglichen Zielbereich auf der angezeigten Karte aus indem er in das Suchfeld im oberen Bereich des Hauptfensters die gewünschten Koordinaten oder die Bezeichnung des Ortes eingibt.
Ebene:		Benutzersicht
Priorität:		Mittel
Normalablauf		
Vorbedingung:		Applikation wurde gestartet, Kartenansicht geladen und Applikation ist für Eingaben bereit.
1	Benutzer	Benutzer gibt in das Suchfeld die gewünschten Koordinaten oder die Bezeichnung des Ortes ein und klickt auf den Button „Suche“ oder bestätigt mit Enter.
2	System1	System1 bewegt den Mittelpunkt der Karte auf den gesuchten Zielbereich.
Nachbedingung:		Hauptfenster zeigt Kartenansicht mit gesuchtem Zielbereich an.

Anwendungsfall 5: Geodaten anzeigen		
Ziel:		Auswahl und Anzeige der Geodaten bzw. Geoinformationen, welche auf der angezeigten Karte im Hauptfenster dargestellt werden sollen.
Akteure:		Benutzer
Beschreibung:		Der Benutzer wählt die gewünschten Geodaten zur Anzeige auf der angezeigten Karte im Hauptfenster aus.
Ebene:		Benutzersicht
Priorität:		Mittel
Normalablauf		
Vorbedingung:		Applikation wurde gestartet, Kartenansicht geladen und Applikation ist für Eingaben bereit.
1	Benutzer	Benutzer klickt auf den Button „Geodaten wählen“.
2	System1	System1 öffnet ein Fenster mit der Auswahl der möglichen Geoinformationen.
3	Benutzer	Benutzer markiert die gewünschten Geoinformationen zur Anzeige und klickt auf den Button „Geodaten anzeigen“.
4	System1	System1 zeigt das Hauptfenster mit den gewählten Geoinformationen in der Kartenansicht an.
Nachbedingung:		Applikation zeigt gewählte Geoinformationen in der Kartenansicht im Hauptfenster an.

Anwendungsfall 6: Optimierung abbrechen		
Ziel:		Abbruch einer gestarteten Entwurfsoptimierung.
Akteure:		Benutzer
Beschreibung:		Der Benutzer bricht eine zuvor gestartete Entwurfsoptimierung eines zuvor gewählten Zielbereichs.
Ebene:		Benutzersicht
Priorität:		Hoch
Normalablauf		
Vorbedingung:		Entwurfsoptimierung eines Zielbereichs wurde gestartet und wird ausgeführt.
1	Benutzer	Benutzer klickt im Informationsfenster des Optimierungsdurchlaufs, das während der Optimierung geöffnet ist, auf den Button „Abbruch“.
2	System1	System1 schließt alle Fenster außer dem Hauptfenster der Applikation und verwirft alle Einstellungen zu der Optimierung.
Nachbedingung:		Entwurfsoptimierung wurde abgebrochen und der Benutzer befindet sich im Hauptfenster.

Anwendungsfall 7: Optimierung starten		
Ziel:		Start der Entwurfsoptimierung des zuvor gewählten Zielbereichs.
Akteure:		Benutzer
Beschreibung:		Der Benutzer startet die Entwurfsoptimierung des zuvor gewählten Zielbereichs.
Ebene:		Benutzersicht
Priorität:		Hoch
Normalablauf		
Vorbedingung:		Applikation wurde gestartet. Zielbereich der Entwurfsoptimierung wurde in der Kartenansicht im Hauptfenster gewählt.
1	Benutzer	Benutzer klickt auf den Button „Optimierung“.
2	System1	System1 öffnet ein Fenster mit der Auswahl der möglichen Zielfunktionen für die Optimierung.
3	Benutzer	Benutzer markiert die gewünschten Zielfunktionen und klickt auf den Button „Optimierung starten“.
4	System1	System1 startet die Optimierung in separatem Fenster, das Informationen zur Optimierung bereitstellt.
5	System1	System1 hat die Optimierung erfolgreich durchgeführt und zeigt Optionen zur Anzeige und Speicherung der Ergebnisse an.
6	Benutzer	Benutzer wählt die Option zur Anzeige der Optimierungsergebnisse.
7	System1	System1 zeigt das Ergebnis als CAD-Modell an.
Nachbedingung:		Entwurfsoptimierung wurde erfolgreich beendet und Applikation zeigt optimierten Zielbereich als CAD-Modell an.

Anwendungsfall 8: Ergebnis der Optimierung speichern		
Ziel:		Speicherung der Ergebnisse der Entwurfsoptimierung.
Akteure:		Benutzer
Beschreibung:		Der Benutzer kann nach erfolgreichem Durchlauf der Entwurfsoptimierung für einen gewählten Zielbereichs die Ergebnisse speichern.
Ebene:		Benutzersicht
Priorität:		Hoch
Normalablauf		
Vorbedingung:		Entwurfsoptimierung für einen gewählten Zielbereich ist erfolgreich durchgelaufen.
1	System1	System1 zeigt ein Fenster mit zwei Optionen an: Ergebnis anzeigen und speichern.
2	Benutzer	Benutzer speichert das Optimierungsergebnis durch Klicken auf den Button „Ergebnis speichern“.
3	System1	System1 zeigt Auswahlfenster für das Zielverzeichnis in das das Ergebnis gespeichert werden soll.
4	Benutzer	Benutzer wählt Zielverzeichnis aus.
5	System1	System1 schließt das Fester und zeigt das Hauptfenster mit dem Ergebnis der Optimierung an.
Nachbedingung:		Applikation zeigt das Hauptfenster mit dem Ergebnis der Entwurfsoptimierung als CAD-Modell an. Gespeichertes Ergebnis liegt im richtigen Datenformat im Zielverzeichnis bereit.

Anwendungsfall 9: Ergebnis der Optimierung anzeigen		
Ziel:		Darstellung der Ergebnisse der Entwurfsoptimierung.
Akteure:		Benutzer
Beschreibung:		Der Benutzer kann nach erfolgreichem Durchlauf der Entwurfsoptimierung für einen gewählten Zielbereichs die Ergebnisse anzeigen lassen.
Ebene:		Benutzersicht
Priorität:		Hoch
Normalablauf		
Vorbedingung:		Entwurfsoptimierung für einen gewählten Zielbereich ist erfolgreich durchgelaufen.
1	System1	System1 zeigt ein Fenster mit zwei Optionen an: Ergebnis anzeigen und speichern.
2	Benutzer	Benutzer lässt das Optimierungsergebnis durch Klicken auf den Button „Ergebnis anzeigen“ darstellen.
3	System1	System1 schließt das Fester und zeigt das Hauptfenster mit dem Ergebnis der Optimierung als CAD-Modell an, ohne die Ergebnisse explizit zu speichern.
Nachbedingung:		Die Applikation zeigt das Hauptfenster mit dem Ergebnis der Entwurfsoptimierung als CAD-Modell an.

Anwendungsfall 10: CAD-Modell anzeigen (intern)		
Ziel:		Darstellung des gewählten Zielbereichs in der Kartenansicht als CAD-Modell.
Akteure:		Benutzer
Beschreibung:		Der Benutzer startet die Erzeugung und Darstellung des gewählten Zielbereichs in der Kartenansicht als CAD-Modell.
Ebene:		Benutzersicht
Priorität:		Mittel
Normalablauf		
Vorbedingung:		Applikation wurde gestartet. Zielbereich der Optimierung wurde in der Kartenansicht im Hauptfenster gewählt.
1	Benutzer	Benutzer klickt auf den Button „CAD-Modell“.
2	System1	System1 erzeugt ein CAD-Modell aus dem gewählten Zielbereich in Abhängigkeit der bezogenen Geodaten.
3	System1	System1 öffnet ein Fenster mit der Darstellung des Zielbereichs als CAD-Modell.
Nachbedingung:		Applikation zeigt das Fenster mit dem erzeugten CAD-Modell an.

Anwendungsfall 11: CAD-Modell anzeigen (extern)		
Ziel:		Darstellung des gewählten Zielbereichs in der Kartenansicht als CAD-Modell in einer externen CAD-Anwendung.
Akteure:		Benutzer
Beschreibung:		Der Benutzer startet die Darstellung des erzeugten CAD-Modells in einer externen CAD-Anwendung.
Ebene:		Benutzersicht
Priorität:		Mittel
Normalablauf		
Vorbedingung:		Applikation wurde gestartet. Zielbereich der Optimierung wurde in der Kartenansicht im Hauptfenster gewählt.
1	Benutzer	Benutzer klickt auf den Button „CAD-Modell“.
2	System1	System1 erzeugt ein CAD-Modell aus dem gewählten Zielbereich in Abhängigkeit der bezogenen Geodaten.
3	System1	System1 öffnet ein Fenster mit der Darstellung des Zielbereichs als CAD-Modell.
4	Benutzer	Benutzer klickt auf den Button „CAD-Modell speichern“.
5	System1	System1 zeigt Auswahlfenster für das Zielverzeichnis in das das Ergebnis gespeichert werden soll.
6	Benutzer	Benutzer wählt Zielverzeichnis aus und speichert das erzeugte CAD-Modell im DXF-Format.
7	Benutzer	Benutzer öffnet System2, das einer externen CAD-Anwendung entspricht, und lädt das gespeicherte CAD-Modell im DXF-Format.
8	System2	System2 öffnet ein Fenster mit der Darstellung des CAD-Modells.
Nachbedingung:		Externes CAD-System zeigt das Fenster mit dem erzeugten CAD-Modell an.

Anwendungsfall 12: CAD-Modell bearbeiten		
Ziel:		Bearbeitung des erzeugten und dargestellten CAD-Modells in einer externen CAD-Anwendung.
Akteure:		Benutzer
Beschreibung:		Der Benutzer kann das erzeugte und dargestellte CAD-Modell in einer externen CAD-Anwendung bearbeiten.
Ebene:		Benutzersicht
Priorität:		Mittel
Normalablauf		
Vorbedingung:		CAD-Modell wurde in der Applikation erzeugt und in einer externen CAD-Anwendung geöffnet und dargestellt.
1	Benutzer	Benutzer bearbeitet in externer CAD-Anwendung das CAD-Modell.
2	System2	System2 erzeugt umgehend ein neues CAD-Modell und stellt dies dar.
3	Benutzer	Benutzer speichert das bearbeitete CAD-Modell als DXF-Format im externen CAD-System.
4	Benutzer	Benutzer importiert das bearbeitete CAD-Modell mit Applikation bzw. System1.
5	System1	System1 zeigt das neue CAD-Modell im Fenster „CAD-Modell“ an.
Nachbedingung:		Applikation zeigt das Fenster mit dem bearbeitetem CAD-Modell an.

5. Optimierungsframework

Dieses Kapitel beschreibt die konzeptuelle Realisierung des Optimierungsframeworks, das auf der Grundlage des entworfenen Lösungsansatzes aus Kapitel 4 basiert. Zunächst wird in ausführlicher Weise auf die theoretische Konzeption des Optimierungsframeworks eingegangen und anschließend der Aufbau sowie die Arbeitsweise des darin enthaltenen Optimierungsmodells beschrieben. Unter anderem werden dabei die verwendete Designsprache für die Erzeugung neuer Stadtformen sowie das verwendete Optimierungsmodell detailliert diskutiert. Die technische Umsetzung des Optimierungsframeworks wird in der internen Applikationskomponente *OptimizeArea* mit Hilfe der Java-Klassenbibliotheken *Opt4J* und *AGG* realisiert und genauer in Kapitel 6 beschrieben. Um den Lesefluss nicht zu stören und das Verständnis dieses Kapitels zu fördern, werden wichtige Illustrationen, die in vorherigen Abschnitten bereits eingeführt und besprochen worden sind, hier nochmals eingefügt.

5.1. Konzeption des Optimierungsframeworks

Wie bereits erwähnt, basiert die Realisierung des Optimierungsframeworks auf der Grundlage des entworfenen Lösungsansatzes aus Kapitel 4. Hierbei zeigt die Abbildung 4.1 die Einordnung des Optimierungsframeworks in das entworfene Lösungsmodell, wobei der dargestellte mittlere Teilbereich mit der Bezeichnung „Optimierer“ das Optimierungsframework bildet. Der eigentliche Optimierungsprozess basiert auf der Arbeit von [Gey09] und läuft auf erzeugten urbanen Daten, die im Kapitel 4.4 eingeführt worden sind und die Bezeichnung *UrbanObj* haben, ab.

Das Optimierungsframework der Applikation *CADtoGIS Schnittstelle* ist in Abbildung 5.1 als Ausschnitt des Lösungsmodells dargestellt. Hierbei erhält der „Optimierer“ zunächst als Dateninput initiale CAD-Modelldaten verknüpft mit raumbezogenen Metainformationen in zuvor erzeugten Stadt-Objekten (siehe Punkt 1 in Abbildung 5.1). Anschließend wird das komplette CAD-Modell anhand einer zuvor definierten Menge von Zielfunktionen nach bestimmten Aspekten bewertet (Punkt 2 in Abbildung 5.1). Entspricht diese Bewertung nicht den geforderten Kriterien wird das komplette Stadt-Objekt dem enthaltenen Optimierungsmodell, welches mit Hilfe eines Graphersetzungssystems (engl. graph rewriting) realisiert wird, übergeben.

In Punkt 3 der Abbildung 5.1 wird mit Hilfe des Optimierungsmodells anhand des übergebenen Stadt-Objekts und der darin enthaltenen Eintrittspunkte (engl. entry points) in den urbanen Zielbereich ein neues Stadt-Objekt erzeugt. Diese Erzeugung des Stadt-Objekts erfolgt durch die Anwendung eines Erzeugungsgraphen für Stadtformen. Der Erzeugungsgraph

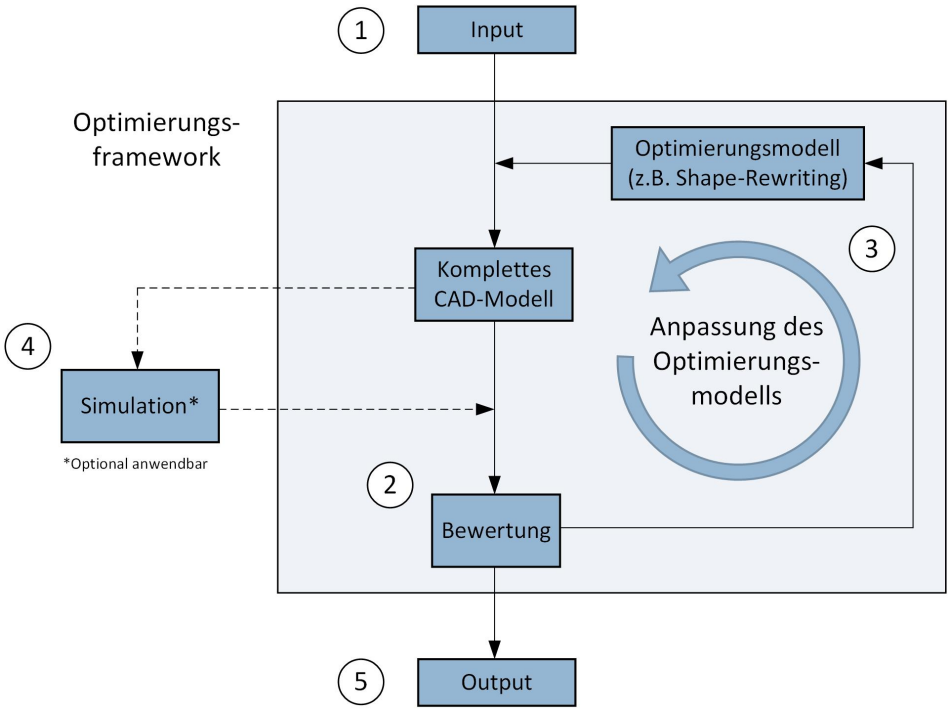

Abbildung 5.1.: Das Optimierungsframework mit enthaltenem Optimierungsprozess für die multikriterielle Optimierung. Dabei stellt der blaue Pfeil den numerischen Optimierungsprozess dar.

wird dabei durch eine Designsprache und der darin enthaltene Formgrammatik definiert. Nach einer anfänglichen Erzeugung eines neuen Stadt-Objekts wird dieses erneut im Hinblick auf die definierten Zielfunktionen bewertet und bei Nichterfüllung der geforderten Kriterien ein erneuter Durchlauf des Optimierungsprozesses gestartet. Anschließend wird nun das Optimierungsmodell bezüglich des verwendeten Erzeugungsgraphen angepasst, um erneut eine neue Variante eines Stadt-Objekts erzeugen zu können. Die Anpassung des Optimierungsmodells erfolgt dabei durch den Umbau des zuvor erstellten Erzeugungsgraphen mit Hilfe eines Optimierungsalgorithmus. Dieser kann zum Beispiel ein evolutionärer Optimierungsalgorithmus, wie ihn die Java-Klassenbibliothek *Opt4J* verwendet, sein. Die Anpassung des Optimierungsmodells wird hier in Beziehung zur Erstellung neuer Stadtformen als *Shape-Rewriting* bezeichnet.

Der Optimierungsprozess wird nun solange durchlaufen bis ein Stadt-Objekt bzw. CAD-Stadtmodell gefunden wird, das die definierten Zielfunktionen und damit die geforderten Kriterien erfüllt. Dieses gefundene Stadt-Objekt bildet zugleich die Output-Daten des Optimierungsframeworks, was bei Punkt 5 in Abbildung 5.1 zu sehen ist. Weiterhin illustriert die Abbildung 5.1 in Punkt 3 diesen Optimierungsprozess als zyklischen blauen Pfeil. Der beschriebene Optimierungsprozess stellt dabei eine abgeänderte Variante des Optimierungsprozesses in der Arbeit [Gey09] dar (siehe Abbildung 3.1).

Weiterhin können optional in die Bewertung des erzeugten Stadt-Objekts die Ergebnisse der Simulation dieses Objekts einfließen (siehe Punkt 4 in Abbildung 5.1). Aufgrund der Tatsache, dass kein passendes Simulationsframework zur freien Verfügung stand, wurde die Anbindung dieses zunächst zurückgestellt. In den nächsten Abschnitten dieses Kapitels ist im Folgenden eine detaillierte Erläuterung des Optimierungsmodells sowie der verwendeten Designsprache zur Erzeugung neuer Stadtformen vorzufinden.

5.2. Designsprache und Formgrammatiken

Für die prozedurale Erstellung jeglicher 2D- wie auch 3D-Formen bzw. Geometrien ist eine Designsprache mit enthaltenen Designmustern, Formgrammatiken und Erzeugungsregeln Voraussetzung. Dieser Abschnitt beschreibt die theoretische Realisierung des Aufbaus einer Designsprache zur prozeduralen Erzeugung neuer Strukturen im Bereich der Städteplanung. Dabei bildet die hier entworfene Designsprache, welche einer bestimmten Stadtontologie zugrunde liegt, die Ausführungsbasis für das entworfene Optimierungsmodell *Shape-Rewriting*, welches im nächsten Abschnitt genauer beschrieben wird. Im Folgenden werden zunächst einige Bezüge zu bereits existierenden Ansätzen für den Entwurf einer Designsprache im Kontext der Stadtplanung angebracht, um damit die Erstellung einer Designsprache zu verdeutlichen. Anschließend wird auf den vereinfachten Entwurf der verwendeten Designsprache näher eingegangen.

5.2.1. Erstellung einer Designsprache

Für die Konzeption einer vereinfachten Designsprache wurden einige bereits existierende Lösungsansätze aus der Literatur als Grundlage verwendet. Dabei stellen die Arbeiten [BDS08] und [DRS07] die interessantesten Informationen im Hinblick auf die Verwendung in einer einfachen Designsprache bereit.

Die Arbeit [BDS08] beschäftigt sich dabei mit der Erstellung eines vollständig generativen Modells für den städtebaulichen Entwurf. Dabei soll dieses generative Modell die Brücke zwischen Geoinformationssystemen und Formgrammatiken schlagen. Wie in Abbildung 5.2 dargestellt, wird hierbei ein Entwurfsprozess, welcher auf einer Stadtontologie aufbaut, definiert. Dieser Entwurfsprozess wird dabei von einem Designwerkzeug verwendet und erhält GIS-Metadaten als initiale Dateneingabe. Anschließend wird eine Designsprache mit enthaltenem Erzeugungsmodell, Designmustern, Formgrammatiken sowie Erzeugungsregeln für die Beschreibung eines urbanen Gebiets erstellt. Aufbauend auf dieser Designsprache werden anschließend neue Stadtformen, wie zum Beispiel Straßen, Gebäude, Gebäudeblöcke und weitere beschrieben und erzeugt. Die Abbildung 5.3 zeigt dabei die Definition von zwei unterschiedlichen Varianten eines Straßenobjekts, welche von der definierten Designsprache abgeleitet worden sind. Hierbei gibt die Nummerierung die einzelnen urbanen Objekte an, welche mit Hilfe der Designsprache und damit dem generativen Modell zu einer neuen Stadtstruktur zusammengesetzt werden können.

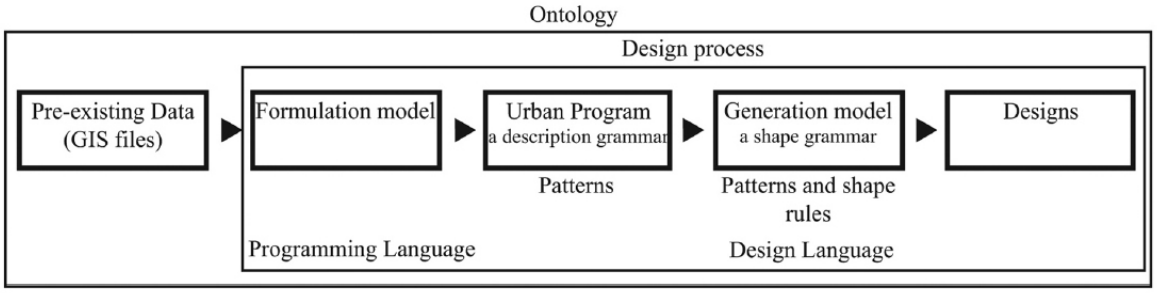

Abbildung 5.2.: Zusammenhang von Ontologie, Designsprache, Entwurfsmuster, Formgrammatiken und Erzeugungsregeln (vgl. [BDS08])

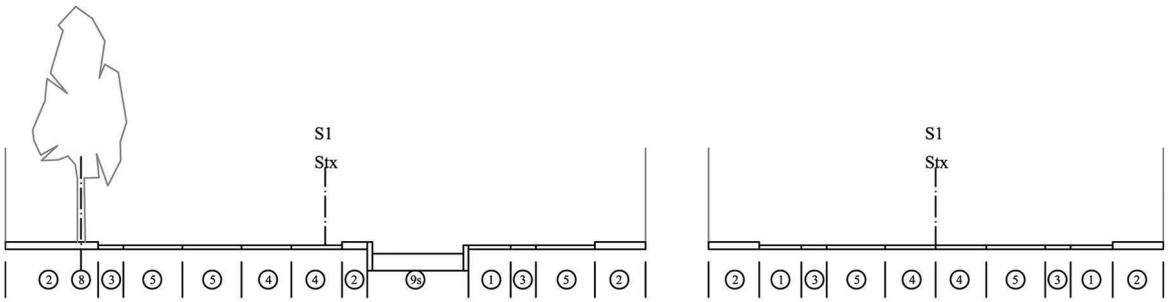

Abbildung 5.3.: Basierend auf der definierten Stadtontologie und Designsprache werden zwei verschiedenen Varianten des Straßenobjekts erstellt (vgl. [BDS08])

Die Arbeit [DRS07] beschäftigt sich mit der Entwicklung einer Designsprache zur Beschreibung eines konkreten Problemfalls einer nordafrikanischen Altstadt mit der Bezeichnung „Medina von Marrakesch“. Dabei wird das vorliegende urbane Teilgebiet der Stadt Marrakesch zunächst mit Hilfe einer Dekomposition in einzelne urbane Grundstrukturen zerlegt und anschließend daraus eine Designsprache mit Designmustern, Formgrammatiken sowie Erzeugungsregeln erstellt. In Abbildung 5.4 ist die prozedurale Erzeugung des urbanen Teilgebiets anhand erstellter Erzeugungsregeln, welche aus der Designsprache abgeleitet worden sind, zu sehen. Hierbei wird eine „bottom-up“ Dekomposition durchgeführt, bei der ausgehend von bestimmten Eintrittspunkten in den urbanen Zielbereich neue Stadtstrukturen erzeugt werden.

Weiterhin werden zur prozeduralen Erzeugung von Stadtstrukturen sogenannte Lindenmayer-Systeme, kurz L-Systeme, verwendet. L-Systeme gehören zur Kategorie der Ersetzungssysteme und bilden im Allgemeinen in der Natur wachsende Formen bzw. Geometrien mit Hilfe von Produktionsregeln ab. Weiterführende Informationen zu L-Systemen sind in einschlägiger Literatur zu finden (vgl. [MP06]).

Abschließend kann festgehalten werden, dass in der bisherigen Forschung die Erstellung von Designsprachen im Kontext der Stadtplanung bereits in größerem Umfang stattfindet.

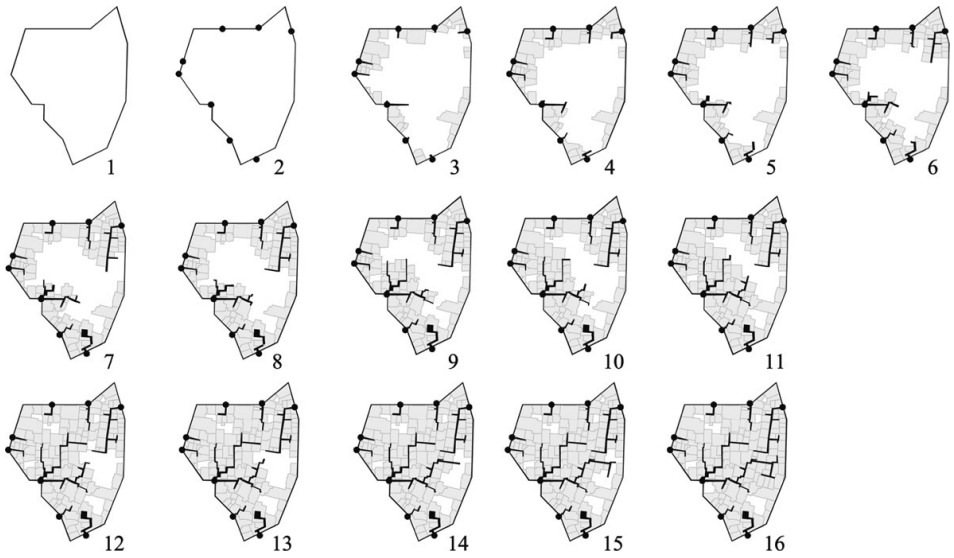

Abbildung 5.4.: Basierend auf der definierten Stadtontologie und Designsprache wird ein Teilgebiet der „Medina von Marrakesch“ prozedural erzeugt (vgl. [DRS07])

Diese Tatsache wird aufgegriffen und anschließend wird versucht mit Hilfe der vorgestellten Arbeiten eine vereinfachte Variante einer Designsprache für einen universellen Einsatz in der Stadtplanung zu entwickeln.

5.2.2. Beschreibung der entworfenen Designsprache

Die in diesem Abschnitt entworfenen vereinfachte Designsprache baut auf dem logischen Datenmodell des vorherigen Kapitels auf (siehe Kapitel 4.4). Dabei stellt das logische Datenmodell die verwendete Stadtontologie, die die eigentliche Designsprache definiert, dar. Abbildung 5.5 zeigt die rudimentären Erzeugungsregeln für neue Stadtstrukturen. Diese werden im Folgenden in der Tabelle 5.1 beschrieben. Dabei ist anzumerken, dass diese Erzeugungsregeln eine Basis für zukünftige Erweiterungen der Designsprache bilden.

Erzeugungsregel	Beschreibung
`01. Setze Straße gerade aus fort`	Diese Regel setzt an einem bereits existierenden Straßenpunkt an und erweitert die Straße in gerader Richtung um eine Längeneinheit (siehe Punkt 1 in Abbildung 5.5).
`02. Setze Straße nach links fort`	Diese Regel setzt an einem bereits existierenden Straßenpunkt an und erweitert die Straße in Linksrichtung um eine Längeneinheit (siehe Punkt 2 in Abbildung 5.5).

`03. Setze Straße nach rechts fort`	Diese Regel setzt an einem bereits existierenden Straßenpunkt an und erweitert die Straße in Rechtsrichtung um eine Längeneinheit (siehe Punkt 3 in Abbildung 5.5).
`04. Verbinde zwei Straßenendpunkte`	Diese Regel setzt an zwei in einem bestimmten Radius befindlichen bereits existierenden Straßenpunkten an und verbindet diese zu einer durchgehenden Straße (siehe Punkt 4 in Abbildung 5.5).
`05. Erzeuge ein Gebäude an Straßenpunkt`	Diese Regel setzt an einem bereits existierenden Straßenpunkt an und platziert parallel zur Straße ein Gebäude mit einer Längeneinheit Breite (siehe Punkt 5 in Abbildung 5.5).
`06. Verbinde alle Eintrittspunkte`	Diese Regel verbindet alle Eintrittspunkte zum urbanen Zielbereich miteinander (siehe Punkt 6 in Abbildung 5.5).

Tabelle 5.1.: Rudimentäre Erzeugungsregeln der einfachen Designsprache

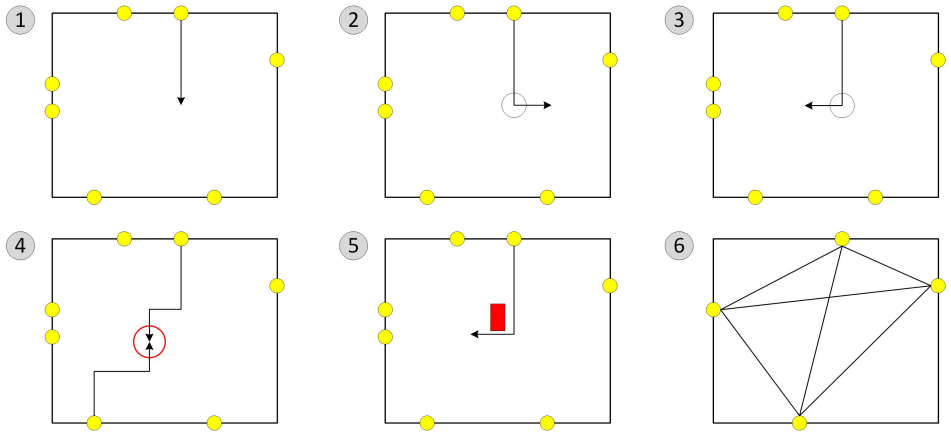

Abbildung 5.5.: Rudimentäre Erzeugungsregeln der vereinfachten Designsprache. Gelb markierte Punkte stellen Eintrittspunkte in den urbanen Zielbereich dar.

5.3. Optimierungsmodelle

Dieser Abschnitt beschreibt im Folgenden die theoretische Konzeption des verwendeten Optimierungsmodells. Dabei wird detailliert auf Graphersetzungssysteme eingegangen und mögliche Alternativen betrachtet. Abschließend folgt die konkrete Beschreibung des verwendeten Optimierungsmodells mit der Bezeichnung *Shape-Rewriting*.

5.3.1. Verwendetes Optimierungsmodell

Um das verwendete Optimierungsmodell nochmals in den groben Zusammenhang der Architektur der Applikation *CADtoGIS Schnittstelle* einordnen zu können, wird die Abbildung 4.1 und anschließend die Abbildung 5.1 betrachtet. Wie in dem ersten Abschnitt dieses Kapitels beschrieben, erhält das verwendete Optimierungsmodell eine initiale Dateneingabe, auf welcher dann mit Hilfe eines Erzeugungsgraphen neue Daten erzeugt werden. Die für das Optimierungsmodell verwendete theoretische Grundlage bilden dabei sogenannte Graphersetzungssysteme.

Graphersetzungssysteme (engl. graph rewriting) dienen der formalen Beschreibung der Veränderung von Graphen und haben eine Vielzahl an Anwendungsmöglichkeiten. In Fall einer Entwurfsoptimierung im Bereich der Stadtplanung, können Graphen dazu verwendet werden neue Stadtstrukturen zu Erzeugen. Dabei werden mit Hilfe von Produktions- bzw. Erzeugungsregeln, welche einzelne Knoten des Graphen darstellen, ganze Erzeugungsgraphen aufgebaut. Diese Graphen beschreiben dann die Erzeugung einer bestimmten Stadtstruktur, wie zum Beispiel den konkreten Verlauf der Straße. Die Graphersetzung bildet dabei den variablen Teil des Systems, welcher bei der eigentlichen Optimierung Verwendung findet.

Die Graphersetzung definiert eine Menge M von Graphersetzungsregeln $p : \mathfrak{L} \rightarrow \mathfrak{R}$, wobei eine Graphersetzungsregel p aus dem Ausgangsgraphen $\mathfrak{L}$ und dem Ersetzungsgraphen $\mathfrak{R}$ besteht. Mit Hilfe der direkten Anwendung der Regel p auf einen Arbeitsgraphen entsteht durch Teilersetzung ein modifizierter Arbeitsgraph. Eine ausführliche Beschreibung des theoretischen Unterbaus aus dem Bereich der Graphentheorie ist in einschlägiger Literatur zu finden.

Das Optimierungsmodell arbeitet damit mit Erzeugungsregeln, die einen Erzeugungsgraphen innerhalb eines Graphersetzungssystems definieren. In Abhängigkeit der im Optimierungsframework durchgeführten Bewertung eines Ergebnisses bei jedem Durchlauf des Optimierungsprozesses wird der zuvor definierte Erzeugungsgraph abgeändert und so eine neue Stadtstruktur erstellt. Dieser ganze Vorgang wird im weiteren Verlauf der Diplomarbeit als *Shape-Rewriting* bezeichnet und im folgenden Abschnitt detaillierter betrachtet.

5.3.2. Beschreibung des Optimierungsmodells

Die Idee des verwendeten *Shape-Rewritings* ist die Erzeugung neuer Stadtformen bzw. Stadtstrukturen durch Verwendung eines Graphersetzungssystems. Dabei sollen mit Hilfe von bestimmten Graphersetzungsregeln neue Erzeugungsgraphen erstellt werden. Diese Erzeugungsgraphen generieren jeweils eine neue Geometrie im urbanen Zielbereich anhand der im Erzeugungsgraphen verwendeten Erzeugungsregeln der zugrunde liegenden Designsprache. Die Abbildung 5.6 stellt hierbei die Ersetzung eines Teilgraphen des Ausgangs-Erzeugungsgraphen links durch einen anderen Teilgraphen im modifizierten Ausgangs-Erzeugungsgraphen rechts dar.

Wie in dem ersten Abschnitt dieses Kapitels beschrieben, erhält das verwendete Optimierungsmodell *Shape-Rewriting* eine initiale Dateneingabe in Form eines Stadt-Objekts mit der

Bezeichnung *UrbanArea*. Weiterhin werden neue Erzeugungsgraphen für die Menge der im Stadt-Objekt vorhandenen Eintrittspunkte in den urbanen Zielbereich erstellt und die neu erzeugte Ergebnisstruktur an das Optimierungsframework weitergegeben. Nach anschließender Bewertung werden die zuvor erstellten Erzeugungsgraphen gegebenenfalls modifiziert. Dieser Ablauf wird im Optimierungsprozess so lange ausgeführt bis ein zufriedenstellendes Ergebnis hinsichtlich der definierten Zielfunktionen erhalten wird.

Die grobe Beschreibung der technischen Umsetzung des verwendeten Optimierungsmodells *Shape-Rewriting* ist im Kapitel 6.6 vorzufinden.

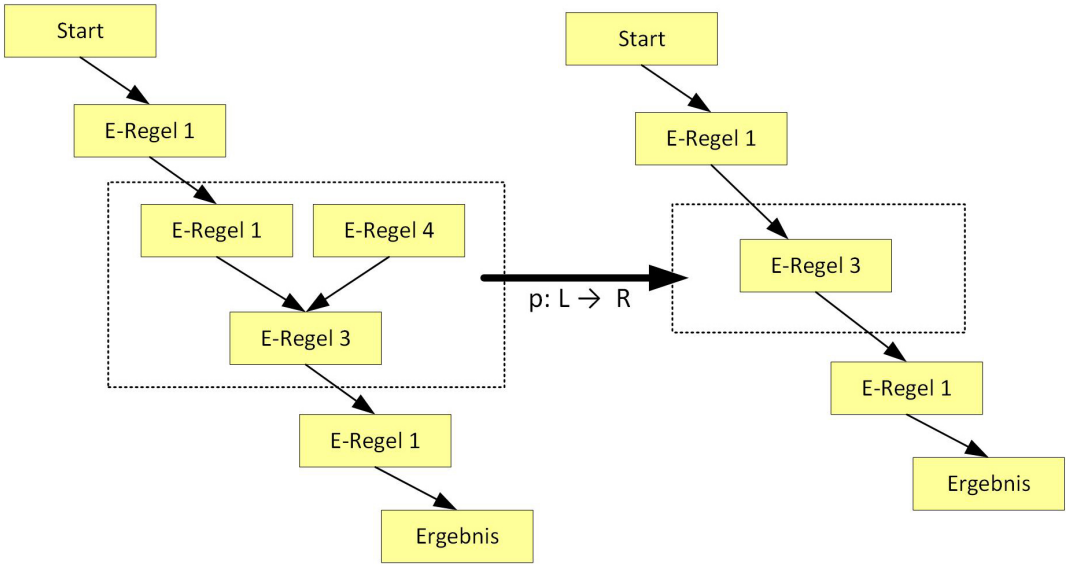

Abbildung 5.6.: Beim *Shape-Rewriting* entsteht durch Graphersetzung ein neuer Erzeugungsgraph (rechter Graph) und damit eine neue Stadtstruktur.

6. Implementierung

Das Kapitel der Implementierung beschreibt die konkrete Umsetzung der Applikation *CADtoGIS Schnittstelle*. Dazu müssen die theoretischen Erkenntnisse der letzten Kapitel erneut betrachtet und hinsichtlich ihrer Umsetzung in der Applikation von ihrer praktischen Seite durchleuchtet werden. Zuallererst soll ein allgemeiner Überblick über die verwendeten Datenformate gegeben werden. Anschließend werden impliziert durch die Verwendung verschiedener Datenformate zur Darstellung von geografischen Informationen die notwendigen Projektionen dieser Daten betrachtet. Weiterhin wird ein Überblick aller verwendeten Technologien aufgelistet, bevor auf die Umsetzung der grafischen Benutzeroberfläche sowie der zentralen Komponente *ApplicationController* eingegangen wird. Zum Ende des Kapitels wird schließlich die Umsetzung der Entwurfsoptimierung in der Komponente *OptimizeArea* durch den Einsatz eines Optimierungsframeworks aufgezeigt.

6.1. Verwendete Datenformate

Da die zu implementierende Applikation *CADtoGIS Schnittstelle* die Umsetzung eines kombinierten Einsatzes verschiedener Technologien und Softwarewerkzeuge darstellt, hat dies die Verwendung verschiedener Datenformate zur Folge. Unter anderem werden diese Datenformate zur Verknüpfung von CAD-Geometrie mit Geoinformationen eines bestimmten geografischen Zielbereiches verwendet. Im Folgenden werden ein Überblick aller Datenformate sowie ihre Verwendung in der Applikation gegeben.

6.1.1. OSM-Format

Das OpenStreetMap[1]-Datenformat, kurz OSM-Format[2], enthält OpenStreetMap-Daten und basiert auf dem XML-Format. Es dient dem Austausch von raumbezogenen Geometrie- und Metadaten basierend auf OpenStreetMap. Die Syntax des OSM-Formats entspricht der Ausgabe einer OpenStreetMap-API[3]-Anfrage und beinhaltetet in Grundelemente eingeteilte raumbezogene Daten. Diese Grundelemente bestehen aus sogenannten Knoten (engl. nodes), Wegen (engl. ways) und Beziehungen (engl. relations) sowie deren Metainformationen, welche in „tags“ gespeichert werden. Ebenfalls können sogenannte Änderungssätze enthalten

[1] siehe `http://www.openstreetmap.de`
[2] siehe `http://wiki.openstreetmap.org/wiki/DE:OSM_XML`
[3] siehe `http://wiki.openstreetmap.org/wiki/API`

sein. Diese beinhalten Änderungen, die von OpenStreetMap-Nutzern zu einer bestimmten Zeit vorgenommen wurden und sollen historische Entwicklungen der geografischen Daten ermöglichen.

Die raumbezogenen Daten von OpenStreetMap werden in einer zentralen Datenbank erfasst und im Normalfall über die OpenStreetMap-API bezogen. Eine alternative Bezugsmöglichkeit dieser Daten besteht in der Verwendung von lokal gespeicherten OpenStreetMap-Daten im OSM-Format. Dafür werden Kopien der gesamten OpenStreetMap-Datenbank wöchentlich in einer sogenannten Planet-Datei[4] [5] zur Verfügung gestellt. Ebenso werden tägliche und stündliche Updates sowie Ausschnitte für einzelne Länder, Regionen und Städte bereitgestellt. Diese können unter anderem auf der Internetseite `http://www.geofabrik.de` bezogen werden.

Die Verwendung von raumbezogenen Daten im OSM-Format in der Applikation *CADtoGIS Schnittstelle* erfolgt in der internen Komponente *GetMap*. Hierbei werden benötigte Geometrien, welche über geografische Koordinaten beschrieben sind, sowie zugehörige Metainformationen aus den OSM-Daten extrahiert und in der zentralen Komponente *ApplicationController* weiterverarbeitet.

6.1.2. Shape-Format

Das Shape-Format der US-amerikanischen Firma Esri[6] dient dem Austausch von Geodaten bzw. Kartendaten und wird als Quais-Standard im GIS-Umfeld angesehen. Das Esri-Shape-Format ist relativ einfach aufgebaut und bietet unter anderem eine hohe Anzahl an freier Software die das Format unterstützen. Ebenfalls wird dieses Format von der Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland zur Speicherung und Darstellung von Geobasisdaten im sogenannten AAA-Modell verwendet. Dieses Thema sowie eine detailliertere Beschreibung aller zum Esri-Shape-Format gehörenden Dateien wird im Folgenden aufgezeigt.

Die Verwendung von raumbezogenen Daten im Shape-Format in der Applikation *CADtoGIS Schnittstelle* erfolgt in der internen Komponente *GetGeoData*. Hierbei werden die raumbezogenen Shape-Daten, die aus den AAA-Geobasisdaten mit Hilfe der normbasierte Austauschschnittstelle (NAS) bezogen worden sind und lokal vorliegen, verarbeitet. Anschließend werden benötigte Geometrien und Metainformationen in einer geografischen Beschreibung aus den Shape-Daten extrahiert und in der zentralen Komponente *ApplicationController* weiterverarbeitet.

[4] siehe `http://planet.openstreetmap.org`
[5] siehe `http://www.geofabrik.de`
[6] siehe `http://www.esri.com`

Shape-Format: SHP-Datei

Das Esri-Shape-Format besteht nicht nur aus einer Datei, die die raumbezogenen Daten speichert, sondern vielmehr aus mindestens drei Dateien. Die wichtigste dieser Dateien ist die SHP-Hauptdatei, welche zur Speicherung der eigentlichen Geometriedaten dient. Die Geometriedaten bzw. die einzelnen Elemente können hierbei zum Beispiel vom Typ Punkt, Linie oder Polygon sein und als 2D- oder 3D-Geometrie vorliegen. Zusätzlich kann innerhalb einer externen Legendendatei das Design dieser Geometrie, wie zum Beispiel die Linienstärke und -farbe, definiert werden.

Shape-Format: DBF-Datei

Die Sachdatendatei im DBF-Format bildet nach der SHP-Datei die zweitwichtigste Datei und beschreibt die Attributdaten der Geometrieelemente. Dabei liegen die Attributdaten im sogenannten dBASE-Format, welches Datenbanktabellen in speziell strukturierten Dateien speichert, vor. Für jedes Geometrieelement der SHP-Hauptdatei gibt es einen Eintrag in der DBF-Datei, was einer eins zu eins Relation entspricht. Außerdem müssen die Attributdaten in der gleichen Reihenfolge gespeichert sein wie die Geometrieelement der SHP-Hauptdatei.

Shape-Format: SHX-Datei

Die Indexdatei im SHX-Format definiert die Verknüpfung zwischen den Sachdaten der DBF-Datei und den Geometrieelementen der SHP-Hauptdatei in Form einer eins zu eins Beziehung. Die SHX-Datei ist demnach eine Link-Datei und dient dem schnellen Auffinden bestimmter Attribute für bestimmte Geometrieelemente.

Shape-Format: PRJ-Datei

Die Projektionsdatei im PRJ-Format beschreibt, wie der Name bereits sagt, die Projektion der Geometriedaten der SHP-Datei. Dabei wird detailliert das Projektionsformat, welches das verwendete Koordinatensystem sowie genaue Informationen zur Projektion umfasst, definiert. Die Beschreibung findet dabei im sogenannten Well-known Text-Format (WKT-Format) statt. Diese Repräsentation stellt eine Auszeichnungssprache dar und dient unter anderem zur Darstellung von Transformationen zwischen einzelnen geografischen Bezugssystemen.

Digitales Landschaftsmodell

Das digitale Landschaftsmodell, kurz DLM[7], ist ein numerisches Modell der Landschaft, das alle darin enthaltenen topographischen Objekte und das Relief der Erdoberfläche im Vek-

[7] siehe `http://www.adv-online.de/AdV-Produkte/Landschafts-und-Gelaendemodelle`

torformat beschreibt. Zur einheitlichen topographischen Beschreibung des ganzen Gebietes der Bundesrepublik Deutschland durch digitale Landschaftsmodelle, werden mehrere Modelle mit unterschiedlicher Informationsdichte bzw. verschiedenen Maßstäben bereitgestellt. Dabei bildet das sogenannte Basis-DLM das maßstabslose digitale Basislandschaftsmodell mit dem höchsten Detailgrad. In diesem Modell wird die Landschaft systematisch nach Objektarten und zugehörigen Metainformationen strukturiert. Demnach werden die Objekte einer bestimmten Objektart zugeordnet und durch ihre räumliche Lage, ihren geometrischen Typ, beschreibende Attribute und Relationen zu anderen Objekten definiert. Die einzelnen Objektarten des Basis-DLM sowie die Bildungsgesetze der Objekte sind im sogenannten ATKIS-Objektartenkatalog, kurz ATKIS-OK, festgelegt. Weiterführende Informationen zum ATKIS-OK sind im nächsten Abschnitt vorzufinden. Detailliertere Informationen zum Thema DLM sowie die einzelnen Objektartenkataloge können unter folgender Internetseite `http://www.adv-online.de` bezogen werden.

AAA-Modell

Hintergrund des AAA-Modells[8] ist die digitale Bereitstellung von raumbezogene Geobasisdaten für Bildung, Verwaltung, Wirtschaft und private Nutzer durch die Vermessungs- und Katasterverwaltungen der einzelnen Bundesländer. Demnach werden derzeit Geobasisdaten in den Datenbeständen AFIS, ALKIS und ATKIS bereitgestellt. Dabei beschreibt das amtliche topographisch-kartographische Informationssystem , kurz ATKIS, die Oberfläche der Erde mit digitalen Landschafts- und Geländemodellen. Das amtliche Liegenschaftskatasterinformationssystem, kurz ALKIS, enthält Daten des amtlichen Liegenschaftskatasters und das amtliche Festpunktinformationssystem, kurz AFIS, alle Festpunktdaten wie zum Beispiel Lagefest- oder Höhenfestpunkte. Das AFIS-ALKIS-ATKIS-Modell, kurz AAA-Modell, soll dazu dienen die einzelnen Datenbestände der Informationssysteme zu einem Grunddatenbestand der Geodaten des amtlichen Vermessungswesens zusammenzuführen.

Das AFIS-ALKIS-ATKIS-Modell bildet das konzeptionelle Anwendungsschema für die Informationssysteme AFIS, ALKIS und ATKIS, die durch die Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV) eingeführt wurden. Das konzeptionelle Anwendungsschema wurde in der Modellierungssprache UML erstellt und besteht dabei aus dem Basisschema und dem Fachschema. Im Basisschema sind grundlegende Eigenschaften von geografischen Objekten beschrieben. Es kann damit auch als Grundlage bzw. konzeptionelle Sicht für die Entwicklung einer Datenverarbeitungsanwendung wie zum Beispiel einem Fachinformationssystem für AAA-Daten dienen. Im Fachschema wird die Gliederung von Objektklassen, Objektartgruppen, Objektarten und deren Attribute beschrieben. Es umfasst sämtliche im amtlichen Vermessungswesen aller Bundesländer vorkommenden Informationen in den Bereichen Liegenschaftskataster, Topographie/Kartographie und Grundlagenvermessung.

[8] siehe `http://www.adv-online.de/AAA-Modell`

Das konzeptionelle AAA-Modell beinhaltet neben dem Basisschema und dem Fachschema ein zusätzliches UML-Schema zur Definition der Operationen der normbasierten Austauschschnittstelle, kurz NAS. Die normbasierte Austauschschnittstelle wurde im Rahmen der Modellierung der Geoinformationssysteme AFIS, ALKIS und ATKIS definiert und stellt eine Datenschnittstelle für den Austausch von Geoinformationen im AAA-Modell dar.

6.1.3. DXF-Format

Das Drawing Interchange File Format, mit der Abkürzung DXF-Format, ist ein von der US-amerikanischen Firma Autodesk[9] spezifiziertes Dateiformat zum Austausch von CAD-Daten und wurde in seinen Anfängen in die CAD-Anwendung AutoCAD[10] integriert. Aufgrund der Tatsache, dass das DXF-Format genau beschrieben und offen dokumentiert ist, wird es fast ausschließlich für den anwendungsübergreifenden Datenaustausch auch über verschiedene Betriebssysteme hinweg verwendet. Als Quasi-Industriestandard beherrscht heute jede CAD-Anwendung den Im- und Export von CAD-Daten im DXF-Format. Das DXF-Format unterstützt alle Elemente, die für technische Zeichnungen sinnvoll und implementierbar sind. Darunter befinden sich unter anderem die Geometrietypen Punkt, Linie, Polylinie und Polygon.

Die Verwendung von CAD-Daten im DXF-Format in der Applikation *CADtoGIS Schnittstelle* erfolgt in den internen Komponenten *GenerateCadModel* und *RenderCadModel*. Hierbei wird ein CAD-Stadtmodell aus raumbezogenen Daten im DXF-Format erstellt und anschließend in der zentralen Komponente *ApplicationController* weiterverarbeitet und anwendungsintern ausgegeben bzw. dargestellt.

6.2. Verwendete Projektionen

Da in der Applikation *CADtoGIS Schnittstelle* geografische Daten in verschiedenen Datenformaten in Kombination mit CAD-Anwendungen verwendet werden, ist der Einsatz von bestimmten Projektionen dieser Daten bzw. der Einsatz von Koordinatentransformationen unabdingbar. Zunächst wird die weit verbreitete Mercator-Kartenprojektion detaillierter betrachtet. Anschließend wird auf die Verwendung der Projektion bei der Verarbeitung der vorliegenden raumbezogenen Daten in der Applikation eingegangen und näher diskutiert.

6.2.1. Mercator-Projektion

Die nach dem Kartografen Gerhard Mercator benannte Mercator-Projektion ist eine Form der Zylinderprojektion und ist insbesondere im Vermessungswesen sowie in der Navigation vorzufinden. Aufgrund der winkeltreue und der Tatsache, dass die Winkelmessung bei der

[9] siehe `http://www.autodesk.com`
[10] siehe `http://www.autodesk.com/products/autodesk-autocad/overview`

Navigation eine wesentliche Rolle spielt, wird die normale Mercator-Projektion häufig im Internet für Kartendienste eingesetzt. Das freie Projekt OpenStreetMap[11], welches frei nutzbare Geodaten sammelt und für alle Interessierten zur Verwendung bereitstellt, sowie einige kommerzielle Kartendienste verwenden vorzugsweise diese Projektionsart der geografischen Daten.

Die vereinfachte Erklärung der Konstruktion einer Mercator-Projektion ist die geometrische Projektion eines jeden Punktes auf der Erdkugel auf einen Zylinder, welcher um die Erdkugel gelegt wird und entlang des Äquators diese schneidet. Eine mathematisch detaillierte Erläuterung ist in einschlägiger Literatur zur Kartenprojektion zu finden (vgl. [Pea90]).

6.2.2. Projektion der Geometriedaten

In der Applikation *CADtoGIS Schnittstelle* wird eine geometrische Projektion von Koordinatenpunkten in zwei Fällen verwendet:

- **Projektion der AAA-Geobasisdaten im Esri-Shape-Format auf geografische Koordinaten der Stadt-Objekte**
- **Projektion der geografischen Koordinaten der Stadt-Objekte auf Koordinaten eines kartesischen Koordinatensystems**

Im ersten Fall erfolgt die Anwendung einer Projektion in der internen Komponente *GetGeoData*. Dabei werden Geodaten, welche vom Landesamtes für Geoinformation und Landentwicklung Baden-Württemberg, kurz LGL[12], bezogen werden und in einer AAA-Modell-konformen Struktur vorliegen, transformiert. Diese LGL-Geodaten, welche im Esri-Shape-Format gespeichert sind, werden in einem räumlichen Koordinatensystem für Vermessungszwecke gespeichert. Die Informationen über die tatsächlich verwendete Projektion, in der sich die LGL-Geodaten befinden, ist in der PRJ-Datei der Shape-Format-Dateistruktur zu finden (siehe 6.1.2). Um die LGL-Geodaten in geografische Koordinaten, welche von den Stadt-Objekten sowie OpenStreetMap verwendet werden, transformieren zu können, müssen explizit mit Hilfe von GeoTools die Projektionsdaten aus der PRJ-Datei ausgelesen werden. Anschließend werden die Geodaten-Geometriepunkte der SHP-Datei mit GeoTools in das weltweite zweidimensionale geodätische Referenzsystem mit der Bezeichnung EPSG:4326 (WGS 84)[13] transformiert.

Im zweiten Fall werden die geografischen Koordinaten der erzeugten Stadt-Objekte auf kartesische *xy*-Koordinaten, welche im Stadt-Objekt verknüpft werden und zur Erzeugung eines CAD-Stadtmodells dienen, abgebildet. Diese Abbildung bzw. Transformation der geografischen Punkte in ein kartesisches Koordinatensystem erfolgt mit Hilfe der normalen Mercator-Projektion. Dabei findet die Anwendung der normalen Mercator-Projektion in

[11] siehe `http://www.openstreetmap.org`
[12] siehe `http://www.lgl-bw.de`
[13] siehe `http://www.epsg-registry.org`

der internen Komponente *GetGeoData* statt. Diese Projektion wird verwendet, weil die geografischen Punkte, welche von OpenStreetMap bezogen werden und die Geometrie der Stadt-Objekte definieren, im sogenannte „World Geodetic System 1984“, abgekürtzt mit WGS 84, mit dem EPSG-Code 4326 liegen. Dieses weltweite zweidimensionale geodätische Referenzsystem, auf dessen Basis auch das Global Positioning System (GPS) arbeitet, ist das am weitesten verbreitete.

Das zweidimensionale geodätische Referenzsystem mit der Bezeichnung EPSG:4326 (WGS 84) wird von OpenStreetMap zur Speicherung raumbezogener Daten in der zentralen Datenbank verwendet. Es liegen demnach alle Koordinaten aus der OSM-Planet-Datei in diesem Koordinatensystem. Der Kartendienst von OpenStreetMap im Internet verwendet dagegen die mit einer gewissen Ungenauigkeit behaftete Pseudo-Mercator-Projektion, bei der die eigentliche Mercator-Projektion auf die rohen WGS-84 Koordinaten angewendet wird. Der Grund dafür ist, dass hier die geografischen Daten im geodätische Referenzsystem mit der Bezeichnung EPSG:3857 (WGS84/Pseudo-Mercator) liegen.

6.3. Verwendete Technologien

Für den Implementierungsteil dieser Arbeit kam die objektorientierte Entwicklungssprache Java SE 7 zum Einsatz. Die Anwendung wird als eigenständige *Java Application* erstellt und als ausführbares Java-Archiv (JAR) ausgeliefert. Voraussetzung für das Ausführen einer JAR bzw. eines Java-Programms ist eine installierte Java Laufzeitumgebung (engl. Java Runtime Environment). Im Folgenden werden die wichtigsten, mit Java verbundenen, Technologien für die Umsetzung der Anforderungen der Applikation *CADtoGIS Schnittstelle* erläutert.

6.3.1. Java Swing

Bei *Swing*[14] handelt es sich um eine Java-Grafikbibliothek mit deren Hilfe grafische Benutzeroberflächen innerhalb Java-Applikationen implementiert werden können. Swing ist Bestandteil der Java-Laufzeitumgebung und baut auf dem Abstract Window Toolkit (AWT) auf. Als leichtgewichtige UI-Komponenten werden Swing-Komponenten direkt von Java gerendert und sind nicht von nativen UI-Komponenten des Betriebssystems abhängig. Dies macht Swing beim Einsatz in Java-Anwendungen plattformunabhängig.

Relevanz für diese Arbeit

Die Grafikbibliothek Swing ist modular und objektorientiert aufgebaut und zudem plattformunabhängig. Dies ermöglicht einen vereinfachten und schnellen Einsatz für die Verwendung innerhalb der Applikation *CADtoGIS Schnittstelle*. Entsprechend wurde die Komponente *UserInterface* mit Swing realisiert.

[14] siehe `http://docs.oracle.com/javase/tutorial/uiswing/start/about.html`

6.3.2. JNetCAD/Java3D

JNetCAD[15] ist eine Java-basierte freie Klassenbibliothek zur Darstellung und Konvertierung von CAD-Modellen in einer Java-Applikation. Die Klassenbibliothek unterstützt verschiedene CAD-Formate, unter anderem das gängige DXF-Format zur Speicherung von CAD-Modellen in 2D oder 3D. Die Java-Klassenbibliothek liegt in Version 1.07 vor und kann für nicht-kommerzielle sowie Bildungszwecke eingesetzt werden. Die Darstellung bzw. das „Rendering" eines CAD-Modells erfolgt entweder mit Hilfe der 3D-Java-Klassenbibliothek Java3D[16], der OpenGL-Java-Klassenbibliothek JOGL[17] oder einer weiteren 3D-Java-Klassenbibliothek jReality[18] der TU-Berlin innerhalb einer Java-Applikation.

Relevanz für diese Arbeit

Die Verwendung von JNetCAD ist Bestandteil der Umsetzung der Anforderungen bezüglich der anwendungsinternen Darstellung eines erzeugten CAD-Stadtmodells in der Komponente *RenderCadModel*. Außerdem wird für die Darstellung die Rendering-Engine Java3D gewählt, da sich diese in das mit Swing realisierte Konzept der grafischen Benutzeroberfläche am besten integrieren lässt. Zusätzlich besteht hier die Möglichkeit einer Erweiterung der Anwendung hinsichtlich des Exports des erzeugten CAD-Stadtmodells in einem unterstützten CAD-Format.

6.3.3. Kabeja (DXF-Parser)

Bei *Kabeja*[19] handelt es sich um eine externe Klassenbibliothek zum Parsen, Verarbeiten und Konvertieren eines 2D/3D-CAD-Modells im gängigen DXF-Format innerhalb einer Java-Applikation. Die Klassenbibliothek ist in der Programmiersprache Java implementiert und liegt in der Version 0.4 vor und wird unter der Apache-Lizenz 2.0 vertrieben. Beim Parsen einer bestehenden DXF-Datei werden alle gängigen Geometrieelemente der DXF-Spezifikation unterstützt.

Relevanz für diese Arbeit

Die interne Komponente *DXF-Reader* wurde mit Hilfe der Java-Klassenbibliothek Kabeja realisiert. Diese Komponente liest mit Hilfe der externen Klassenbibliothek Kabeja lokal gespeicherte CAD-Modelle im DXF-Format aus und speichert diese in Stadt-Objekte.

[15] siehe `http://www.johannes-raida.de/jnetcad.htm`
[16] siehe `http://java.sun.com/javase/technologies/desktop/java3d`
[17] siehe `http://jogamp.org`
[18] siehe `http://www3.math.tu-berlin.de/jreality`
[19] siehe `http://kabeja.sourceforge.net`

Listing 6.1 Osmosis Area Extraction

```
osmosis \
--read-pbf enableDateParsing=no file=stuttgart.osm.pbf \
--bounding-box top=49.5138 left=10.9351 bottom=49.3866 right=11.201 \
--write-xml lglarea.osm
```

6.3.4. JMapViewer

JMapViewer[20] ist ein Werkzeug und besteht aus einer externen Java-Klassenbibliothek, die die OpenStreetMap-API verwendet, um innerhalb einer Java-Applikation eine einfache und schnelle Darstellung einer OpenStreetMap-Kartenansicht zu ermöglichen. JMapViewer basiert auf einer Erweiterung des Java OpenStreetMap Editors, kurz JOSM[21], und ist unter der GPL-Lizenz frei verfügbar.

Relevanz für diese Arbeit

Der Hauptbereich des Startfensters der Applikation wird mit Hilfe des JMapViewers implementiert. Dabei wird eine OpenStreetMap-Kartenansicht innerhalb einer Swing-basierten grafischen Benutzeroberfläche erzeugt sowie sämtliche Interaktionen mit dieser realisiert.

6.3.5. Osmosis

Bei *Osmosis*[22] handelt es sich um eine freie Java-Klassenbibliothek, die der Verarbeitung von OpenStreetMap-Daten innerhalb einer Java-Applikation dient. Unter anderem ist es mit Osmosis mögliche OSM-Dateien einzulesen, Daten aus einem rechteckigen Kartenausschnitt (engl. bounding box) oder einem Polygon zu extrahieren oder innerhalb des Datenbestands bestimmte Daten zu finden, zu extrahieren und daraus Java-Objekte zu erzeugen. Osmosis liegt in der Version 0.6 vor und ist frei verfügbar.

Relevanz für diese Arbeit

Mit Hilfe von Osmosis wird die interne Komponente *GetMap* realisiert. Dabei wird mit Osmosis eine lokal gespeicherte OSM-Planet-Datei auf einen gewählten geografischen Zielbereich verkleinert. Dazu wird die Osmosis-Anweisung, wie in Listing 6.1 zu sehen, innerhalb der Komponente *GetMap* ausgeführt. Ebenfalls wird mit Hilfe von Osmosis die erzeugte OSM-Datei eingelesen und in Stadt-Objekte überführt.

[20] siehe `http://wiki.openstreetmap.org/wiki/JMapViewer`
[21] siehe `http://wiki.openstreetmap.org/wiki/JOSM`
[22] siehe `http://wiki.openstreetmap.org/wiki/DE:Osmosis`

6.3.6. GeoTools

GeoTools[23] ist eine freie Java-Klassenbibliothek zur Bearbeitung und Darstellung geografischer Daten. GeoTools verwendet dabei die freie Java-Bibliothek *JTS Topology Suite*[24], welche ein räumliches Objektmodell bereitstellt und grundlegende Funktionen zur Verarbeitung von geometrischen Objekten bietet. Ebenfalls kann GeoTools durch verschiedene Plug-ins in seiner Funktionalität erweitert werden. Demnach ist es mit Hilfe von GeoTools möglich eine Java-Applikation mit einem Geoserver zu verbinden und auf einen bereitgestellten Web Feature Service (WFS) zuzugreifen, um Geoinformationen bzw. Geodaten für eine weitere Verarbeitung zu beziehen. GeoTools ermöglicht zudem das Auslesen und Verarbeiten von lokal gespeicherten Geodaten im Esri-Shape-Format. Die Java-Klassenbibliothek GeoTools liegt in der Version 10.2 vor und wird unter der LGPL-Lizenz zur freien Verwendung angeboten.

Relevanz für diese Arbeit

Mit Hilfe von GeoTools wird die interne Komponente *GetGeoData* realisiert. Dabei werden lokal gespeicherte Geodaten des AAA-Modells im Esri-Shape-Format ausgelesen und die enthaltenen Geometrien sowie raumbezogenen Metainformationen für die weitere Verarbeitung in der Komponente *ApplicationController* verwendet.

6.3.7. Geospatialmethods

Die Java-Klassenbibliothek *Geospatialmethods*[25] stellt Methoden für Berechnungen auf geografischen Koordinatensystemen bereit. Dabei werden Polygon-, Punkt- und Bounding-Box-Berechnungen sowie Vergleichsberechnungen innerhalb von Polygonen auf der Erdoberfläche effizient unterstützt. Geospatialmethods wird zur freien Verwendung angeboten.

Relevanz für diese Arbeit

Mit Hilfe von Geospatialmethods werden Berechnungen in einem geografischen Koordinatensystem für den Einsatz in der Komponente *ApplicationController* realisiert. Unter anderem werden mit Geospatialmethods die einzelnen Zugehörigkeiten der Stadt-Objekte zu bestimmten Nutzflächenbereichen ermittelt und für die Entwurfsoptimierung verwendet.

[23] siehe `http://geotools.org`
[24] siehe `http://tsusiatsoftware.net/jts/main.html`
[25] siehe `http://geospatialmethods.org`

6.3.8. Opt4J

Opt4J[26] ist eine Java-Klassenbibliothek und stellt eine modulares Framework für eine metaheuristische Problemoptimierung innerhalb einer Java-Applikation bereit. Zur Lösung von Optimierungsproblemen verwendet Opt4J unter anderem evolutionäre Algorithmen, Partikelschwarmoptimierung sowie Simulated Annealing. Das Ziel von Opt4J ist dabei die vereinfachte Anwendung der evolutionären Optimierung von anwendungsspezifischen Problemen sowie die Implementierung von beliebigen metaheuristischen Optimierungsalgorithmen. Zu diesem Zweck ist Opt4J modular aufgebaut, bietet optional einen grafische Benutzeroberfläche und visualisiert bei Bedarf den Optimierungsprozess. Opt4J liegt in der Version 3.0.1 vor und ist unter der LGPL-Lizenz frei zu beziehen.

Relevanz für diese Arbeit

Mit Hilfe von Opt4J wird die interne Komponente *OptimizeArea* und damit ein Teil der multidisziplinären Entwurfsoptimierung realisiert. Dabei wird mit Opt4J ein Opimierungsframework implementiert, das als Eingabe die vollständige Beschreibung des geografischen Stadtbereichs im Stadt-Objekt *UrbanArea* erhält.

6.3.9. AGG

Die Java-Klassenbibliothek *AGG*[27] der technischen Universität Berlin stellt eine Umgebung zur Erstellung und Transformation von zusammenhängenden Graphen dar. AGG ist dabei im Allgemeinen Sinne ein Graphersetzungssystem (engl. graph rewriting), das der formalen Beschreibung der Veränderung von Graphen dient. AGG ermöglicht den Einsatz eines Graphersetzungssystems innerhalb einer Java-Applikation und liegt in der Version 2.0.5 vor. AGG wird unter der GNU GPL-Lizenz zur freien Verfügung gestellt.

Relevanz für diese Arbeit

Mit Hilfe von AGG wird die interne Komponente *OptimizeArea* und damit ein Teil der multidisziplinären Entwurfsoptimierung realisiert. Dabei wird mit AGG ein Erzeugungsgraph, der auf einer bestimmten Designsprache aufbaut, erstellt und in das Optimierungsframework, welches mit Hilfe von Opt4J implementiert wird, integriert.

[26] siehe `http://opt4j.sourceforge.net/index.html`
[27] siehe `http://user.cs.tu-berlin.de/~gragra/agg`

6.4. CADtoGIS Schnittstelle User-Interface

Das Hauptziel bei der Entwicklung der grafischen Benutzeroberfläche der Applikation *CADtoGIS Schnittstelle* ist es eine möglichst gute Bedienbarkeit (engl. usability) zu erreichen. Das grafische User-Interface muss hierbei einheitlich in seiner Erscheinung sein und immer den gleichen Gestaltungsrichtlinien folgen. Um dieser Anforderung gerecht zu werden, ist es unabdingbar auf Technologien zu setzen, die weit verbreitet und plattformunabhängig sind.

Die Java-Grafikbibliothek Swing vereint diese Eigenschaften und erlaubt die universelle Nutzung der Applikation *CADtoGIS Schnittstelle*. Die Implementierung dieses grafischen User-Interfaces wurde als Teil dieser Diplomarbeit durchgeführt und im Anhang C grafisch illustriert.

Hauptfenster und Anzeige von Geodaten

Nach dem Start der Applikation wird als erstes das Hauptfenster eingeblendet. Hier sieht der Benutzer zuerst eine standardmäßig ausgewählte Kartenansicht in der Mitte des Fensters. Für den Start der Applikation muss der Benutzer keine Eingaben tätigen. Im Hauptfenster kann der Benutzer einen gewünschten Kartenbereich wählen und auf diesem bestimmten Aktionen durchführen (siehe Abbildung C.1).

Detaillierter kann der Benutzer im oberen Teil des Hauptfenster einen bestimmten Zielbereich direkt suchen und auf der Karte anzeigen lassen oder diesen in der Mitte des Fensters festlegen (siehe Abbildung C.2). Rechts neben der Kartenansicht befinden sich die möglichen Aktionen, die vom Benutzer durchgeführt werden können. Hier kann der gewählte Kartenbereich für weitere Aktionen festgelegt werden, ein rudimentäres CAD-Modell des Bereiches generiert werden, bestimmte Geodaten zur Anzeige ausgewählt und die multidisziplinäre Optimierung gestartet werden. Ebenfalls kann die Applikation hier beendet werden.

Nach der Auswahl der Schaltfläche „Geodaten wählen“ im Hauptfenster öffnet sich die Geodaten Auswahl. Hier kann der Benutzer bestimmte Geoinformationen zur Anzeige auswählen oder entfernen (siehe Abbildung C.3). Nach der Auswahl kehrt der Benutzer wieder zum Hauptfenster zurück und sieht die gewählten Geodaten im angezeigten Kartenausschnitt als Overlay. Die Abbildung C.4 und C.5 zeigen jeweils die Nutzflächenbereiche des Zielbereichs in verschiedenen Farben[28] unterteilt. Die Abbildung C.6 und C.7 zeigen Metainformationen in Form von Straßenkreuzungen und Eintrittspunkten in den urbanen Zielbereich.

Alle Benutzereingaben werden gemäß dem MVC-Architekturmuster an die interne Komponente *ApplicationController* der Applikation übergeben und dort verarbeitet.

[28] Legende: Rot=Wohngebiet, Grau=Industriegebiet, Braun=Mischgebiet, Grün=Grünfläche

CAD-Modell Anzeige

Nach der Auswahl der Schaltfläche „CAD-Modell“ im Hauptfenster öffnet sich die CAD-Modell Anzeige (siehe Abbildung C.8 und C.9). In der Mitte der CAD-Modell Anzeige sieht der Benutzer den im Hauptfenster ausgewählten Kartenbereich als generiertes CAD-Modell. Rechts neben der CAD-Modell Anzeige befindet sich die Schaltfläche zum Speichern des CAD-Modells sowie die zum Verlassen der CAD-Modell Anzeige. Weiterhin wird die Bearbeitung des CAD-Modells von einer externen CAD-Komponente übernommen. Diese kann aus jeder gängigen CAD-Anwendung wie zum Beispiel AutoCAD bestehen. Nach einer erfolgreichen Optimierung des Stadtbereiches wird das Ergebnis hier ebenfalls angezeigt.

Mögliche Funktionalität für weiterführende Versionen:
Rechts neben der CAD-Modell Anzeige sind Optionen für die CAD-Modellierung vorhanden, sodass der Benutzer interaktiv das rudimentäre CAD-Modell verändern kann.

Optimierungfenster

Nach der Auswahl der Schaltfläche „Optimierung“ im Hauptfenster öffnet sich die Anzeige zur Optimierung. Hier kann der Benutzer bestimmte Zielfunktionen, nach denen der Entwurf optimiert werden soll, auswählen und die Entwurfsoptimierung starten. Nachdem die Entwurfsoptimierung erfolgreich durchgelaufen ist, wird das Ergebnis in der CAD-Modell-Anzeige vom Benutzer visuell bewertet. Ebenfalls gibt es nach erfolgreichem Durchlauf der Optimierung die Möglichkeit das Ergebnis zu speichern. Dabei wird das Ergebnis als CAD-Modell im DXF-Format sowie als Geodaten im Esri-Shape-Format gespeichert.

6.5. CADtoGIS Schnittstelle ApplicationController

Die interne Komponente *ApplicationController* stellt die zentrale Hauptkomponente der Applikation *CADtoGIS Schnittstelle* dar. Diese Komponente verwendet alle anderen internen Komponenten und dient als zentrales Bindeglied und Kommunikationsknoten in der Applikation. Die Implementierung der wichtigsten Funktionen der Applikation findet in dieser Applikationskomponente statt und wird in den nächsten Abschnitten detailliert beschrieben.

6.5.1. Bezug von Geodaten

Der Bezug der Geodaten erfolgt jeweils in den internen Komponenten *GetMap* und *GetGeoData*, die beide von der Komponente *ApplicationController* verwendet werden. Dabei wird zunächst mit Hilfe der Klasse *GetMap* ein Kartenobjekt erstellt. Dieses *GetMap*-Objekt lädt in der Methode *parseOSMfile* raumbezogene Geometrien aus lokalen OSM-Dateien in eine Liste aus Knoten- und Weg-Objekten (siehe Listing B.1 im Anhang). Anschließend werden diese in der Methode *processOSMData* im *ApplicationController* in *UrbanObj*-Objekte überführt. Dies wird

im nächsten Abschnitt dieses Kapitels näher erläutert. Das *GetMap*-Objekt wird ebenfalls von der Komponente *UserInterface* zur Darstellung einer OpenStreetMap-Kartenansicht auf dem Startbildschirm der Applikation verwendet.

Weitere raumbezogenen Daten, die in die Erzeugung der *UrbanObj*-Objekte sowie in die Erstellung der Zielfunktion für die Entwurfsoptimierung einfließen, werden in der internen Komponenten *GetGeoData* aus lokal gespeicherten Esri-Shape-Dateien gelesen. Dazu wird mit Hilfe der Klasse *GetGeoData* ein *GetGeoData*-Objekt im *ApplicationController* erstellt. Dieses *GetGeoData*-Objekt ließt in der Methode *readShapeFile* die raumbezogenen Geometrien in Abhängigkeit der definierten Projektion aus (siehe Listing B.2 im Anhang). Anschließend werden in der Methode *generateUrbanData* des *ApplicationControllers* die erhaltenen Geometrien in die *UrbanObj*-Objekte gespeichert.

6.5.2. Erzeugung urbaner Objekte

Die Implementierung der Erzeugung des zentralen urbanen Objekts *UrbanArea* erfolgt in der Komponente *ApplicationController* und stellt eine Dekomposition des urbanen Zielbereichs dar. Das *UrbanArea*-Objekt beinhaltet alle im urbanen Zielbereich befindlichen Stadt-Objekte, welche auf dem logischen Datenmodell der Applikation basieren (siehe Klassendiagramme in Anhang A). Dabei werden die einzelnen Stadt-Objekte mit der Bezeichnung *UrbanObj* mit Hilfe der Methode *generateUrbanObjs* im *ApplicationController* erzeugt (siehe Listing B.3 im Anhang). Zunächst werden die einzelnen Stadt-Objekttypen *Street* und *Building* aus den OSM-Daten, die aus zwei Listen mit jeweils OSM-Knoten (engl. nodes) und OSM-Wegen (engl. ways) bestehen, erstellt. Hierbei werden aus den Daten der OSM-Wege mit Hilfe der enthaltenen „tags“ die Stadt-Objekttypen bestimmt und die geografische Geometrie durch Zuordnung der einzelnen OSM-Knoten, welche geografische Koordinatenpunkte darstellen, ermittelt. Anschließend wird die geografische Geometrie der Stadt-Objekte transformiert und mit dem Objekt verknüpft. Für die Transformation der geografischen Geometrie der *UrbanObjs* in ein CAD-konformes Koordinatensystem werden die *GetGeoData*-Methoden *tfGeoCoordToXy* und *tfXyToGeoCoord* verwendet. In diesen Methoden wird die Mercator-Projektion implementiert. Zusätzliche raumbezogenen Metainformationen, wie zum Beispiel Straßen-, Gebäudenamen und die entsprechenden Objekttypen, werden ebenfalls in der Methode *generateUrbanObjs* in die Stadt-Objekte geschrieben. Dabei werden unter anderem die Geodaten der AAA-Geobasisdaten des LGL im Esri-Shape-Format verwendet, um die Nutzflächeninformationen der einzelnen Stadt-Objekte zu beziehen. Dazu werden Methoden der externen Komponente *Geospatialmethods* verwendet, um die geografische Zugehörigkeit innerhalb eines Nutzflächenpolygons zu ermitteln. Die finale Zusammenführung aller benötigter Informationen für das *UrbanArea*-Objekt findet im *ApplicationController* in der Methode *generateUrbanData* statt.

Nach der erfolgreichen Erzeugung des zentralen urbanen Objekts *UrbanArea* kann dieses Objekt zur Erzeugung und Darstellung eines CAD-Stadtmodells verwendet werden. Ebenfalls wird dieses Objekt zur multidisziplinären Entwurfsoptimierung verwendet.

6.5.3. Erzeugung eines CAD-Stadtmodells

Die Erzeugung eines CAD-Stadtmodells aus dem zuvor erstellen zentralen *UrbanArea*-Objekt wird in der Methode *writeUrbanArea2dModel* der Klasse *GenrateCadModel* implementiert. Nach dem im *ApplicationController* eine *GenrateCadModel*-Objekt instanziiert worden ist, wird mit Hilfe dieses und dem zentralen *UrbanArea*-Objekt die Erstellung eines CAD-Stadtmodells in der Methode *generateUrbanCadModel* angestoßen. Dabei wird die Geometrie jedes enthaltenen Stadt-Objekts aus dem *UrbanArea*-Objekt gelesen und die Straßen- bzw. Gebäudegeometrien in verschiedene Layer des CAD-Modell-Ausgabeformats DXF gespeichert. Nach einer erfolgreichen Entwurfsoptimierung können diese Geometrien anhand der Layer wieder in ein *UrbanArea*-Objekt überführt werden.

6.5.4. Darstellung eines CAD-Stadtmodells

Die Darstellung eines zuvor erzeugten CAD-Stadtmodells wird in der internen Komponente *RenderCadModel*, welche vom *ApplicationController* verwendet wird, implementiert. Um die Darstellung des CAD-Stadtmodells zu starten, wird im *ApplicationController* ein *RenderCadModel*-Objekt instanziiert und das 3D-Darstellungsfenster für die Komponente *UserInterface* bezogen. Dabei wird das von der internen Komponente *GenrateCadModel* gespeicherte CAD-Stadtmodell im DXF-Format von der *RenderCadModel*-Methode *getCanvas3D* gelesen und dargestellt. Diese Implementierung bietet in ihrer ersten Ausführung noch keine Funktionalität zur Manipulation des CAD-Stadtmodells.

6.6. CADtoGIS Schnittstelle OptimizeArea

In der internen Komponente *OptimizeArea* wird die Implementierung der eigentlichen Entwurfsoptimierung realisiert. Die theoretische Grundlage hierzu wurde im Kapitel 5 beschrieben. Hierbei wird ein *OptimizeArea*-Objekt im *ApplicationController* instanziiert und mit Hilfe dessen eine Entwurfsoptimierung angestoßen.

Dabei wird mit Hilfe von Opt4J das eigentliche Opimierungsframework, das als Eingabe die vollständige Beschreibung des geografischen Stadtbereichs im Stadt-Objekt *UrbanArea* erhält, implementiert. Hierbei wird in der Klasse *OptimizeArea* ein Objekt *optFramework* für das Opimierungsframework erstellt und mit dessen Hilfe der Optimierungsprozess der Entwurfsoptimierung realisiert.

Die Implementierung des im Opimierungsframework enthaltenen Graphersetzungssystems wird mit Hilfe von AGG realisiert. Dabei wird in der Klasse *OptimizeArea* ein Graph-Objekt aufgebaut, das mit fest definierten Erzeugungsregeln arbeitet. Dieses Graph-Objekt dient als Dateninput in die *OptimizeArea*-Methode *optPorcess*, welche ebenfalls das Objekt *optFramework* verwendet. Die Methode *optPorcess* erstellt neue *UrbanArea*-Objekte mit modifizierten Erzeugungsgraphen und bewertet diese hinsichtlich der gegebenen Zielfunktionen. Das gefundene

Ergebnis wird an den *ApplicationController* über die *OptimizeArea*-Methode *getOptResult* übergeben. Das Ergebnis stellt hierbei ein neu erzeugtes und optimiertes *UrbanArea*-Objekt dar. Abschließend wird das Optimierungsergebnis im *ApplicationController* mit Hilfe der Methoden zur Darstellung eines urbanen Zielbereich als CAD-Stadtmodell dem Städtebauarchitekten präsentiert.

6.7. Schwierigkeiten bei der Implementierung

Dieser Abschnitt des Kapitel 6 erläutert einige Schwierigkeiten, die während der Implementierung aufgekommen sind. Ebenfalls werden die gefundenen Lösungen diskutiert und näher beschrieben.

Die Implementierung der internen Komponente *GetMap* wurde unter der Verwendung der OpenStreetMap-API sowie des OSM-Formats realisiert. Da es sich bei OpenStreetMap um ein oft verwendetes freies Projekt handelt, wurde angenommen, dass die Dokumentation des OSM-Formats von einer zufriedenstellenden Qualität ist. Bei näherer Recherche hat sich aber herausgestellt, dass die verwendete Dokumentation trotz neuester Dokument-Version in Bezug auf die Verwendete OpenStreetMap-API lückenhaft ist. Demnach waren nicht alle bereits integrierten Auszeichnungen des OSM-Formats in der Dokumentation vorhanden. Als Lösung wurde hier die strikte Verwendung der Grundfunktionalität der OpenStreetMap-API sowie des OSM-Formats in Betracht gezogen.

Eine weiteres Problem kam bei der Implementierung der Komponente *GenrateCadModel* zur Erzeugung eines CAD-Stadtmodells auf. Im Laufe der Recherche zu möglichen externen Komponenten, die zur Implementierung einer CAD-Engine verwendet werden könnten, kamen die externen Klassenbibliotheken *OpenSCAD*, *JavaSCAD* und *Ycad* in Betracht. Dabei sollte die gewählte Klassenbibliothek den definierten Anforderungen entsprechen und in der Lage sein 2D- und 3D-Geometrieobjekte in einem CAD-konformen Format zu erzeugen sowie die Funktionalität der Manipulation dieser bieten. Aufgrund der Tatsache, dass die Klassenbibliothek *OpenSCAD* auf der Programmiersprache C++ basiert und nur als Kommandozeilenanwendung in Betracht gekommen wäre, wurde diese nicht verwendet. Eine Alternative ist die in Java implementierte Klassenbibliothek *JavaSCAD*, welche auf der Funktionalität von *OpenSCAD* aufbaut und die Erzeugung einer Vielzahl von 3D-Geometrien erlaubt. Ein nicht unerheblicher Nachtteil dieser Java-Klassenbibliothek ist, dass diese in einer sehr frühen Version zur Verfügung steht und noch keine 2D-Geometrien unterstützt. Aufgrund dessen wurde *JavaSCAD* ebenfalls nicht verwendet. Die dritte Alternative war die Java-Klassenbibliothek *Ycad*, welche der ersten Untersuchung nach die gewünschten Anforderungen erfüllte. Nichtsdestotrotz war diese Klassenbibliothek zu veraltet für einen Einsatz in der Applikation. Als Lösung wurde daher ein einfacher Ansatz bestehend aus der Java-Klassenbibliothek *Kabeja* sowie der Verwendung des gängigen CAD-Austauschformats DXF verwendet.

Ebenso gestaltete sich die Suche nach einer passenden Rendering-Engine für CAD-Modelle im DXF-Format als schwierig. Dabei wurden zwar einige kommerzielle Java-Klassenbibliotheken,

aber nur sehr wenige Alternative zur freien Verfügung gefunden. Anfänglich wurde die alternative Java-Klassenbibliothek *JavaView* zur Darstellung von gängigen CAD-Datenformaten vorgesehen. Bei genauerer Untersuchung stellte sich heraus, dass hier die Unterstützung für das DXF-Format nicht vollständig integriert ist. Daher wurde der Einsatz von *JavaView* verworfen und die alternative Java-Klassenbibliothek *JNetCAD/Java3D* eingesetzt. Die Verwendung dieser hat sich als eine einfache wie auch effektive Lösung herausgestellt.

6.8. Beispielhafte Anwendung der CADtoGIS Applikation

In diesem Unterabschnitt werden die Ergebnisse der Implementierung in einem ausgewählten Beispieldurchlauf präsentiert. Das Verhalten der Applikation *CADtoGIS Applikation* wird schrittweise aufgezeigt und analysiert. Zunächst folgen eine genauere Beschreibung des gewählten Beispiel-Use-Cases und anschließend der eigentliche Durchlauf.

6.8.1. Beschreibung eines Beispiel-Use-Cases

Bei dem gewählten Beispiel-Use-Case handelt es sich um den Start einer multidisziplinären Entwurfsoptimierung auf Basis der bezogenen Geodaten. Dabei startet der Städtebauarchitekt die Applikation *CADtoGIS Schnittstelle*, wählt einen Zielbereich und stößt anschließend die Entwurfsoptimierung an.

Da die Applikation einem Prototypen entspricht, kann es vorkommen, dass bei bestimmten Benutzereingaben kein Fehlerstatus zurückgegeben wird. Damit die Anzahl von auftretenden Sonderfällen gering gehalten werden kann, müssen einschränkende Annahmen gewählt werden. Der Städtebauarchitekt beschränkt sich auf die konkrete Durchführung des gewählten Beispiel-Use-Cases. Außerdem ist die Auswahl des geografischen Zielbereichs auf einen genau definierten Bereich festgelegt. Der Grund hierfür ist das kostenpflichtige Angebot der Geodaten des LGL. Dieses stellt raumbezogene Daten nur für eine Fläche bis 1 km^2 kostenfrei zur Verfügung. Diese getroffenen Maßnahmen garantieren zudem die Übersichtlichkeit des Durchlaufs.

Use-Case Durchlauf

Das Aktivitätsdiagramm aus Abbildung 4.4 in Kapitel 4.6 wird zur Orientierungshilfe für den Testdurchlauf herangezogen. Dabei wird ebenfalls vereinfacht das Zusammenspiel der einzelnen Applikationskomponenten aufgezeigt. Beginnend mit dem Applikationsstart wird in den aufgelisteten Ausführungsschritten ein kompletter Anwendungsdurchlauf dargestellt. Neben der strikten Ausführung des Beispiel-Use-Cases wird an passender Stelle auch ein kurzer Einblick auf alternative Anwendungsfälle gegeben. Dadurch kann der Beispiel-Use-Case besser in den Gesamtzusammenhang eingeordnet werden.

Ausführungsschritt	Beschreibung
`01. Programmstart`	Hierdurch wird die Applikation gestartet und der Startbildschirm, der aus dem Hauptfenster der Applikation besteht, angezeigt. Das Universitätsgelände der Universität Stuttgart ist als initialer Bereich in der Kartenansicht eingestellt (siehe Abbildung C.1 im Anhang C).
`02. Zielbereich wählen`	Der Städtebauarchitekt wählt einen geografischen Zielbereich indem er die Schaltfläche „Testbereich“ auswählt. Dabei wird der vordefinierte urbane Zielbereich automatisch gewählt und in der Kartenansicht angezeigt. Falls Geodaten des ganzen Bundeslandes zur freien Verfügung stehen, kann der Städtebauarchitekt hier einen beliebigen urbanen Zielbereich wählen indem er einen rechteckigen Bereich auf der Kartenansicht mit Hilfe von zwei gewählten Punkten definiert (siehe Abbildung C.2 im Anhang).
`03. Zielbereich festlegen`	Nachdem der Zielbereich gewählt worden ist, müssen alle benötigten Daten für diesen Zielbereich initialisiert werden. Dabei werden Geodaten bezogen, alle Stadt-Objekte erzeugt sowie Informationen zur Definition der Zielfunktionen für die bevorstehende Entwurfsoptimierung erstellt. Für diese Aufgabe wählt der Städtebauarchitekt die Schaltfläche „Bereich festlegen“ aus und wartet ab bis die Daten des Zielbereichs initialisiert sind.
`04. Geodaten anzeigen`	Dieser Ausführungsschritt kann optional ausgeführt werden und ist nicht für die Ausführung des Beispiel-Use-Cases zwangsläufig notwendig. In diesem Schritt kann der Städtebauarchitekt die bezogenen Geodaten in der Kartenansicht anzeigen lassen, um eine visuelle Bewertung des gewählten urbanen Zielbereichs durchführen zu können. Hierbei wählt er die Schaltfläche „Geodaten wählen“ und anschließend die gewünschten Daten, welche angezeigt werden sollen, aus (siehe Abbildung C.3 im Anhang C). Nach dem Speichern der gewählten Geodaten kehrt der Städtebauarchitekt zurück auf das Hauptfenster, welches die Darstellung der Daten bereitstellt (siehe Abbildung C.4 bis C.7 im Anhang C).

`05. CAD-Stadtmodell anzeigen`	Dieser Ausführungsschritt kann optional ausgeführt werden und ist nicht für die Ausführung des Beispiel-Use-Cases zwangsläufig notwendig. In diesem Schritt kann der Städtebauarchitekt den gewählten Zielbereich als CAD-Stadtmodell darstellen lassen. Damit kann er diesen im Hinblick auf die Entwurfsoptimierung visuell auf bestimmte Aspekte bewerten. Dazu wählt er die Schaltfläche „CAD-Modell“ und gelangt zum Darstellungsfenster des CAD-Modells (siehe Abbildung C.8 und C.9 im Anhang C). In dieser Ansicht kann der Städtebauarchitekt das erzeugte CAD-Stadtmodell getrennt mit den Geodaten speichern.
`06. Zielfunktion wählen`	Nach der Initialisierung des Zielbereichs kann eine multidisziplinäre Entwurfsoptimierung für diesen gestartet werden. Dafür wählt der Städtebauarchitekt die Schaltfläche „Optimierung“ aus und gelangt in das Optimierungsfenster, welches die Auswahl der möglichen Zielfunktionen für die Entwurfsoptimierung bietet.
`07. Optimierung starten`	Nachdem der Städtebauarchitekt die gewünschten Zielfunktionen markiert hat, wählt er die Schaltfläche „Optimierung starten“ aus und stößt damit die eigentliche multidisziplinäre Entwurfsoptimierung des urbanen Zielbereichs an. Anschließend wird ein Informationsfenster zur Optimierung eingeblendet.
`08. Optimierungsergebnisse anzeigen und speichern`	Nach erfolgreicher Optimierung wird im Optimierungsfenster die Option zur Anzeige und Speicherung der Ergebnisse der Entwurfsoptimierung angezeigt. Wählt der Städtebauarchitekt die Schaltfläche „Ergebnis anzeigen“ aus, wird das Ergebnis der Entwurfsoptimierung als CAD-Stadtmodell dargestellt und kann hier ebenfalls gespeichert werden. Andernfalls wählt der Städtebauarchitekt direkt die Schaltfläche „Ergebnis speichern“, um dieses getrennt in CAD-Modell und Geodaten zu speichern. Anschließend kehrt der Städtebauarchitekt zum Hauptfenster zurück.

`09. Programm beenden`	Nach erfolgreicher Anzeige und Speicherung kann der Städtebauarchitekt die Applikation *CADtoGIS Schnittstelle* im Hauptfenster beenden indem er die Schaltfläche „Beenden“ auswählt.

Tabelle 6.1.: Ausführungsschritte beim Testdurchlauf des Beispiel-Use-Cases: Optimierung starten

7. Evaluation

Dieses Kapitel befasst sich mit der Durchführung und Dokumentation der Evaluation der Applikation *CADtoGIS Schnittstelle*. Software-Evaluation ist per Definition die Analyse und Bewertung von Software-Systemen. Die Evaluation untersucht unter anderem welches Leistungsspektrum durch das betrachtete Softwareprodukt geboten wird sowie in welcher Qualität das Zusammenwirken einzelner Softwarebestandteile erfolgt. Ergänzt wird die Evaluation um eine Betrachtung der Benutzerschnittstelle. Hierbei wird analysiert und bewertet inwiefern ein Benutzer der evaluierenden Software auf Schwierigkeiten bei der Lösung eines spezifischen Problems mit der Software stößt. Ziel der Evaluation ist es aus der Analyse eine Beschreibung zu erstellen, aus der zum einen die Eignung für den Einsatz der Software im spezifizierten Anwendungsbereich und zum anderen der Umfang und die Qualität der Unterstützung der Problemlösung durch die evaluierte Software hervorgeht.

7.1. Evaluationsdurchführung

Die Evaluation ist per Definition abhängig vom Zeitpunkt ihrer Ausführung. Dementsprechend wird unterschieden zwischen der formativen Evaluation, also der Überprüfung während der Software-Entwicklung und der summativen Evaluation, der Bewertung eines bereits bestehenden Software-Produkts (siehe Abbildung 7.1) (vgl. [PRS+94]).

Im Fall dieser Diplomarbeit wird eine summative Evaluation mit dem konkreten Ziel der Überprüfung der implementierten Applikation bezüglich der definierten Anforderungen durchgeführt. Abhängig von diesem Ziel ergeben sich hier unterschiedliche Kriterien, die in die Evaluation einfließen und festlegen, was und in welchem Umfang evaluiert werden soll. Die Evaluationskriterien der implementierten Applikation *CADtoGIS Schnittstelle* sind hierbei unter anderem die Akzeptanz der Software bei den Städtebauarchitekten, die Erlernbarkeit, Bedienbarkeit, Wartbarkeit, Austauschbarkeit der Softwarekomponenten sowie die Innovationsförderung bei der Stadtplanung.

7.1.1. Verwendetes Vorgehensmodell

Das verwendete Vorgehensmodell zur Software-Evaluation basiert auf der Grundlage eines Modells, welches einen Wirklichkeitsausschnitt in dem die betrachtete Software eingesetzt werden soll, darstellt. Im Falle dieser Arbeit besteht dieses Modell aus dem Entwurfsprozess eines möglichst optimalen Stadtplans in den der Städtebauarchitekt einbezogen wird. In Abhängigkeit von diesem Modell findet dann die Untersuchung der Software im Hinblick

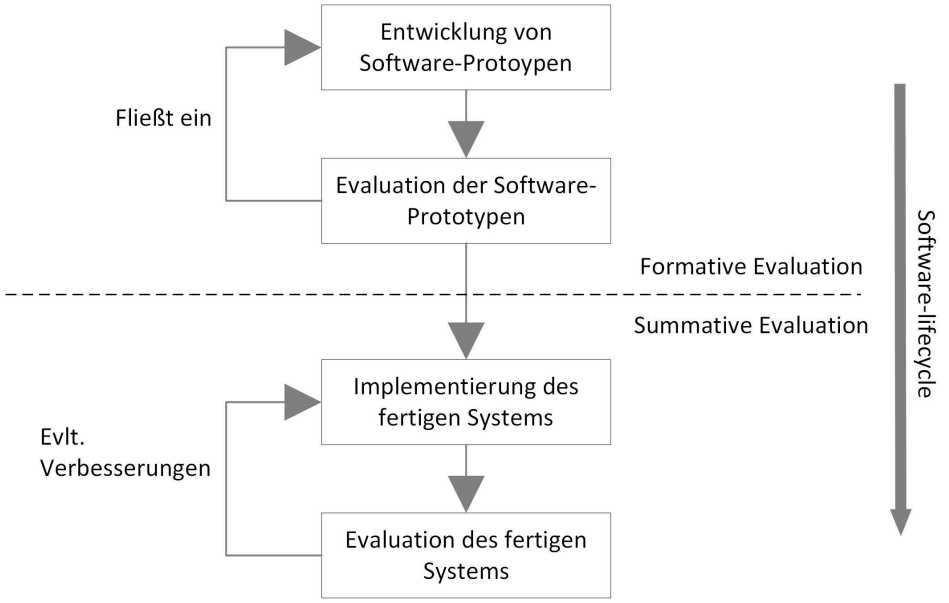

Abbildung 7.1.: Formative und summative Evaluation

auf die festgelegten Evaluationskriterien statt. Diese Untersuchung gliedert sich dabei in eine quantitative Evaluation, bei der die Bestimmung des Leistungsumfangs sowie die Aufdeckung der durch die Software gebotenen Unterstützungsmittel im Vordergrund steht, und in eine qualitative Evaluation, bei der die Art und Weise der Software-Unterstützung bei der Problemlösung untersucht wird.

7.1.2. Darstellung der Evaluationsergebnisse

Die Darstellung der Evaluationsergebnisse wird eingeteilt in die jeweiligen Anforderungen der Applikation *CADtoGIS Schnittstelle* und erfolgt textuell. Dabei wird zunächst die Anforderung als Analyseziel detaillierter beschrieben und anschließend der Ablauf der eigentlichen Evaluation im Hinblick auf das Analyseziel erläutert. Weiterhin werden zusätzlich die bei der Evaluation involvierten Anwendungsfälle (engl. use-cases) aufgelistet. Um schließlich eine Schlussfolgerung der Analyse zum betrachteten Evaluationsteil im Hinblick auf die definierte Anforderung ziehen zu können, wird zu erst eine Bewertung des durchgeführten Evaluationsteils erstellt.

7.2. Evaluation

In diesem Abschnitt wird die Applikation *CADtoGIS Schnittstelle* auf die erfolgreiche Umsetzung der einzelnen Anforderungen hinsichtlich der Evaluationskriterien analysiert und im Anschluss bewertet. Im Folgenden erfolgt eine Auflistung der Evaluationsdurchführung.

Evaluation: Anforderung 1

Anwender: Städtebauarchitekt

Anforderung: Austauschbarkeit der Programmkomponenten der Applikation *CADto-GIS Schnittstelle*

Beschreibung: Die externen Programmkomponenten zur Anzeige, Bezug, Verarbeitung und Optimierung von Geodaten und CAD-Daten sollen optional ausgetauscht bzw. ersetzt werden können.

Ablauf der Evaluation: Beim Start der Applikation werden die Geometrie- und Metainformationen der notwendigen raumbezogenen Daten aus OSM-Dateien, bzw. der gepackten Variante PBF-Dateien, und Esri-Shape-Dateien gelesen. Nach erfolgter Verarbeitung werden die Daten, welche die raumbezogenen Metainformationen enthalten, in Esri-Shape-Dateien und die Geometriedaten in DXF-Dateien gespeichert.

Enthaltene Use-Cases: Programmstart, Kartenbereich festlegen, Optimierung starten, Ergebnis der Optimierung speichern

Bewertung: Die Austauschbarkeit der Applikationskomponente für den Bezug von Geodaten sowie einer Kartenansicht ist relativ schlecht, da hier das OSM-Format von OpenStreetMap verwendet wird. Dieses Format wird ausschließlich von OpenStreetMap verwendet. Alle anderen verwendeten Datenformate können von zahlreichen Softwaresystemen verarbeitet werden und gewährleisten eine sehr gute Ersetzbarkeit der jeweiligen Applikationskomponenten.

Schlussfolgerung: Die meisten verwendeten Datenformate sind gängige und weitverbreitete Datenaustauschformate. Diese Daten können mit externen Entwurfsanwendungen verarbeitet werden. Folglich gewährleisten diese, dass die Komponenten zur Beziehung und Speicherung der raumbezogenen Daten durch anderweitige Softwarekomponenten ausgetauscht werden können.

Evaluation: Anforderung 2

Anwender: Städtebauarchitekt

Anforderung: Bezug und Anzeige von Geoinformationen

Beschreibung: Die Applikation soll Geoinformationen aus bestimmten Quellen beziehen, verarbeiten und darstellen können. Die Darstellung soll auf einer Kartenansicht erfolgen.

Ablauf der Evaluation: Nach dem Start der Applikation wird das Hauptfenster angezeigt auf dem sich eine Kartenansicht von OpenStreetMap befindet. Der Städtebauarchitekt wählt nun Geodaten aus indem er die Schaltfläche „Geodaten wählen“ betätigt und bestimmte Geodaten wählt. Nach verlassen dieses Fensters werden die gewählten Geodaten in der Kartenansicht angezeigt. Der Städtebauarchitekt

bewertet den Zielbereich anhand der angezeigten Daten und startet bei positiver Bewertung die Entwurfsoptimierung.

Enthaltene Use-Cases: Programmstart, Kartenbereich Suche, Kartenbereich festlegen, Geodaten anzeigen

Bewertung: Der Bezug und die Anzeige der Geodaten ermöglichen es dem Städtebauarchitekten bereits vor der eigentlichen Optimierung den Zielbereich visuell zu erfassen und zu bewerten. Die Bedienbarkeit der Applikation wird durch die visuelle Auswahl der gesuchten Daten vereinfacht.

Schlussfolgerung: Im Hinblick auf die geeignete Auswahl eines urbanen Zielgebietes sowie der Integration dieses in ein bestehendes Stadtgebiet ist der Bezug und die Anzeige der Geoinformationen von hoher Bedeutung.

Evaluation: Anforderung 3

Anwender: Städtebauarchitekt

Anforderung: Erzeugung eines CAD-Stadtmodells aus den Geodaten

Beschreibung: Die Applikation soll eine automatische Erzeugung eines CAD-Stadtmodells anhand der bezogenen Geodaten bzw. Geoinformationen bereitstellen.

Ablauf der Evaluation: Nach dem Start der Applikation und der Auswahl des Zielbereiches lässt sich der Städtebauarchitekt den gewählten urbanen Bereich als CAD-Stadtmodell anzeigen. Dabei wählt er die Schaltfläche „CAD-Modell“ im Hauptfenster aus.

Enthaltene Use-Cases: Programmstart, Kartenbereich festlegen, CAD-Modell anzeigen

Bewertung: Die Erzeugung eines CAD-Stadtmodells aus den bezogenen Geodaten ist im Hinblick auf die Akzeptanz der Applikation bei Städtebauarchitekten ein wichtiger Punkt, da hier eine visuelle Bewertung des Zielbereichs bzw. des optimierten Zielbereichs erfolgt.

Schlussfolgerung: Im Hinblick auf die planerische Bewertung eines urbanen Gebiets ist dessen Darstellung als CAD-Modell eine gute Unterstützung für den Städtebauarchitekten.

Evaluation: Anforderung 4

Anwender: Städtebauarchitekt

Anforderung: Verknüpfung von Geodaten und CAD-Stadtmodell

Beschreibung: Die Applikation soll eine konsistente Kopplung der vorliegenden Geodaten mit dem erzeugten CAD-Stadtmodell ermöglichen.

Ablauf der Evaluation: Nach dem Start der Applikation und der Auswahl des Zielbereiches kann vom Städtebauarchitekten die Ausführung der Entwurfsoptimierung gestartet werden. Die Entwurfsoptimierung verwendet dabei nach dem logischen Datenmodell der Applikation *CADtoGIS Schnittstelle* definierte urbane Objekte, welche die bezogenen Geodaten und CAD-Geometrie verbinden. Dies wird dem Städtebauarchitekt bei der Speicherung des Ergebnisses der Entwurfsoptimierung verdeutlicht.

Enthaltene Use-Cases: Programmstart, Kartenbereich festlegen, Optimierung starten, Ergebnis der Optimierung speichern

Bewertung: Bei der Speicherung der Ergebnisdaten werden jeweils raumbezogene Metainformationen und Geometriedaten gespeichert. Dies zeigt die Kopplung der Daten auf und erhöht den Einsatzbereich der Applikation als unterstützendes Softwarewerkzeug beim Entwurfsprozess eines Stadtplans.

Schlussfolgerung: Die Kopplung von Geodaten und CAD-Stadtmodell verdeutlicht den Schnittstellencharakter der Applikation und steigert damit den Nutzen und die Akzeptanz der Applikation beim Städtebauarchitekten.

Evaluation: Anforderung 5

Anwender: Städtebauarchitekt

Anforderung: Optimierung von CAD-Modell und Geodaten

Beschreibung: Die Applikation soll unter anderem die Einspeisung der erzeugten Daten in ein Optimierungsframework gewährleisten. Außerdem soll die Optimierung des CAD-Modells anhand der bezogenen Geodaten und anhand von festgelegten Zielfunktionen erfolgen. Insofern sollen beide Daten aufgrund der Kopplung von Geodaten und CAD-Modell optimiert werden.

Ablauf der Evaluation: Nach dem Start der Applikation und der Auswahl des Zielbereiches startet der Städtebauarchitekt die multidisziplinäre Entwurfsoptimierung durch Auswahl der Schaltfläche „Optimierung“. Nach Auswahl einer Zielfunktion wird die eigentliche Optimierung gestartet und neue Stadtplanentwürfe bei Berücksichtigung der bezogenen Geodaten, der zuvor erzeugten Geometrie sowie der gewählten Zielfunktion erzeugt. Anschließend bewertet der Städtebauarchitekt das optimierte Ergebnis der Daten und verwendet dieses gegebenenfalls zur Entwurfserstellung.

Enthaltene Use-Cases: Programmstart, Kartenbereich festlegen, Optimierung starten, Ergebnis der Optimierung speichern, CAD-Modell anzeigen, Ergebnis der Optimierung anzeigen

Bewertung: Die Optimierung eines geografischen urbanen Zielbereichs wird mit den Komponenten der Applikation ergebnisorientiert umgesetzt und stellt dem Städtebauarchitekten eine nützliche Unterstützung zur Problemlösung bereit.

Schlussfolgerung: Die Optimierung eines geografischen urbanen Zielbereichs ist im Hinblick auf die ungeplante Entwurfsfindung ein wichtiger Teil bei der Innovationsförderung beim Stadtentwurf.

7.3. Schlussfolgerungen

Die durchgeführte Software-Evaluation der Applikation *CADtoGIS Schnittstelle* lässt die Schlussfolgerung zu, dass die definierten Anforderungen in einer für den Städtebauarchitekten nützlichen Art und Weise umgesetzt worden sind. Die kritischen Komponenten der Applikation erfüllen ihre Soll-Aufgabe im Hinblick auf die geforderte Funktionalität und tragen einen großen Teil zur Beantwortung der gesamten Problemstellung dieser Arbeit bei. Weiterhin wurde durch die Evaluation der Anforderung „Optimierung von CAD-Modell und Geodaten“ geprüft, dass die verwendeten Komponenten der Applikation erwartungsgemäß zusammenspielen. Schließlich hat die Evaluation auch gezeigt, dass der gesamte Leistungsumfang der Applikation dem durch die Anforderungen definierten entspricht. Demnach wurden alle gebotenen Applikationsteile zur Unterstützung der Problemlösung in ihrer korrekten Funktionsweise bestätigt.

In einer erweiterten Implementierung könnte die Bearbeitung des erzeugten CAD-Stadtmodells das Zusammenspiel zwischen der Optimierungskomponente und der Komponenten zur Anzeige des CAD-Stadtmodells konsistenter im Entwurfsprozess machen. Dies ist in der aktuellen Version jedoch nicht vorgesehen.

8. Zusammenfassung und Ausblick

Das letzte Kapitel dieser Diplomarbeit fasst die theoretischen und praktischen Erkenntnisse, die während der Arbeit gesammelt worden sind, zusammen und gibt einen Ausblick für zukünftige Entwicklungen im Einsatzbereich der Problemstellung dieser Diplomarbeit. Im Folgenden wird zunächst eine ausführliche Zusammenfassung der bisherigen Kapitel bereitgestellt. Hierbei wird unter anderem auf die gewonnenen Resultate des Lösungsansatzes mit Blick auf die Problemstellung eingegangen sowie die Vor- und Nachteile beim Einsatz der gefundenen Lösung herausgearbeitet. Bevor abschließend ein Ausblick die möglichen Potentiale der gefundenen Lösung für die zukünftige Forschung und Anwendung im Bereich der Städteplanung beleuchtet, werden die voraussichtlichen Auswirkungen der Lösung auf das praktische Problem des Stadtplaners vorgestellt.

8.1. Zusammenfassung der Arbeit

In Zeiten immer komplexer werdender städtischer, technischer sowie gesellschaftlicher Infrastrukturen besteht in der Architektur, wie auch in der Stadtplanung, eine hohe Notwendigkeit für computerbasierte Entwurfsmethoden. In der Stadtplanung finden sich dabei eine Vielzahl von einander abhängiger und zu einander in Konkurrenz stehender Aspekte wieder, welche im Entwurfsprozess einer Stadt eine wichtige Rolle spielen und in Einklang gebracht werden sollen. Das Ziel des Entwurfsprozesses ist dabei im Hinblick auf die Verwendung der zur Verfügung stehenden Bodenfläche eine homogene und nachhaltige Planung der Verteilung benötigter Stadtformen. Dabei soll diese Verteilung in Bezug auf die entstehenden Stadtstrukturen den wichtigsten technischen sowie sozialen Aspekten einer Stadt möglichst optimal gerecht werden.

Dieses zentrale Problem einer multidisziplinären Entwurfsoptimierung in der Stadtplanung wurde in der Vergangenheit fast immer fallspezifisch für bestimmte Stadtbereiche oder Städte gelöst. Als Grund wurde hier die spezifische Komplexität jeder Stadt vorgebracht. Seit einigen Jahren wird hingegen versucht mit Hilfe von Lösungsmethoden für multikriterielle Optimierungsprobleme das Problem der multidisziplinären Entwurfsoptimierung in der Stadtplanung zu lösen. Dabei werden alle Aspekte einer Stadt in Betracht gezogen und mit bestimmten Lösungsalgorithmen, wie zum Beispiel evolutionären Algorithmen, eine möglichst optimale Stadtstruktur hinsichtlich der Entwurfsziele erstellt. Die theoretischen Grundlagen für die näherungsweise Lösung dieses praktischen Problems, das die Suche eines optimalen Stadtentwurfs umfasst, sind bereits gegeben. Daher wird versucht diese Grundlagen in Form von computergestützten Entwurfswerkzeugen technisch umzusetzen. Mittlerweile ist eine hohe Anzahl an verfügbaren Softwaresystemen zur Unterstützung des

Städtebauarchitekten beim Entwurfsprozess vorhanden. Dies impliziert die Notwendigkeit eines kombinierten Einsatzes der zur Verfügung stehenden Softwarewerkzeuge und damit die Entwicklung einer allgemeinen Schnittstelle zwischen diesen existierenden Systemen.

8.1.1. Interpretation der Resultate

In dieser Diplomarbeit wurde eine Schnittstelle mit dem Ziel eines kombinierten Einsatzes bereits existierender Technologien bzw. Softwaresysteme im Bereich des computerbasierten Entwerfens und der räumlichen Stadtplanung vorgestellt. Die Schnittstelle wird in Form einer Java-Applikation bereitgestellt. Diese Applikation kombiniert hierbei die Verwendung von raumbezogenen Geodaten aus dem freien Projekt OpenStreetMap sowie dem LGL[1] mit der Möglichkeit einer automatischen Erzeugung und Darstellung eines CAD-Stadtmodells. Zudem wird eine Optimierung dieser kombinierten Daten und die anschließende Bewertung des Optimierungsergebnisses ermöglicht. Um eine Austauschbarkeit der eingesetzten Softwarekomponenten zu ermöglichen, wurde auf eine modulare Implementierung geachtet und gängige Datenaustauschformate verwendet. Die erreichten Resultate ermöglichen eine zielgerichtete Lösung im Hinblick auf die anfänglich formulierte Problemstellung.

8.1.2. Vor- und Nachteile

Ein möglicher Vorteil der Anwendung der Applikation *CADtoGIS Schnittstelle* ist die Vereinfachung des Entwurfsprozesses eines Stadtbereichs in der Stadtplanung. Dabei kann der Städtebauarchitekt durch die visuelle Bewertung eines urbanen Bereichs einfacher und schneller feststellen, ob dieser für den weiteren Entwurfsverlauf geeignet ist oder nicht. Hinzu kommt, dass durch eine integrierte multidisziplinäre Entwurfsoptimierung des gewählten Stadtbereichs, neue Entwürfe in Abhängigkeit von den gegebenen Rahmenbedingungen ohne vorherige Planung erzeugt werden können. Diese Tatsache bietet einen weiteren Vorteil für den Städtebauarchitekten. Die erzeugte Entwurfsmenge liefert eine Vielzahl neuer Entwurfsideen, welche in den gesamten Entwurfsprozess einfließen können und gegebenenfalls den Entwurf verbessern können. Weiterhin unterstützt die Applikation den Städtebauarchitekten beim Entwurf ohne hierbei eine Vielzahl an Entwurfswerkzeugen parallel oder nacheinander starten und bedienen zu müssen.

Als nachteilhaft in der Bedienung der Applikation hat sich die fehlende Funktionalität einer möglichen Bearbeitung eines erzeugten CAD-Stadtmodells innerhalb der Applikation bemerkbar gemacht. Aufgrund des Prototypcharakters der Applikation wurde explizit auf diese Funktionalität verzichtet. Diese kann aber durch den Austausch der Komponente zur Darstellung des CAD-Stadtmodells nachträglich integriert werden.

[1] siehe `https://www.lgl-bw.de/lgl-internet/opencms/de/index.html`

8.1.3. Auswirkungen des Lösungsansatzes

Der entworfene Lösungsansatz zeigt in erster Linie, dass die Machbarkeit einer Schnittstelle im Anwendungsbereich der Stadtplanung gegeben ist. Demnach ist es möglich, raumbezogene Daten aus Geoinformationssystemen, CAD-Anwendungen, Simulationswerkzeuge und Softwareumgebungen zur Optimierung kombiniert mit Hilfe einer Schnittstelle einzusetzen. Bezogen auf das praktische Problem des Städtebauarchitekten, welches die Suche eines möglichst optimalen Entwurfsplans eines Stadtbereichs umfasst, ist der entworfene Lösungsansatz ein erster Schritt hin zu einer universellen Schnittstelle, die nicht nur für fallspezifische sondern auch für allgemeine Problemlösungen im Bereich der Stadtplanung verwendet werden kann.

8.2. Ausblick

Viele der eingesetzten Methoden und Technologien lassen sich beliebig erweitern oder austauschen. Qualitätsstandards sowie Anforderungsbeschreibungen für die Interaktion und Modifikation von CAD-Modellen können verbessert werden. Ebenfalls kann im Bereich der Entwurfsoptimierung die entworfenen Designsprache weiter ausgebaut und mit feiner definierten Erzeugungsregeln erweitert und verbessert werden. Der Bezug von Geodaten, welcher in der ersten Version auf lokal gespeicherten Daten basiert, kann zukünftig vollständig auf den Bezug über die angebotenen Webservices umgestellt werden. Dabei wird auf Webservices der einzelnen Datenquellen für den Bezug von Geodaten zugegriffen. Im Weiteren erhöht dies die Flexibilität und verbessert die Handhabung der Applikation.

Ausgehend von der relativ komplexen Problemstellung dieser Arbeit lässt sich der Lösungsansatz beliebig komplex gestalten. Dies spiegelt sich nicht nur in der Anzahl der eingesetzten Softwaresysteme sondern auch im Entwurf der Designsprache zur Entwurfsoptimierung wieder. Hierbei wären zum Beispiel Anwendungsszenarien im Universitätsbereich wie auch in jedem anderen urbanen Bereich denkbar. Ein Beispiel wäre die optimierte Entwurfsplanung beim Bau der Fakultätsgebäude einer Universität. Demnach könnte die Zielfunktion der Entwurfsoptimierung aus der Anzahl der angebotenen Kurse einer Fakultät mit den meisten Studenten, die den kürzesten Weg zur Mensa haben sollen, abgeleitet werden. Daraufhin würde das urbane Universitätsgelände in Abhängigkeit dieser Zielfunktion entsprechend optimiert, sodass die Fakultäten mit den Kursen, in denen die meisten Studenten eingeschrieben sind, die geringste Entfernung und damit den kürzesten Fußweg zur Mensa haben.

Abschließend kann festgestellt werden, dass zukünftige Entwicklungen in der multidisziplinären Entwurfsoptimierung in vielen städtebaulichen Bereichen Anwendung finden werden und die Notwendigkeit einer einheitlichen Softwarelösung mehr als gegeben ist.

A. Klassendiagramme

In diesem Anhang werden Klassendiagramme des logischen Datenmodells sowie gegebenenfalls weiterer relevanter Komponenten der Anwendungsarchitektur gezeigt.

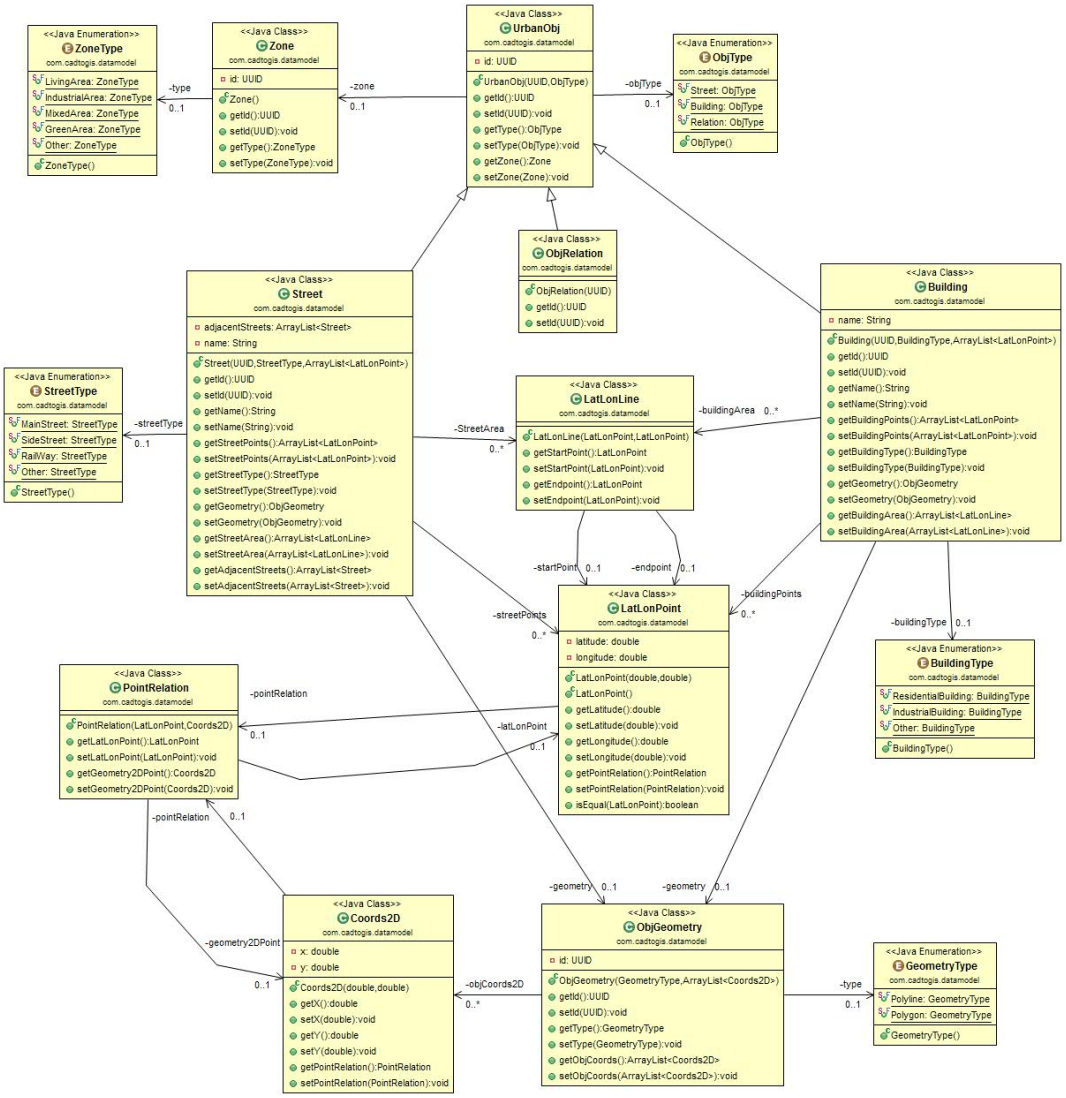

Abbildung A.1.: Klassendiagramm des logischen Datenmodells (1/2)

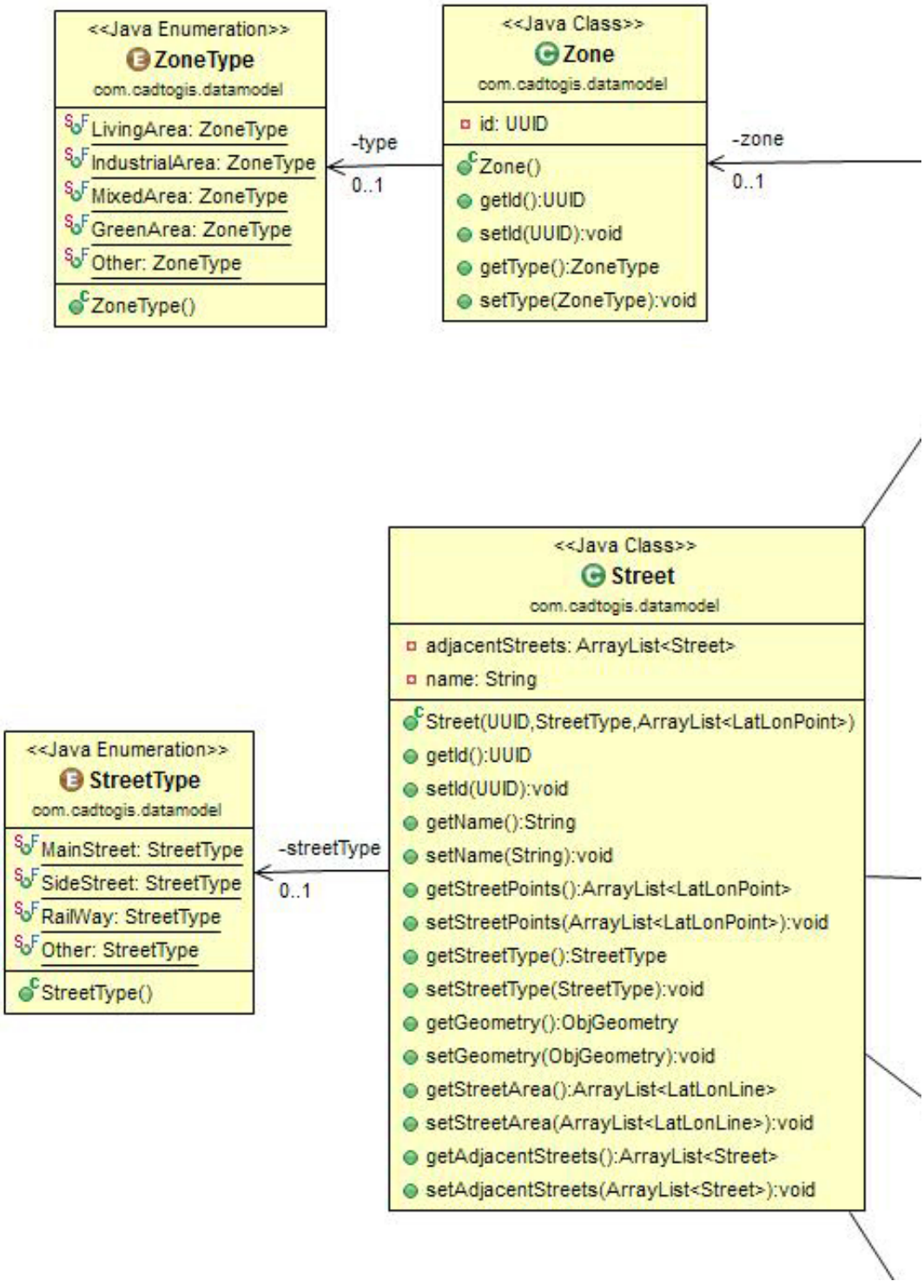

Abbildung A.2.: Klassendiagramm des logischen Datenmodells (1.1/2)

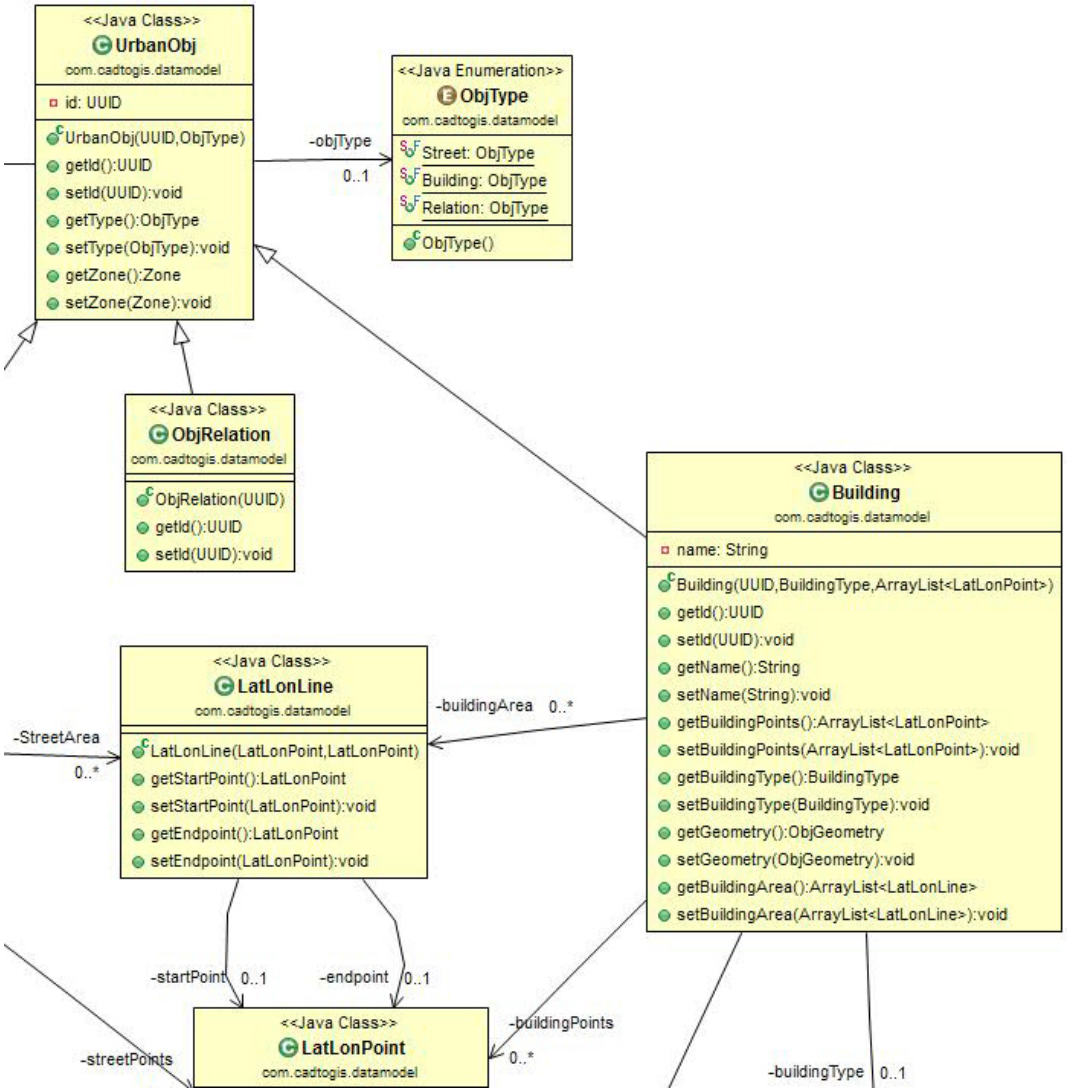

Abbildung A.3.: Klassendiagramm des logischen Datenmodells (1.2/2)

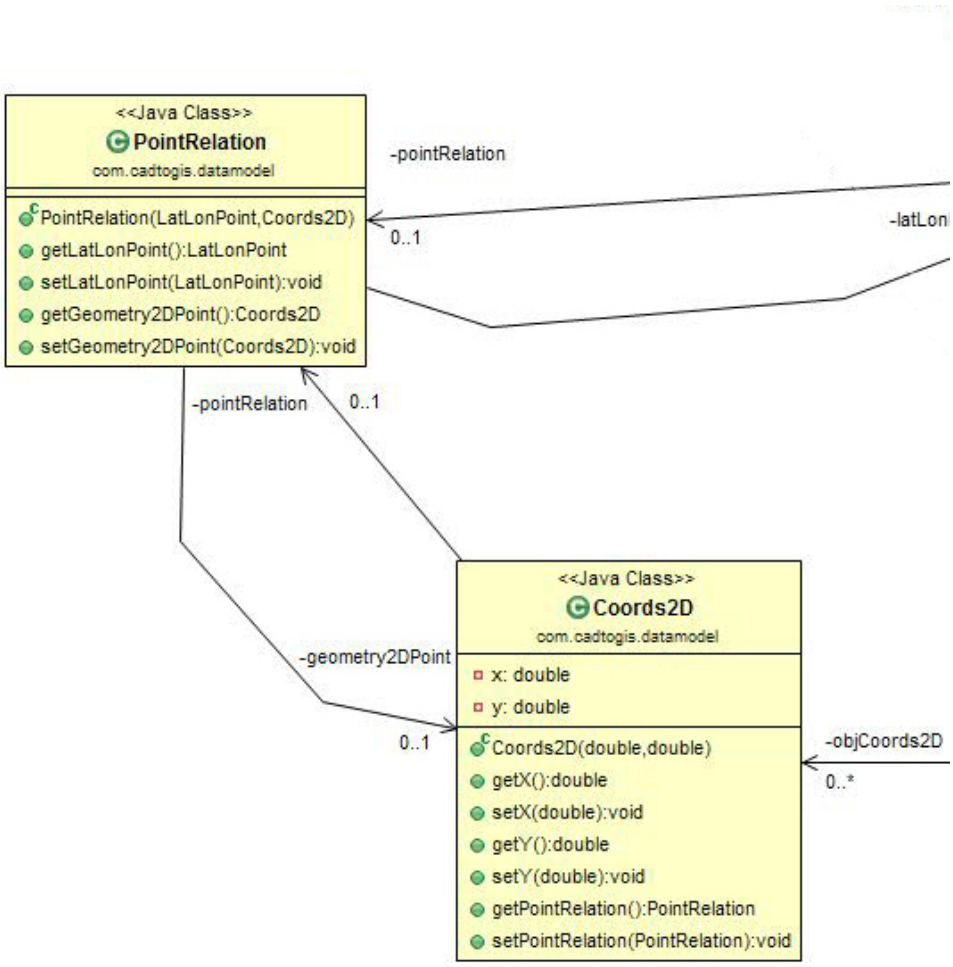

Abbildung A.4.: Klassendiagramm des logischen Datenmodells (1.3/2)

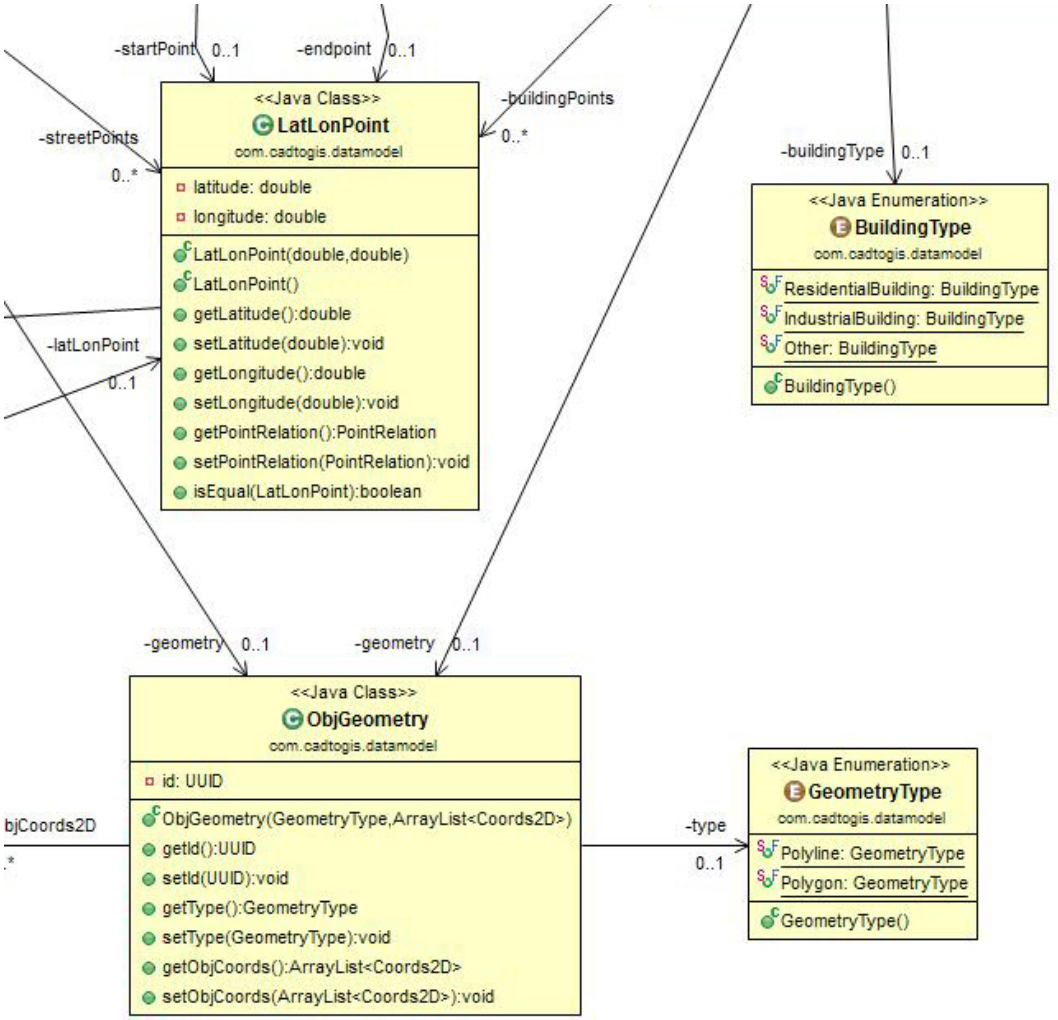

Abbildung A.5.: Klassendiagramm des logischen Datenmodells (1.4/2)

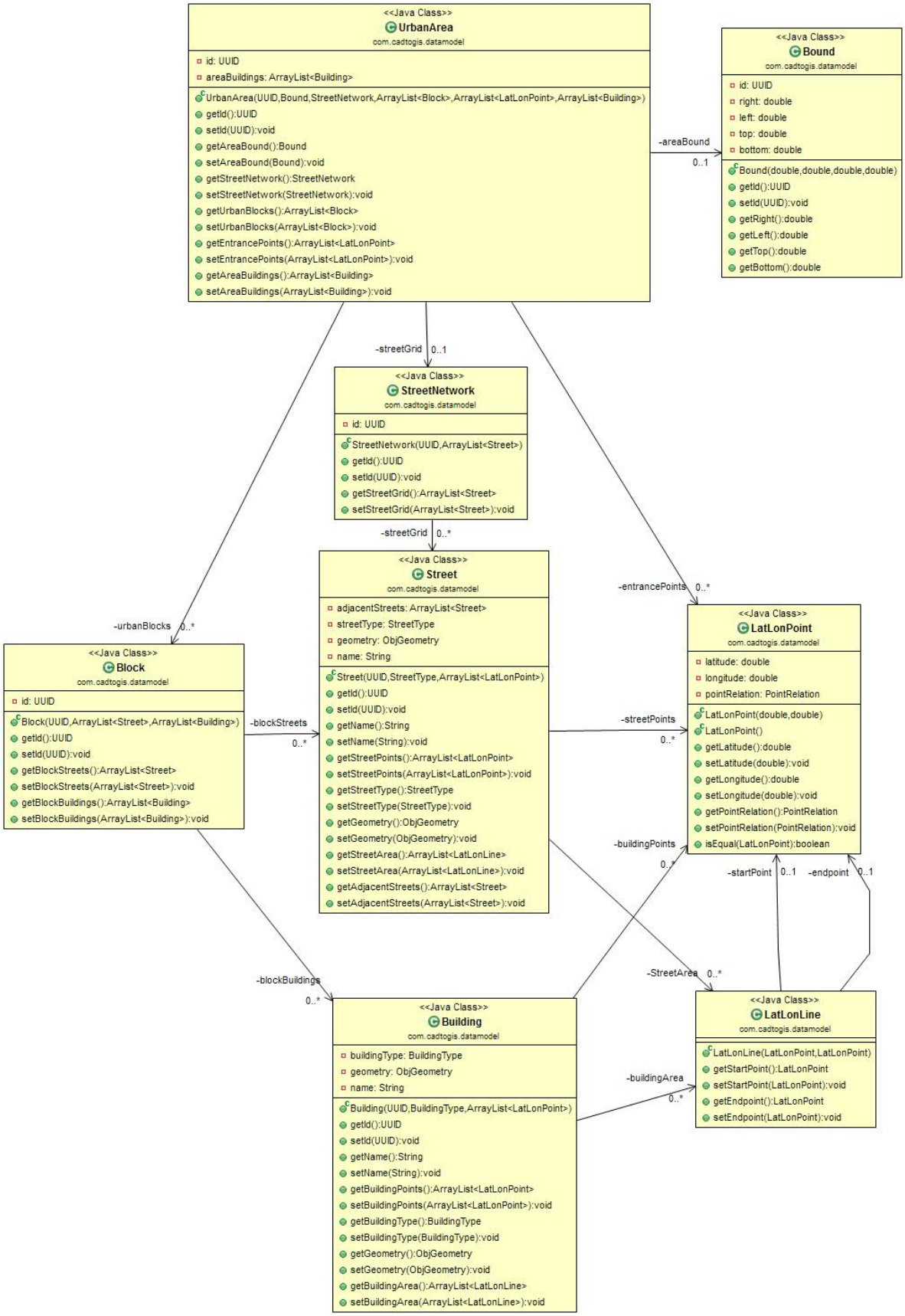

Abbildung A.6.: Klassendiagramm des logischen Datenmodells (2/2)

B. Listings

Dieser Anhang beinhaltet wichtige Quellcode-Ausschnitte bzw. größere Listings, die für das Verständnis des Aufbaus des Projekts wichtig sind.

Listing B.1: Parse OSM-Data with Osmosis

```
/**
 * After generating a new OSM-File with new bounds parse this file into "UrbanObjects".
 *
 * @throws FileNotFoundException
 */
public void parseOSMfile() throws FileNotFoundException{

      //The input file
      File file = new File(path + "lglarea.osm");

      Sink sinkImplementation = new Sink() {
            public void process(EntityContainer entityContainer) {
                  Entity entity = entityContainer.getEntity();
                  if (entity instanceof Node) {

                        //Do something with the node
                        nodes.add(entity);
                  } else if (entity instanceof Way) {

                        //do something with the way
                        ways.add(entity);
                  } else if (entity instanceof Relation) {

                        //Do something with the relation
                  }
            }
            public void release() { }
            public void complete() {
                  LOG.info("File generation complete.");

                  //Set finish flag
                  fileParseComplete = true;
            }

            @Override
            public void initialize(Map<String, Object> arg0) {

            }
      };
```

```
      boolean pbf = false;
      CompressionMethod compression = CompressionMethod.None;

      if (file.getName().endsWith(".pbf")) {
            pbf = true;
      } else if (file.getName().endsWith(".gz")) {
            compression = CompressionMethod.GZip;
      } else if (file.getName().endsWith(".bz2")) {
            compression = CompressionMethod.BZip2;
      }

      RunnableSource reader;

      if (pbf) {
            reader = new crosby.binary.osmosis.OsmosisReader(
                        new FileInputStream(file));
      } else {
            reader = new XmlReader(file, false, compression);
      }

      reader.setSink(sinkImplementation);

      Thread readerThread = new Thread(reader);
      readerThread.start();

      while (readerThread.isAlive()) {
            try {
                  readerThread.join();
            } catch (InterruptedException e) {
                  /* do nothing */
            }
      }
}
```

Listing B.2: Read Esri-Shape-Data with GeoTools

```
/**
 * This method reads a given shape-file with its mandatory files .shx and .dbf.
 * The method returns the geographical points of the geometry of all polygons in the shape-file.
 *
 * @param path
 * @param tf
 * @return ArrayList<Coordinate[]> - List with all polygons of a given shapefile
 * @throws IOException
 * @throws MismatchedDimensionException
 * @throws TransformException
 */
public ArrayList<Coordinate[]> readShapeFile(String path, MathTransform tf) throws IOException,
    MismatchedDimensionException, TransformException {

      ArrayList<Coordinate[]> coordPoly = new ArrayList<Coordinate[]>();
      String shpFilePath = path + ".shp";

      File file = new File(shpFilePath);
      Map<String, URL> map = new HashMap<String, URL>();
      map.put("url", file.toURI().toURL());

      DataStore dataStore = DataStoreFinder.getDataStore(map);
      SimpleFeatureSource featureSource = dataStore.getFeatureSource(dataStore.getTypeNames()[0]);
      SimpleFeatureCollection collection = featureSource.getFeatures();

      SimpleFeatureIterator iterator = collection.features();
      try {
            while(iterator.hasNext()){

                  //Get the geometry and transform it
                  SimpleFeature feature = (SimpleFeature) iterator.next();
                  Geometry geometry = (Geometry) feature.getDefaultGeometry();
                  Geometry geometry2 = JTS.transform(geometry, tf);
                  Coordinate[] coordinatesArr = geometry2.getCoordinates();
                  coordPoly.add(coordinatesArr);
            }
      } catch (Exception problem) {
            problem.printStackTrace();
      } finally {
            iterator.close();
      }
      return coordPoly;
}
```

Listing B.3: Creation of urban objects

```
/**
 * Generates a lists of all buildings, highways and railways in the target area from the OSM-File.
 * This is a decomposition of the urban structure.
 */
public void generateUrbanObjs() {
      for (int i = 0; i <= osmWays.size() - 1; i++) {
            Entity way = osmWays.get(i);

            //Get the tag of way -> should be building, highway, railway
            Collection<Tag> wayTags = way.getTags();
            Object[] tagArr = wayTags.toArray();

            for (int k = 0; k <= tagArr.length - 1; k++) {
                  Tag t = (Tag) tagArr[k];
                  String key = t.getKey();
                  String value = t.getValue();

                  //Get all buildings
                  if (key.equals("building")) {
                        List<WayNode> wayNodes = ((Way) way).getWayNodes();
                        ArrayList<LatLonPoint> points = new ArrayList<LatLonPoint>();

                        for (int l = 0; l <= wayNodes.size() - 1; l++) {
                              WayNode wayNode = wayNodes.get(l);
                              Long nodeId = wayNode.getNodeId();

                              if (getNodePosition(nodeId) != null) {
                                    points.add(getNodePosition(nodeId));
                              }
                        }

                        //Generate a new building object and set initial values
                        Building newBuilding = new Building(UUID.randomUUID(),
                            BuildingType.Other, points);
                        newBuilding.setBuildingArea(genBuildingArea(points));
                        newBuilding.setGeometry(genUrbanObjGeometry(newBuilding, points));
                        newBuilding.setType(ObjType.Building);
                        newBuilding.setZone(zones.get(4));

                        //Set building type explicit
                        if (value.equals("apartments") || value.equals("dormitory") ||
                            value.equals("house") || value.equals("detached")
                                  || value.equals("residential") || value.equals("terrace")) {
                              newBuilding.setBuildingType(BuildingType.ResidentialBuilding);
                        } else if (value.equals("industrial") || value.equals("warehouse")) {
                              newBuilding.setBuildingType(BuildingType.IndustrialBuilding);
                        }

                        //Set the zone in which the building is in
                        newBuilding.setZone(getZone(newBuilding));

                        //Add the new building to global list
                        allBuildings.add(newBuilding);
```

```
                //Get all highways
                } else if (key.equals("highway")) {

                      //Get all tag values of the OSM key highway which are important
                      if (isHighway(value)) {
                            List<WayNode> wayNodes = ((Way) way).getWayNodes();
                            ArrayList<LatLonPoint> points = new ArrayList<LatLonPoint>();

                            for (int l = 0; l <= wayNodes.size() - 1; l++) {
                                  WayNode wayNode = wayNodes.get(l);
                                  Long nodeId = wayNode.getNodeId();

                                  if (getNodePosition(nodeId) != null) {
                                        points.add(getNodePosition(nodeId));
                                        allStreetPoints.add(getNodePosition(nodeId));
                                  }
                            }

                            //Generate a new street object
                            Street newStreet = new Street(UUID.randomUUID(), StreetType.Other,
                                points);
                            newStreet.setType(ObjType.Street);

                            //Set the name of the street
                            newStreet.setName(getStreetName(tagArr));

                            //Set the geometry
                            newStreet.setGeometry(genUrbanObjGeometry(newStreet, points));

                            //Set street type explicit
                            if (value.equals("primary") || value.equals("primary_link")) {
                                  newStreet.setStreetType(StreetType.MainStreet);
                            } else if (value.equals("secondary") ||
                                value.equals("secondary_link")) {
                                  newStreet.setStreetType(StreetType.SideStreet);
                            }

                            //Set the zone to "Other"
                            newStreet.setZone(getZone(newStreet));

                            //Add the new street to global list
                            allStreets.add(newStreet);
                      }
                } else if (key.equals("railway")) {
                      List<WayNode> wayNodes = ((Way) way).getWayNodes();
                      ArrayList<LatLonPoint> points = new ArrayList<LatLonPoint>();

                      for (int l = 0; l <= wayNodes.size() - 1; l++) {
                            WayNode wayNode = wayNodes.get(l);
                            Long nodeId = wayNode.getNodeId();

                            if (getNodePosition(nodeId) != null) {
                                  points.add(getNodePosition(nodeId));
                            }
                      }
```

```
                        //Generate a new highway/railway object
                        Street newStreet = new Street(UUID.randomUUID(), StreetType.RailWay,
                            points);
                        newStreet.setType(ObjType.Street);

                        //Set the geometry
                        newStreet.setGeometry(genUrbanObjGeometry(newStreet, points));

                        //Set the zone to "Other"
                        newStreet.setZone(getZone(newStreet));

                        //Add the new railway to global list
                        allStreets.add(newStreet);
                    }
                }
            }
}
```

C. Graphische Benutzeroberfläche der Applikation CADtoGIS Schnittstelle

In diesem Anhang wird die grafische Umsetzung der Benutzeroberfläche der Applikation *CADtoGIS Schnittstelle* gezeigt.

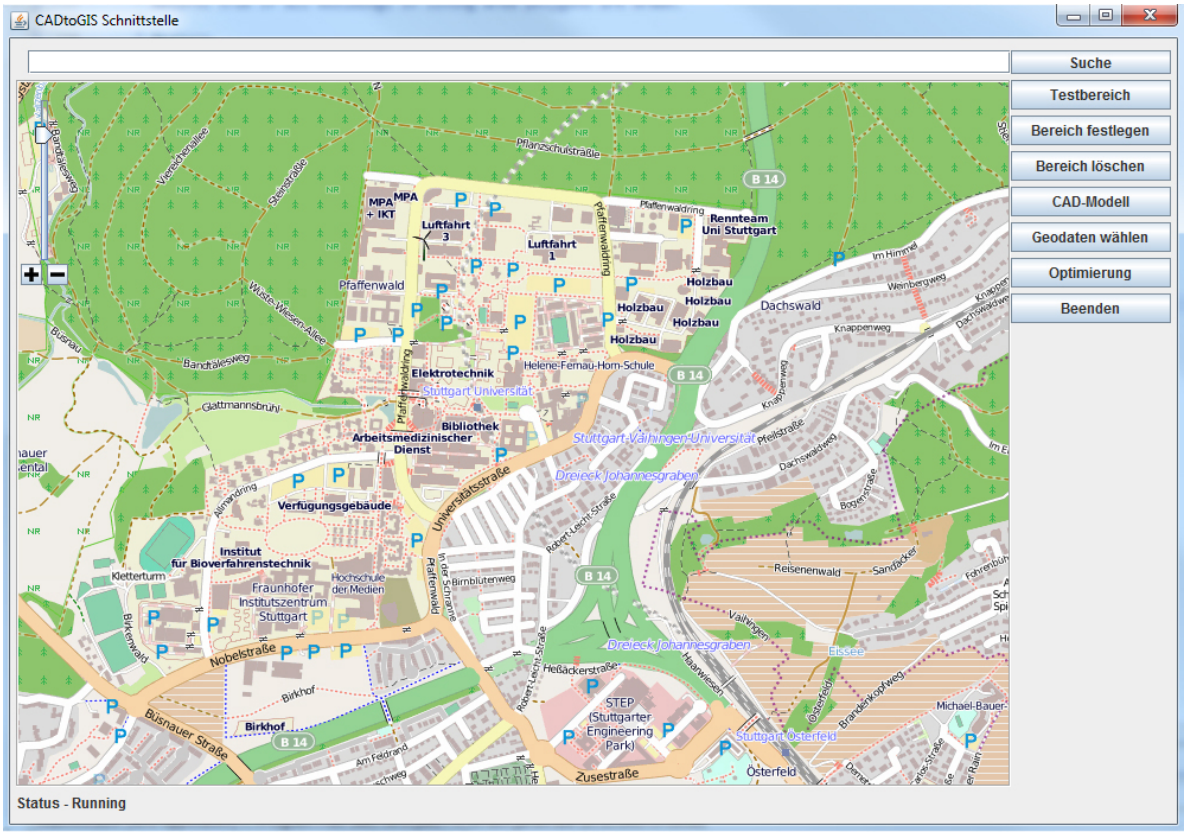

Abbildung C.1.: Das Hauptfenster der CADtoGIS Schnittstelle

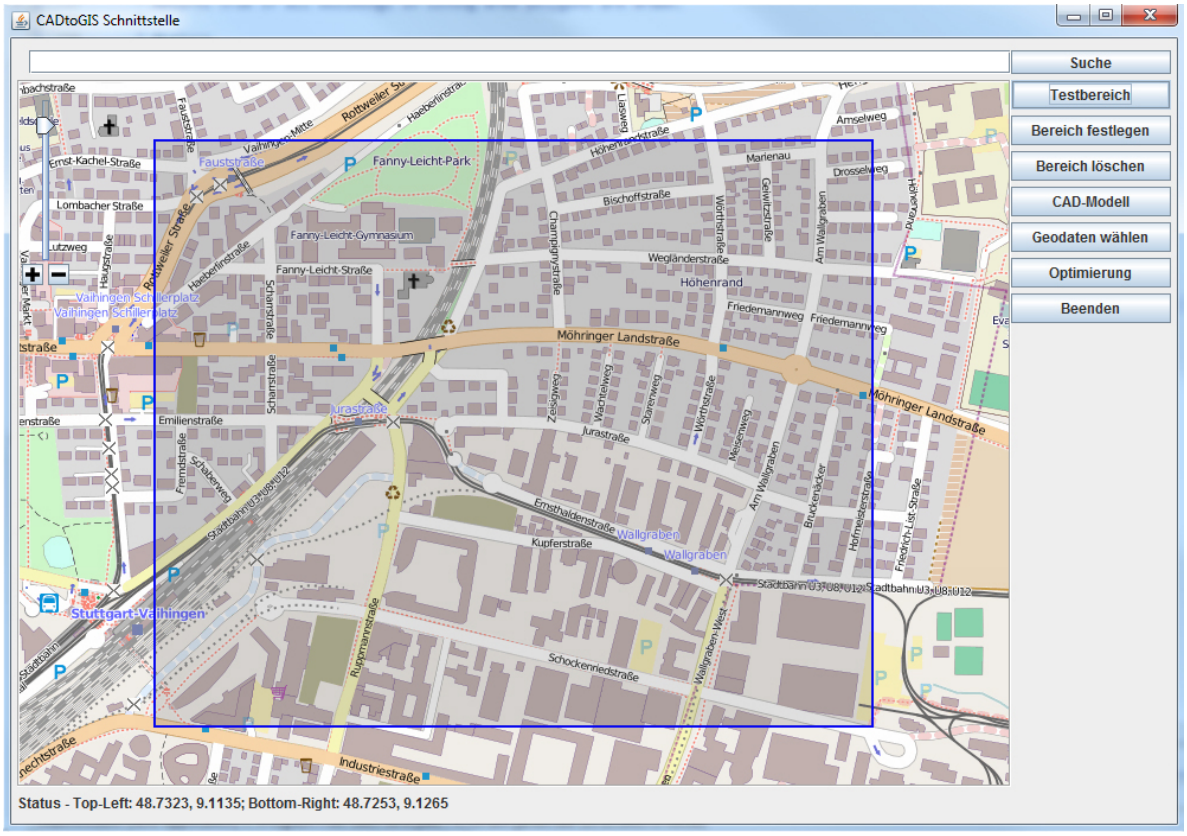

Abbildung C.2.: Vordefinierter Zielbereich aufgrund der verfügbaren LGL-Geodaten

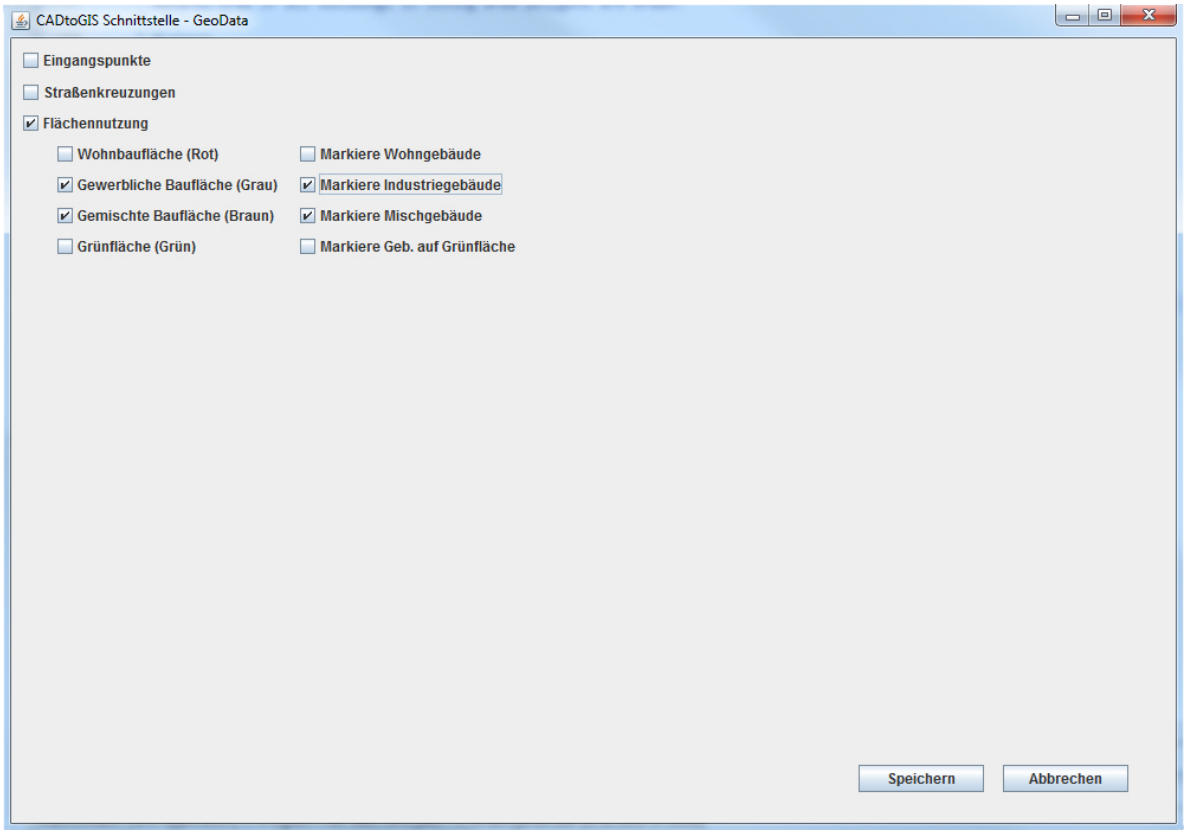

Abbildung C.3.: Auswahl der Geodaten zur Anzeige im Hauptfenster

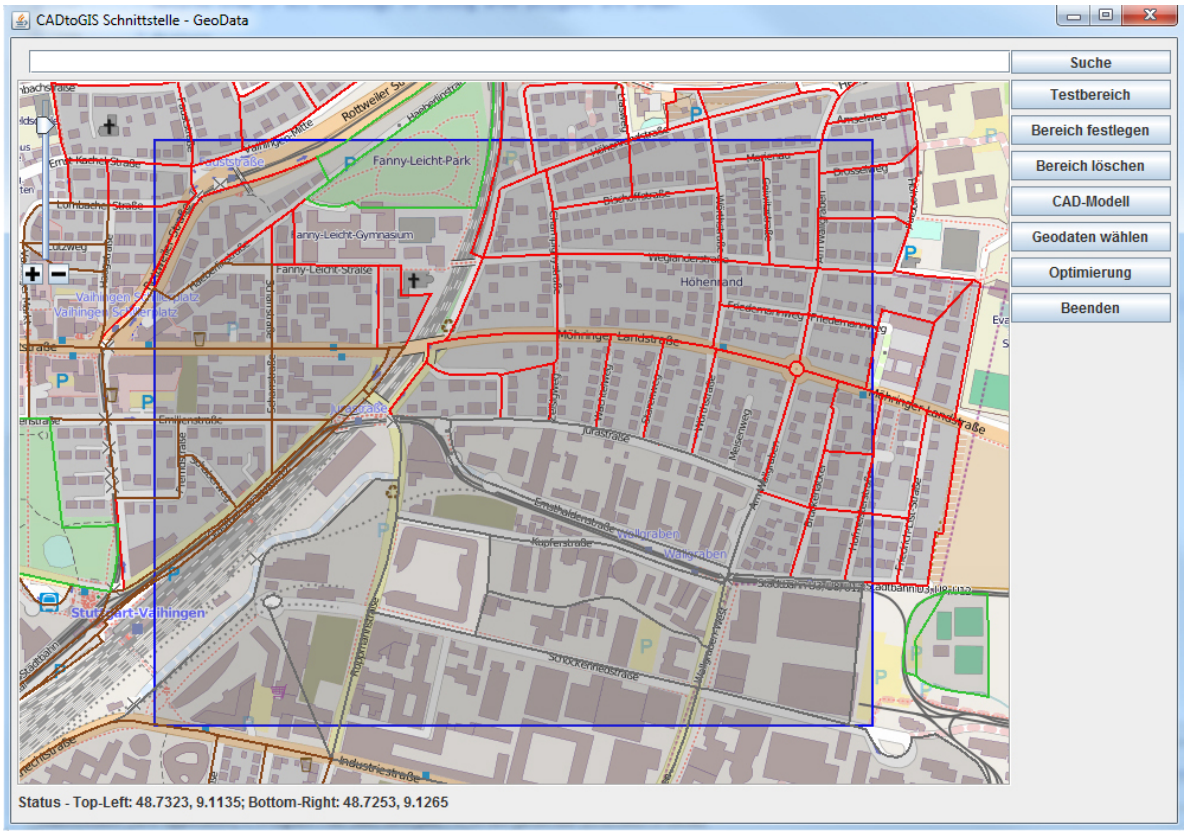

Abbildung C.4.: Anzeige der Geodaten im Hauptfenster - Nutzflächenbereiche

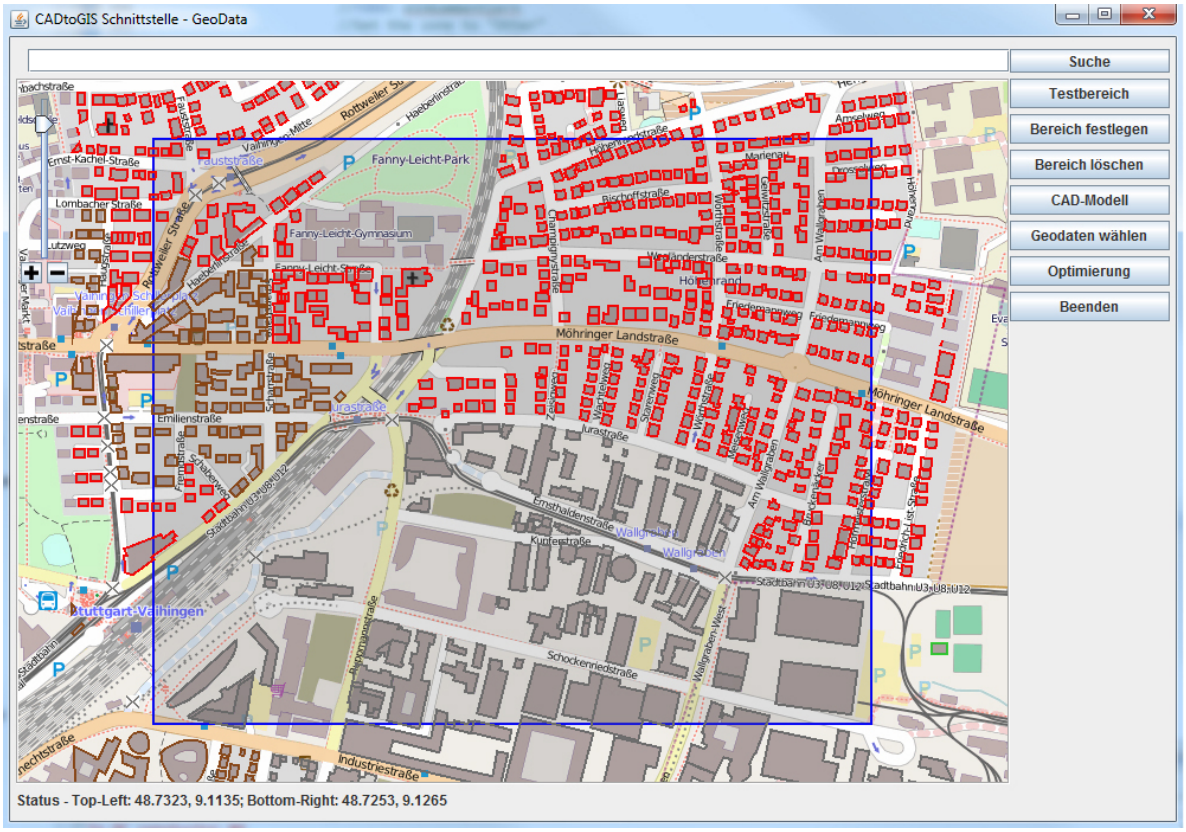

Abbildung C.5.: Anzeige der Geodaten im Hauptfenster - Alle Gebäude nach den Nutzflächenbereichen gefärbt

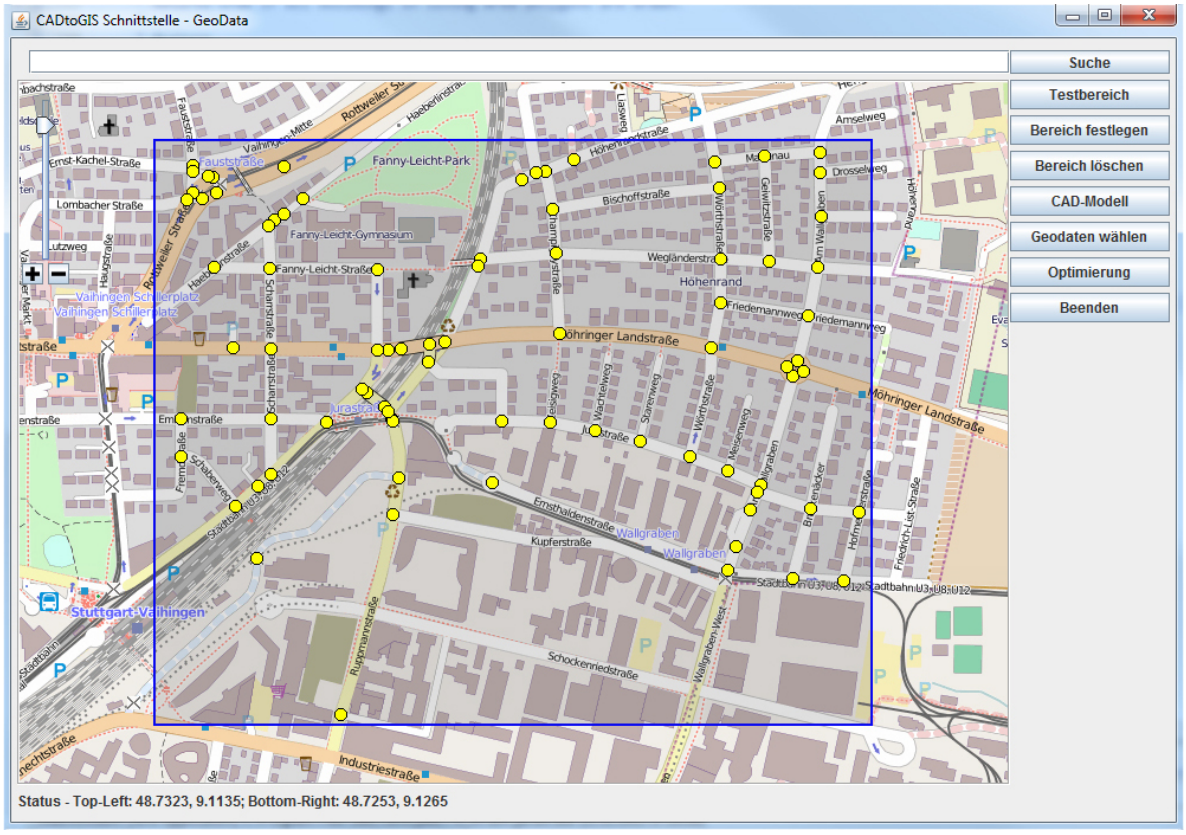

Abbildung C.6.: Anzeige der Geodaten im Hauptfenster - Alle Straßenkreuzungen

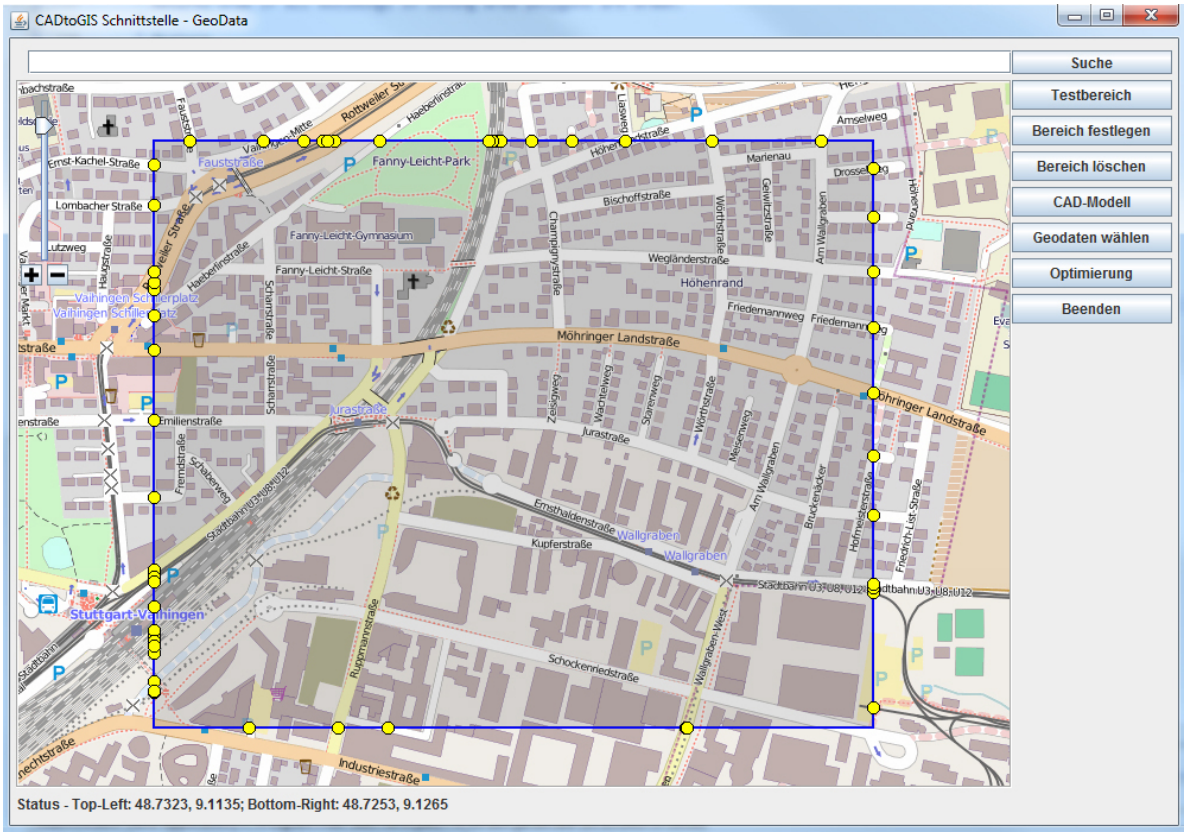

Abbildung C.7.: Anzeige der Geodaten im Hauptfenster - Alle Eintrittspunkte in den Zielbereich

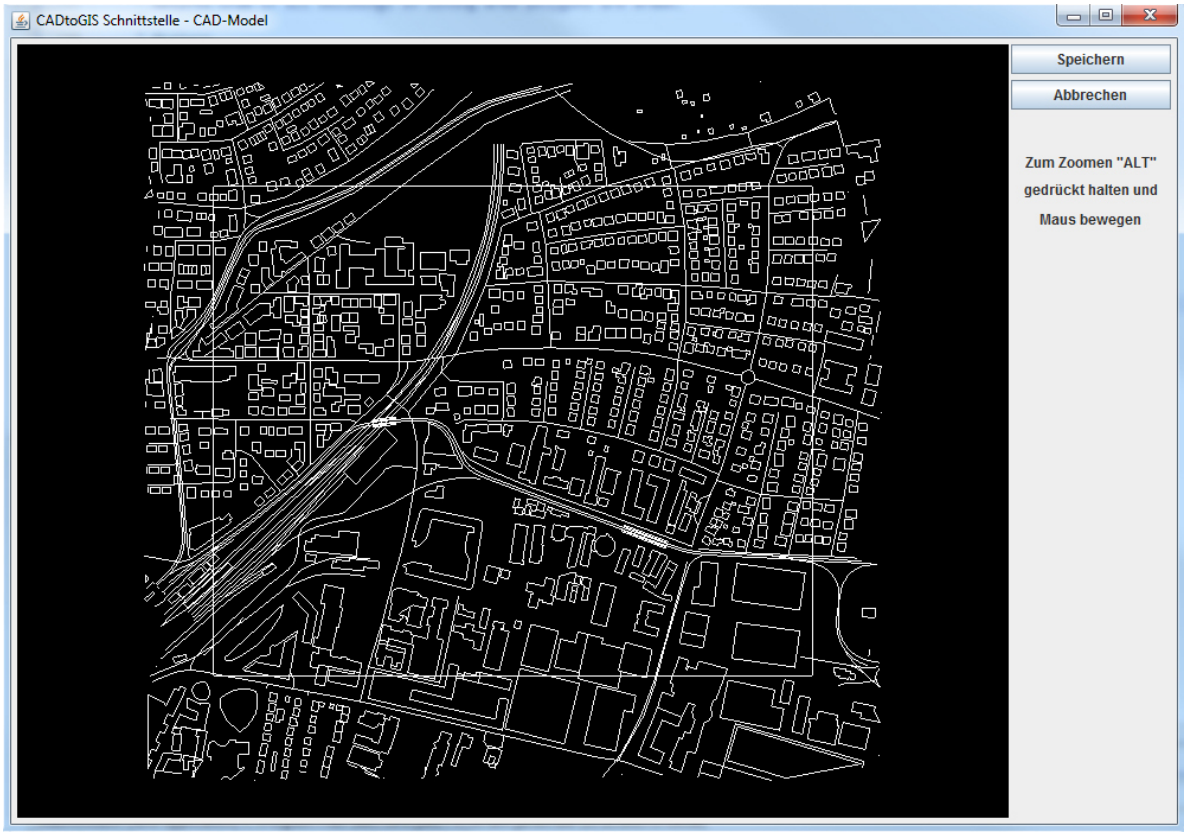

Abbildung C.8.: Anzeige des CAD-Stadtmodells (1/2)

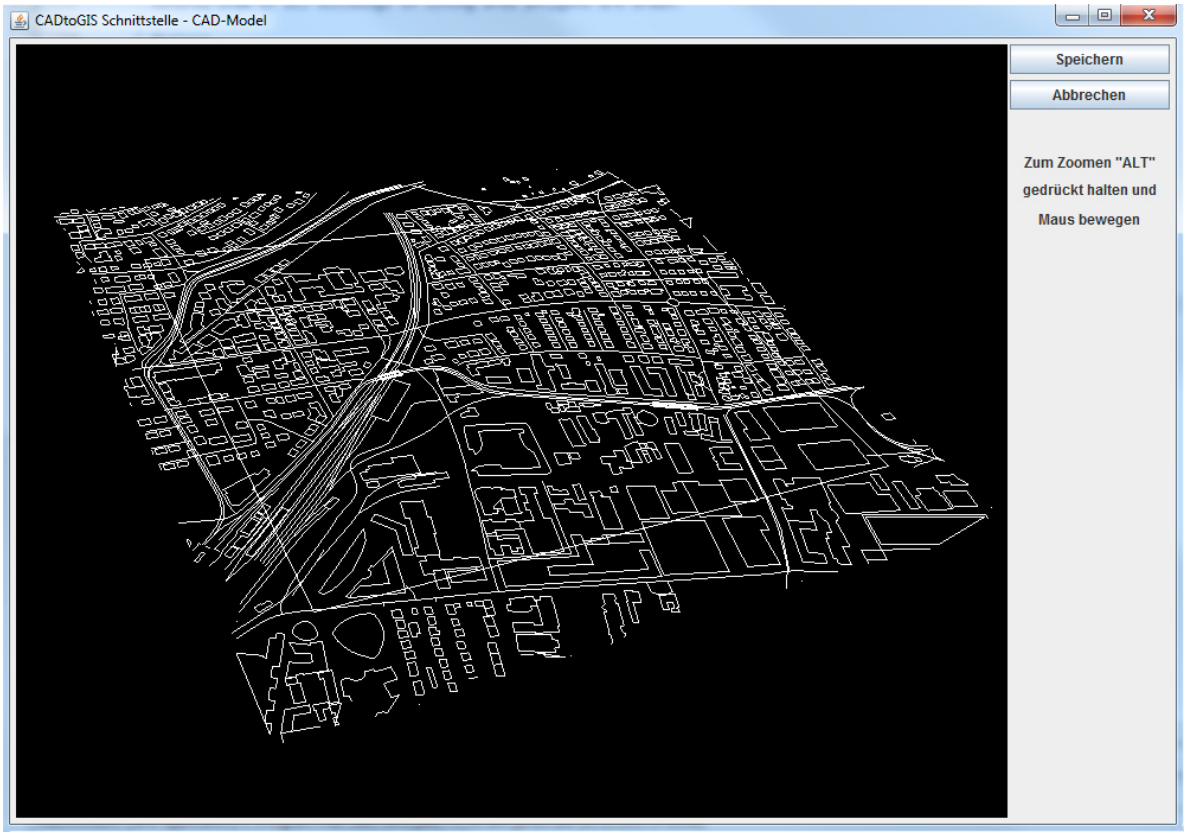

Abbildung C.9.: Anzeige des CAD-Stadtmodells (2/2)

Literaturverzeichnis

[AEHS03] J. C. Aerts, E. Eisinger, G. B. Heuvelink, T. J. Stewart. Using Linear Integer Programming for Multi-Site Land- Use Allocation. *Geographical Analysis*, 35, 2003. (Zitiert auf den Seiten 20, 24 und 25)

[BDS08] J. Beirão, J. Duarte, R. Stouffs. Structuring a Generative Model for Urban Design: Linking GIS to Shape Grammars. *Section 23: City Modelling 2 - eCAADe*, 26:929–938, 2008. (Zitiert auf den Seiten 26, 27, 28, 37, 57 und 58)

[BTBD99] R. J. Balling, J. T. Taber, M. R. Brown, K. Day. Multiobjective Urban Planning using genetic Algorithm. *Journal of Urban Planning and Development*, 1999. (Zitiert auf Seite 24)

[CBH+11] K. Cao, M. Batty, B. Huang, Y. Liu, L. Yu, J. Chen. Spatial multi-objective land use optimization: extensions to the non-dominated sorting genetic algorithm-II. *International Journal of Geographical Information Science*, 2011. (Zitiert auf Seite 24)

[Chu93] E. Chuvieco. Integration of linear programming and GIS for land-use modelling. *International Journal of Geographical Information Systems*, 7:1:71–83, 1993. (Zitiert auf Seite 24)

[CHX09] M. Chandramouli, B. Huang, L. Xu. Spatial Change Optimization: Integrating GA with Visualization for 3D Scenario Generation. *Photogrammetric Engineering & Remote Sensing*, 75:1015–1022, 2009. (Zitiert auf Seite 24)

[Cur96] G. Curdes. *Entwicklung des Städtebaus. Perioden, Leitbilder und Projekte des Städtebaus vom Mittelalter bis zur Gegenwart.* Lehrstuhl und Institut für Städtebau und Landesplanung der Rheinisch-Westfälischen Technischen Hochschule Aachen, 1996. (Zitiert auf Seite 12)

[DB11] J. P. Duarte, J. Beirao. Towards a methodology for flexible urban design: designing with urban patterns and shape grammars. *Environment and Planning B: Planning and Design*, 38:879–902, 2011. (Zitiert auf Seite 26)

[DRS07] J. P. Duarte, J. M. Rocha, G. D. Soares. Unveiling the structure of the Marrakech Medina: A shape grammar and an interpreter for generating urban form. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 21:317–349, 2007. (Zitiert auf den Seiten 26, 57, 58 und 59)

[Ehr00] M. Ehrgott. *Multicriteria Optimization*. Springer Verlag, 2000. (Zitiert auf Seite 21)

[Gey09] P. Geyer. Component-oriented decomposition for multidisciplinary design optimization in building design. *Advanced Engineering Informatics*, 23:12–31, 2009. (Zitiert auf den Seiten 23, 55 und 56)

[GWF11] P. Guarneri, M. Wiecek, G. Fadel. Optimization of non-hierarchically decomposed problems. *Fifth International Conference on Advanced Computational Methods in Engineering (ACOMEN)*, 14-17, 2011. (Zitiert auf Seite 23)

[JMT02] D. Jones, S. Mittazavi, M. Tamiz. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, 137:1–9, 2002. (Zitiert auf Seite 22)

[JS04] F. Jarre, J. Stoer. *Optimierung*. Springer, 2004. (Zitiert auf Seite 20)

[MCF93] A. Montanari, G. Curdes, L. Forsyth. *Urban Landscape Dynamics - A Multi-Level Innovation Process*. Ashgate Publishing, Limited, Avebury UK, illustrated Auflage, 1993. (Zitiert auf Seite 12)

[MMH12] Z. Masoomi, M. S. Mesgari, M. Hamrah. Allocation of urban land uses by Multi-Objective Particle Swarm Optimization algorithm. *International Journal of Geographical Information Science*, S. 1–25, 2012. (Zitiert auf den Seiten 22 und 25)

[MP01] P. Müller, Y. Parish. Procedural Modeling of Cities. *ACM SIGGRAPH*, S. 12–17, 2001. (Zitiert auf Seite 28)

[MP06] R. Mach, P. Petschek. *Visualisierung digitaler Gelände- und Landschaftsdaten*. Springer Verlag, 2006. (Zitiert auf Seite 58)

[Pea90] F. Pearson. *Map Projections: Theory and Applications*. CRC Press Inc., 1990. (Zitiert auf Seite 68)

[PRS+94] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey. *Human-Computer Interaction*. Addison-Wesley-Verlag, 1994. (Zitiert auf Seite 83)

[RN09] S. Russell, P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009. (Zitiert auf Seite 22)

[SJFY11] M. Shifa, H. Jianhua, L. Feng, Y. Yan. Land-Use Spatial Optimization Based on PSO Algorithm. *Geo-spatial Information Science*, 14:54–61, 2011. (Zitiert auf Seite 25)

[WMP03] D. P. Ward, A. T. Murray, S. R. Phinn. Integrating spatial optimization and cellular automata for evaluating urban change. *The Annals of Regional Science*, 37:131–148, 2003. (Zitiert auf Seite 25)

[Wu98] F. Wu. SimLand: a prototype to simulate land conversion through the integrated GIS and CA with AHP-derived transition rules. *Geographical Information Science*, 12:63–82, 1998. (Zitiert auf Seite 25)

Alle URLs wurden zuletzt am 15. 12. 2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Neuhausen a. d. F., den 15. Dezember 2013

Oleg Martin