Hochschule Neubrandenburg
University of Applied Sciences

Studiengang Geoinformatik
6. Semester

Bachelorarbeit

Haltung und Fortführung von 3D Stadtmodellen
mit der 3D City Database und dem CityServer3D

Zum Erlangen des akademischen Grades
"Bachelor of Engineering"(B.Eng.)

URN: urn:nbn:de:gbv:519-thesis 2012-0634-5

eingereicht von: Andreas Krietemeyer

eingereicht am: 27.08.2012

Betreuer: Herr Prof. Dr. Ing. Andreas Wehrenpfennig
Herr Dipl. Ing. Jan Tischer

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

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

Ort, Datum	*Andreas Krietemeyer*

Danksagung

Zu Beginn dieser Arbeit möchte ich mich herzlich bei meinen Betreuern Prof. Dr.-Ing. Andreas Wehrenpfennig und Dipl.-Ing. Jan Tischer für ihre professionelle Unterstützung sowie deren konstruktive Kritiken und Anregungen bedanken.

Ein besonderer Dank gilt meiner Freundin Anika für die Motivation bei der Erstellung und dem Korrekturlesen dieser Arbeit. Danken möchte ich in diesem Zuge auch meinen Eltern, die mir dieses Studium ermöglicht haben.

Abschließend gilt mein Dank all denjenigen Menschen, die mich neben und während meines Studiums begleitet haben.

Kurzfassung

3D Stadtmodelle gewinnen in der heutigen Welt immer mehr an Bedeutung. Durch die steten Veränderungen von Stadtbildern müssen Geodaten erneuert werden. Aufgrund dessen ist eine effektive Speicherung und Fortführung der Datenbestände notwendig.
Die vorliegende Arbeit untersucht einen derartigen Arbeitsprozess. Die Speicherung findet häufig in ungünstigen Dateisystemen statt. In dieser Arbeit werden daher die Datenbanklösungen CityServer3D und 3D City Database untersucht. Thematisiert werden die Datenhaltung und -fortführung sowie ein Vergleich der Datenbanken.
Es werden Tests mit verschiedenen Datensätzen durchgeführt und die Performance sowie Features verglichen. Im CityServer3D wird mithilfe der integrierten regelbasierten Sprache ein simpler Fortführungsprozess umgesetzt. Für die 3D City Database wird für diesen Prozess ein eigenes Tool entwickelt.

Abstract

Nowadays 3D city models are gaining more and more importance. Because of the constant changes of townscapes, these geo data need to get renewed. As a result of this, the effective storage and continuation of stored data is required.
Within this paper such a business process will be investigated and realized. The storage often occurs at inappropriate file based systems. Due to this, the database solutions CityServer3D and 3D City Database will be analyzed. The themes are storage, continuation and a comparison of the databases.
Different dataset tests will be done and performance and features will be compared. Using the integrated rule based language, a simple continuation process will be implemented in the CityServer3D. An own tool will be developed for this process in the 3D City Database.

Inhaltsverzeichnis

1. Einleitung

3D Stadtmodelle gewinnen seit Jahren immer mehr an Bedeutung. Auftraggeber wie Stadtverwaltungen und Firmen benötigen sie beispielsweise für Stadtplanungen, Solarpotentialanalysen, Wirtschaftsförderungen und Immobilienvermarktung.
Um 3D Stadtmodelle mit Geoinformationssystemen (GIS) nutzen zu können, müssen Daten erfasst, konvertiert und dauerhaft gespeichert werden. Diese Daten haben einen räumlichen Bezug und werden als Geodaten bezeichnet. Die Datenkonvertierung kann u.a. durch den 2002 von der Special Interest Group 3D (SIG 3D) entwickelten Standard City Geography Markup Language (CityGML) realisiert werden [GKCN08]. Um die entstandenen Geodaten dauerhaft zu speichern, bieten sich spezielle Datenbankmodelle wie der CityServer3D und die 3D City Database an.
Durch den steten Wandel des Stadtbildes und die sich verändernden Anforderungen an 3D Stadtmodelle müssen die gespeicherten Daten erneuert werden. Die Erhöhung des Detaillierungsgrades bzw. die Aktualisierung der Geodaten wird als Datenfortführung bezeichnet.
Das Thema dieser Arbeit wurde von der Firma GTA Geoinformatik GmbH gestellt. Die Bachelorarbeit gibt einen Überblick zu 3D Stadtmodellen und befasst sich mit deren Speicherung in den Datenbanklösungen CityServer3D und 3D City Database. Um u.A. eine anwendungsspezifische Beratung bei der Wahl der Datenbanklösung geben zu können, erfolgt eine Untersuchung des CityServer3D und der 3D City Database hinsichtlich der Haltung von 3D Stadtmodellen sowie ein Funktions- und Performancevergleich dieser Datenbanklösungen. Für beide Systeme wird zudem eine Möglichkeit zur Datenfortführung beschrieben, um den Datenbestand aktuell zu halten.

2. Analyse: vom Auftrag zur Datenhaltung / Fortführung

In der Einleitung wird der Ablauf zur Herstellung eines virtuellen 3D Stadtmodells kurz geschildert. Je nach Auftrag wird zunächst die Erfassung von Rohdaten vorgenommen. Bevor Daten konvertiert und gespeichert werden können, werden Rohdaten in Nutzdaten umgewandelt. Ein möglicher Anwendungsfall dieser Prozesse ist in Abbildung 1 (Arbeitsprozess 1) dargestellt:

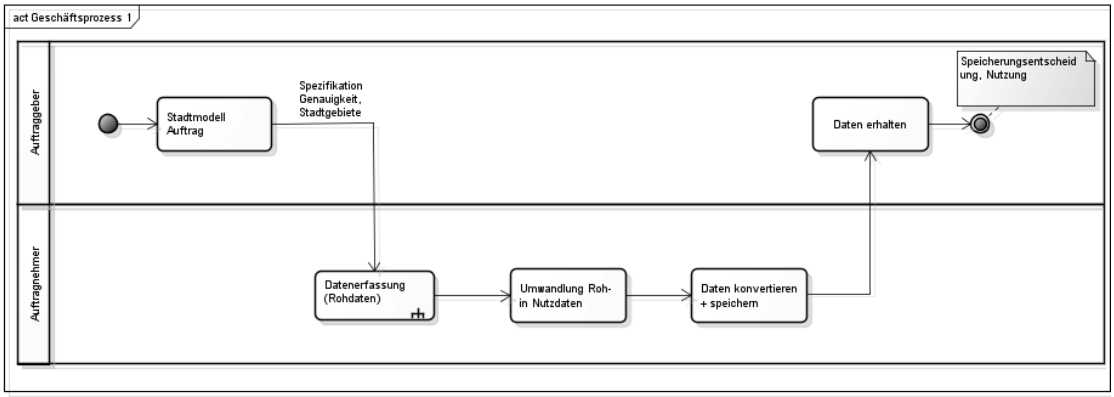


Abbildung 1: Datenproduktion (Arbeitsprozess 1)

In dem Diagramm sind zwei Akteure dargestellt: der Auftraggeber und der Auftragnehmer. In Einzelfällen kann der Auftraggeber auch die Aufgaben des Auftragnehmers übernehmen. Laut dem Diagramm fordert der Auftraggeber beim Auftragnehmer ein 3D Stadtmodell an und spezifiziert das aufzunehmende Gebiet und das gewünschte Level of Detail (LoD) des Modells. Der Auftragnehmer führt daraufhin die Erfassung des Stadtgebietes durch (Kapitel 3.1). Mithilfe spezieller Auswertetechnik (Stereobildauswertung) und Software (z.B. Tridicon) werden aus den aufgenommenen Rohdaten Gebäudemodelle entwickelt. Anschließend werden die Daten in ein standardisiertes Format (Kapitel 3.5) konvertiert, gespeichert und dem Auftraggeber ausgeliefert.
Der Arbeitsprozess 2 (Abb. 2) zeigt den Ablauf, nachdem der Auftraggeber die Daten des 3D Stadtmodells z.B. in Form von CityGML Dateien erhalten hat.

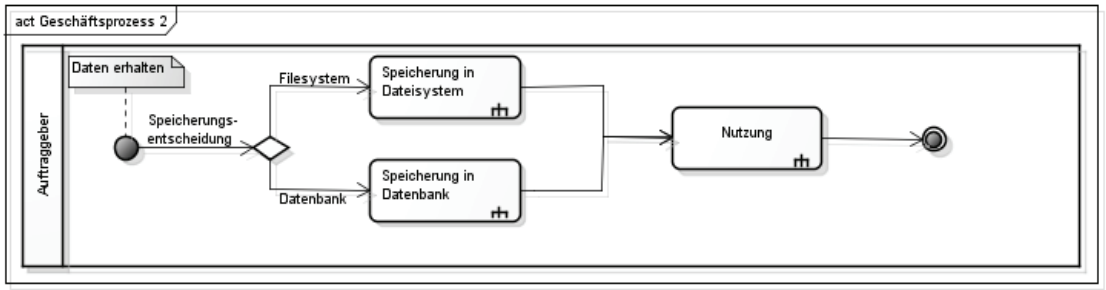


Abbildung 2: Datenhaltungsoptionen (Arbeitsprozess 2)

Es wird deutlich, dass der Auftraggeber zwei grundlegende Möglichkeiten zur Speicherung der Geodaten hat: die Speicherung in ein gewöhnliches Filesystem oder in eine Datenbank. Die Entscheidung wird heutzutage auf ein Datenbanksystem fallen, da die Vorteile eines solchen Systems sehr vielseitig sind und Filesysteme die Anforderungen moderner Anwendungen nicht erfüllen. Die Vorteile eines Datenbanksystems werden in Kapitel 4.1 beschrieben.Nach der Speicherung können die Daten von GIS genutzt werden. Typische Anwendungen sind in Kapitel 3.3 beschrieben.
Diese Arbeit beschäftigt sich mit der Speicherung von 3D Stadtmodellen in den Datenbanklösungen CityServer3D und 3D City Database. Arbeitsprozess 3 (Abb. 3) zeigt die erste Problemstellung dieser Arbeit:

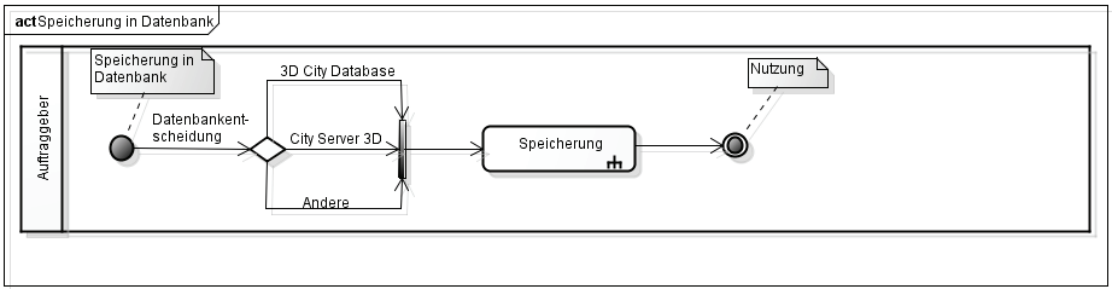


Abbildung 3: Speicherung in Datenbank (Arbeitsprozess 3)

Laut dem Diagramm kann der Akteur zur Speicherung von 3D Stadtmodellen zwischen den Datenbanklösungen 3D City Database, CityServer3D oder einer anderen Datenbanklösung entscheiden. In dieser Arbeit wird auf die Speicherung von Geodaten in der 3D City Database und dem CityServer3D eingegangen. Abbildung 4 zeigt den allgemeinen Aufbau eines solchen Datenbanksystems:

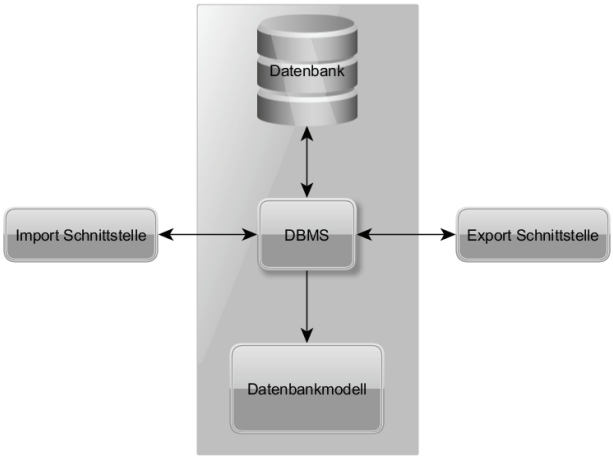


Abbildung 4: Datenbanküberblick

Die speziellen Datenbankmodelle der beiden Datenbanklösungen übernehmen die Modellierung der in der Datenbank gespeicherten Daten. Das Datenbankmanagementsystem (DBMS), sowie die konkrete Datenbank wird von bekannten Datenbanken wie Oracle, PostgreSQL, MongoDB oder MySQL realisiert. Die in der Abbildung 4 gezeigten Im- und Export Schnittstellen werden von eigenständigen Programmen realisiert. Anforderungen an die Schnittstellen sind, dass weder geometrische, topologische oder semantische Eigenschaften verloren gehen dürfen sowie verschiedene Datenformate unterstützt werden sollen. Ein Test und ein Vergleich dieser Schnittstellen werden in dieser Arbeit ebenso durchgeführt (Kapitel 5).

In den oben genannten Arbeitsprozessen wird die Alterung der Daten jedoch nicht berücksichtigt. Im Laufe der Zeit können die gespeicherten Daten veralten oder müssen veredelt werden. Arbeitsprozess 4 (Abb. 5) zeigt einen solchen Fortführungsprozess:

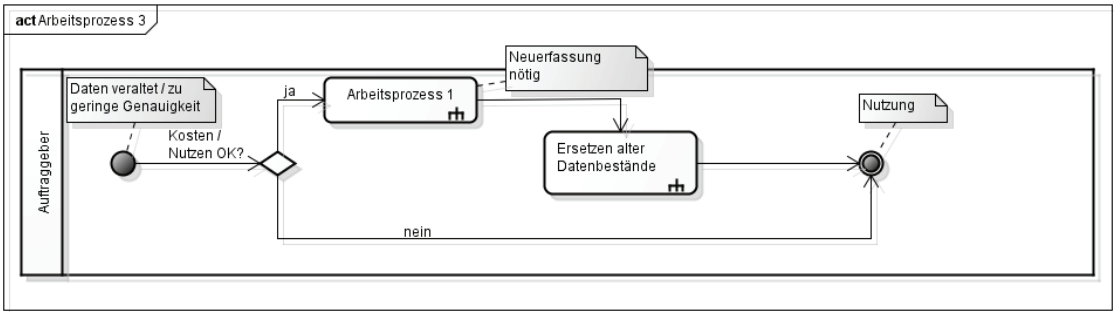


Abbildung 5: Fortführung (Arbeitsprozess 4)

Bei einer Veredelung müssen die Daten nicht unbedingt neu erfasst werden. Dies ist beispielsweise der Fall, wenn ein Stadtmodell bereits in mehreren Genauigkeitsstufen vorliegt. In dem Szenario aus Abbildung 5 wird allerdings von einer Neuerfassung der Daten ausgegangen. Dies entspricht dem Arbeitsprozess 1. Nach Auslieferung der neuen Daten müssen diese in der Datenbanklösung ersetzt werden, ehe sie wiederum von GIS genutzt werden können.
Die korrekte Identifizierung der Objekte geschieht über identische IDs, welche meist aus der Automatisierten Liegenschaftskarte (ALK) übernommen werden. Diese müssen demnach sowohl im neuen sowie im alten Datensatz gleich sein. Eine weitere Möglichkeit ist ein geometrischer Abgleich, welcher die Geometrie von Objekten miteinander vergleicht. Diese Alternative ist jedoch sehr rechenintensiv.
In den Fortführungsprozessen wird die Objektidentifizierung demnach über IDs umgesetzt. Ein solcher Fortführungsprozess wird in den beiden geschilderten Datenbanklösungen nicht direkt unterstützt. Das Hauptaugenmerk in dieser Arbeit liegt daher in einer Schnittstelle zur 3D City Database zur Speicherung, Löschung und Fortführung von 3D Gebäudemodellen, sowie einer simplen Fortführungsmöglichkeit von Objekten im CityServer3D.

3. 3D Stadtmodelle

Die vorliegende Arbeit beschäftigt sich mit der Speicherung von 3D Stadtmodellen in den Datenbanklösungen CityServer3D und 3D City Database. In diesem Abschnitt werden zunächst die theoretischen Grundlagen von 3D Stadtmodellen erläutert. Dabei wird ein Überblick zur Entstehung dieser gegeben, sowie die Anwendungsbereiche und die Anforderungen beschrieben.
Ein digitales 3D Stadtmodell ist ein visualisiertes, vereinfachtes, idealisiertes, maßstäbliches und georeferenziertes Abbild von Gebäuden, Straßen, Vegetation sowie sämtlicher damit verbundener Objekte der Erdoberfläche. Den Objekten werden Informationen für eine räumliche Rekonstruktion zugeordnet und möglichst realitätsnah abgebildet. [Lor96] Dieser Eindruck wird mithilfe von originalgetreuen Texturen vermittelt. Für die geometrische Rekonstruktion muss das Modell mit Geodaten angereichert werden. Ein 3D Stadtmodell ist zur Analyse, Simulation oder Planung eines bestimmten Gebietes geeignet [SK].

3.1. Erfassungsmethoden von Quelldaten

Datenerfassung bezeichnet das Sammeln und Aufzeichnen von Daten für die Datenverarbeitung. Wie bereits erwähnt, werden für 3D Stadtmodelle Geodaten benötigt. Das Hauptmerkmal ist die realitätsnahe und georeferenzierte Darstellung von Gebäuden. Geodaten werden durch verschiedene Erfassungsmethoden erzeugt. Insbesondere handelt es sich dabei um ausgewertete Stereo- (Luft-)bilder, Laserscandaten oder die Nutzung bereits vorhandener Daten (z.B. Katasterdaten). Die durch die Datenerfassung entstandenen Daten werden in dieser Arbeit als Rohdaten bezeichnet.

3.2. Modellierungsarten

Die geometrische 3D Stadtmodellierung unterscheidet zwischen dem Volumenmodell und dem Boundary Representation (B-Rep) Modell.
Ein Volumen- oder Constructive Solid Geometry (CSG) Modell besteht aus beliebig vielen miteinander kombinierbaren 3D Körpern (z.B. Quader, Kugel, Zylinder). Kombiniert werden die Objekte über grundlegende boolsche Operatoren wie Schnitt, Vereinigung und Differenz. Durch diese Methode können komplexe Gebilde in Form und Gestalt abgebildet werden. Allerdings ist die Konstruktion nicht eindeutig, da es immer mehrere Möglichkeiten gibt, ein Objekt auf diese Weise zu beschreiben. Durch die Kombination mehrerer Operationen entsteht ein hierarchisch strukturierter CSG - Baum, bei dem alle Teile beschreibbar und demnach Informationen über das Volumen der Körper ableitbar sind. Die digitale Rekonstruktion von Objekten mit dieser Methode ist dementsprechend einfach und wird vorrangig in Konstruktionssoftware eingesetzt [AG 04].
Das weitaus bekanntere Modell zur Darstellung von 3D Stadtmodellen basiert auf Begrenzungsflächen. Es handelt sich dabei um eine Randflächenmodellierung (B-Rep), in der Flächen, die ein Objekt umschließen (z.B. Polygone), zu einem zusammengesetzten Objekt aggregiert werden. Bei dieser Methode werden Punkt- und Kanteninformatio-

nen sowie Flächeneigenschaften definiert. Einzelne Flächen können separat angesprochen werden, wodurch eine Texturierung der Flächen erfolgen und ein realitätsnaher Eindruck einfach erzeugt werden kann. Durch diese schnelle Visualisierungs- und Ansprechbarkeit von Objektteilen kommen in den meisten 3D Anwendungen fast ausschließlich B-Rep Modelle zum Einsatz [AG 04]. Die Erzeugung erfolgt (halb-)automatisch oder manuell über spezielle Software.

3.3. Anwendungsbereiche

Die Erstellung eines 3D Stadtmodells ist aufgrund der Kostenfaktoren nur für bestimmte Branchen interessant. Im folgenden Abschnitt werden die Zielgruppen mit der zugehörigen Verwendung und Detaillierung vorgestellt und erläutert.

Stadtplanung
In der Stadtplanung ist die visuelle Präsentation von Bestandssituationen oder Planungsszenarien die häufigste Verwendungsart. Die Änderungsvorschläge werden verständlich präsentiert und können den Entscheidungsprozess beschleunigen. Dadurch können auch Bürger an der städtebaulichen Entwicklung mitwirken. 3D Stadtmodelle können in diesem Zuge ebenso zur Bewertung von Immobilienstandorten oder zur Fördermittelvergabe genutzt werden. Aufgrund der Kosten wird für Großraumprojekte meist auf einen geringen Detaillierungsgrad (LoD1-2) zurückgegriffen, während eine höhere Detaillierung nur für Einzelgebäude in Betracht kommt [ABH03].

Tourismus
Um das Interesse an einem Reiseziel zu steigern spielt in der Tourismusbranche ein hoher Wiedererkennungswert, eine detaillierte Modellierung, sowie die Erfassung von Hintergrundinformationen eine große Rolle. Die 3D Stadtmodelle sollten dabei in einer möglichst hohen Detailstufe vorliegen (mind. LoD2).

Navigation
Auch in der Fahrzeug- sowie Fußgängernavigation können 3D Stadtmodelle eingesetzt werden. Mithilfe von realitätsnahen 3D Darstellungen der Umgebung wird die visuelle Orientierung des Anwenders unterstützt. Entwickelt wurde ein solches „Navigationssystem der nächsten Generation“in dem Pilotprojekt „Mobile Navigation 3D - 3D Stadtmodelle für mobile Navigationssysteme“(MoNa 3D) von der Firma GTA Geoinformatik, Heidelberg Mobil, Navigon und Tele Atlas. Unterstützt wurden die Firmen durch die Fachhochschule Mainz, die Universität Bonn und die Hochschule für Technik Stuttgart.

Denkmalschutz
Um bestehende historische Gebäude zu Präsentationszwecken oder auch zerstörte Bauwerke zu rekonstruieren, kann ein Modell angefertigt werden. Dabei handelt es sich meist um 3D Modelle der Genauigkeitsstufen LoD 2-4. Aufgrund des hohen Kosten- und Zeitaufwands beim Erfassen historischer Datenbestände wird ein Modell jedoch nur in Einzelfällen erstellt.

Analyse und physikalische Simulation
Die gespeicherten Daten in 3D Stadtmodellen können nicht nur zur Visualisierung, sondern auch zur Analyse und Simulation genutzt werden. Durch ein integriertes Höhenmodell und die Gebäudehöhen im Modell kann die Ausbreitung von Funkwellen eines Funkturms im Modell simuliert werden, um einen passenden Standpunkt für eine höchstmögliche Netzverfügbarkeit zu finden.
Im Katastrophenschutz finden 3D Stadtmodelle häufig in Überflutungssimulationen, Feuerausbreitungen oder zur Simulation von Gas- und Luftbewegungen (z.B. Verbreitung von Abgasen) Anwendung. Dadurch kann ermittelt werden, welche Bereiche besonders gefährdet sind, um an erkannten Schwachpunkten entsprechende Schutzmaßnahmen zu ergreifen.
Weiterhin können sie eingesetzt werden, um die Lärmausbreitung in bestimmten Gebieten zu ermitteln, um zulässige Grenzwerte zu kontrollieren und entsprechende Gegenmaßnahmen zu ergreifen. Eingesetzt wird dies vorrangig beim Flughafen- oder Straßenbau [Rhe09].
Durch die modellierten Dachflächen in 3D Stadtmodellen können Neigung und Richtung von Dachflächen zur Solarpotentialanalyse genutzt werden. Die Firma GTA Geoinformatik hat beispielsweise eine Software entwickelt, die eine automatisierte Erstellung eines Solarpotentialkatasters unter Einbeziehung von Verschattungen anbietet.
Ebenso finden 3D Stadtmodelle in der Ausbildung von Sicherheitskräften, sowie bei Rettungs- und Löscheinsatzsimulationen Anwendung, die in der Realität zu gefährlich wären. Weiterhin werden sie im militärischen Bereich verwendet. Dazu gehört die Ausbildung von Streitkräften durch verschiedene Simulationen, wie z.B. Flug- und Gefechtssimulationen [Rhe09]. Ferner finden 3D Stadtmodelle auch in der Planung militärischer Einsätze durch Verwendung realitätsnaher Modelle Anwendung.

3.4. Anforderungen und Eigenschaften

Abhängig von der jeweiligen Anwendung, variieren die Anforderungen an 3D Stadtmodelle. Während private Nutzer mehr Wert auf realitätsnahe Darstellungen und photorealisitische Texturen legen, liegt der Schwerpunkt in der gewerblichen Nutzung in der Effektivität. Die Erstellung eines detaillierten 3D Stadtmodells kostet verhältnismäßig viel Geld. Daher muss vorher genau abgewogen werden, ob sich eine Erstellung für den Erzeuger hinsichtlich Kosten und Nutzen auszahlt.
Die folgenden Anforderungen an 3D Stadtmodelle richten sich aufgrund der vielfältigen Nutzung nach der jeweiligen Anwendungsart [AG 04].

Anschaulichkeit
Eine Visualisierung ist in den meisten Projekten die Hauptanforderung. Ausnahmen bilden Modellierungen für Analysen.

Wiedererkennungswert
Durch einen realitätsnahen Eindruck wird die Orientierung des Anwenders erleichtert.

Simulation / Navigation
Die freie Navigation im Stadtmodell ist für die meisten Anwendungen ebenso essentiell wie die Simulation bestimmter Szenarien (z.B. Schattenwurf an bestimmten Tageszeiten) [LL97].

Unterschiedliche Detaillierung
Zur qualitativen Unterscheidung sind verschiedene Detaillierungsgrade erforderlich. Für unterschiedliche Anwendungsbereiche stehen auch unterschiedliche Versionen des Modells zur Verfügung.

Wirtschaftlichkeit
Um die Kosten möglichst gering zu halten, ist bei der Erstellung von 3D Stadtmodellen ein hoher Automatisierungsgrad erforderlich.

Integrität / Aktualität / Redundanzarmut
Die in dem Modell gespeicherten Daten müssen korrekt gespeichert werden und aktualisierbar sowie möglichst redundanzfrei sein [Bri08].

Wie bereits erwähnt, richten sich die Anforderungen nach der Intention des Auftraggebers, bzw. des Nutzers. Allgemein sollte ein 3D Stadtmodell jedoch vielfältig einsetzbar und erweiterbar sein. Gemeinsame Anforderungen sind in jedem Modell jedoch immer korrekte Topologie, Geometrie und Semantik [Bri08].

3.4.1. Geometrie

Die Geometrie beschreibt die räumliche Lage im zwei- oder dreidimensionalen Raum. Sie umfasst alle Angaben zur absoluten räumlichen Lage und Ausdehnung eines Objekts auf der Basis eines räumlichen Bezugssystems. Zur Beschreibung von Geometrien in 3D Stadtmodellen wird für gewöhnlich das Vektormodell genutzt. Die Alternative bildet das Rastermodell [Bri08].
Das Vektormodell baut auf Punkten und Linien auf. Die Lage der Punkte wird über Koordinaten eines Bezugssystems eindeutig beschrieben. Auf dieser Basis werden die für die 3D Stadtmodelle wichtigen Polygone definiert.
Das Grundelement im Rastermodell ist die Rasterzelle, eine spezielle Form von Mosaikmodellen. Die Rasterzellen zerteilen den Datenraum in gleichförmige, meist rechteckige oder quadratische Teilflächen, die besser als Pixel bekannt sind. Da die geometrische Genauigkeit durch die Größe der Rasterzelle abhängt, kann es bei thematischen Eigenschaften eines Gebietes zu einem Verlust an Genauigkeit kommen. Aufgrund ihrer Eigenschaften wird das Rastermodell vorwiegend für Texturen verwendet [Sta07].

3.4.2. Topologie

In den Geowissenschaften beschreibt die Topologie die relativen räumlichen Beziehungen von Objekten zueinander und wird daher auch als Geometrie der relativen Lage

bezeichnet. Es wird von der Geometrie (Form, Größe und Lage) abstrahiert, um die topologischen Beziehungen der Objekte zu modellieren [Bri08]. Die Topologie beschreibt demnach Eigenschaften, die invariant gegenüber geometrischen Transformationen sind. Die topologischen Grundformen sind Knoten (nulldimensionale topologische Primitive ohne Begrenzung), Kante (eindimensional und durch zwei Knoten begrenzt) und Masche (zweidimensional und von mindestens drei Kanten umrandet, sowie geschlossen, orientiert und nicht selbst schneidend). [Tis09]

3.4.3. Semantik

Allgemein beschreibt die Semantik die Bedeutung bzw. den Inhalt von Zeichen und ist demnach Teilgebiet der Linguistik, welche sich wiederum in die Teilgebiete der Bedeutungslehre (Semasiologie) und der Bezeichnungslehre (Inimasiologie) teilt. Die Semasiologie beschäftigt sich mit der Frage, was bezeichnet wird (z.B. kann ein Schloss ein Gebäude oder Objekt zum verschließen sein). Die Inimasiologie hingegen beschäftigt sich mit der Frage, wie etwas bezeichnet wird (z.B. kann Geld als Mäuse, Moos oder Kohle bezeichnet werden) [SK07]. In den Geowissenschaften bildet die Semantik aber vor allem eine Ergänzung zur Geometrie, indem dem Objekt nicht geometrische Attribute zugewiesen werden (Adresse, Nutzungsarten, Zustände) [Sta07].

3.4.4. Detaillierungsgrad

3D Stadtmodelle zeichnen sich durch ihren Detaillierungsgrad aus. Wie bereits erwähnt, entscheidet die Art der Nutzung des Modells über den Detaillierungsgrad. In der Tourismusbranche ist z.B. ein hoher Wiedererkennungswert mit einer entsprechend hohen Genauigkeit notwendig. Um diese Stufen zu charakterisieren, wurden die Level of Details (LoDs) definiert.
Laut der SIG 3D, der Gesellschaft für Geodateninfrastruktur Deutschland (GDI DE) wird ein 3D Stadtmodell in Basis- und Anwendungsmodell unterschieden. [Spe]
Das Basismodell stellt dabei die universelle Grundlage für thematische Modellierungen und muss offen für verschiedene Anwendungsszenarien sein. Es besteht aus Primitiven für 0-3 dimensionale Geometrieobjekte (Knote, Kante, Masche, Volumenkörper) und kann zu Aggregaten der entsprechenden Dimension zusammengefügt werden, welche wiederum die Grundsteine für Anwendungsobjekte im Anwendungsmodell bilden.
Das Anwendungsmodell bildet thematische, sowie weitere, zum Objekt gehörende Strukturen ab. Bei der Abbildung werden die von der SIG 3D entworfenen LoDs unterschieden. Die SIG 3D legt die LoDs in fünf, statt wie von [KB98] beschriebenen drei Stufen fest. Dabei reicht der Detaillierungsgrad von einem digitalen Geländemodell über ein simples Klötzchenmodell bis hin zu einem realitätsnahen Innenraummodell. [GKD+04]
Die Tabelle 1 aus [GKD+04] beschreibt die LoD Stufen 0-4 und ihre Besonderheiten.

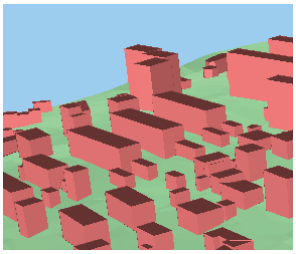
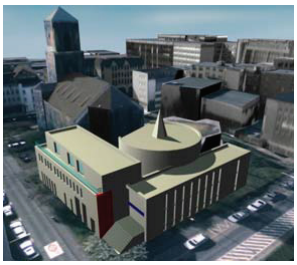
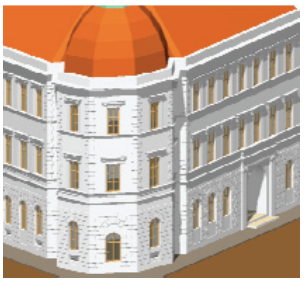
Überblick über die Detaillierungsgrade LoD0 bis LoD4 Die angegebenen Genauigkeiten sind Richtwerte	
	**LoD 0 - Regionalmodell** DGM (2,5D) mit Textur/Orthophoto und Flächennutzung Erfassungsgeneralisierung: maximal; Klassifizierung nach Flächennutzung Dachform/-struktur: keine Punktgenauigkeit (Lage/Höhe): >5m / >5m
	**LoD 1 - Stadt- / Standortmodell** Klötzchenmodell ohne Dachstrukturen Erfassungsgeneralisierung: Objektblöcke in generalisierter Form > 6m*6m Grundfläche Dachform/-struktur: ebene Flächen Punktgenauigkeit (Lage/Höhe): 5m / 5m
	**LoD 2 - Stadt- / Standortmodell** Texturierte Modelle; differenzierte Dachstrukturen; Vegetationsmerkmale (z.B. Bäume) Erfassungsgeneralisierung: Objektblöcke in generalisierter Form > 4m*4m Grundfläche Dachform/-struktur: Dachtyp und Ausrichtung Punktgenauigkeit (Lage/Höhe): 2m / 1m
	**LoD 3 - Stadt- / Standortmodell** Geometrisch fein ausdifferenzierte Architekturmodelle; Vegetation; Straßenmöblierung Erfassungsgeneralisierung: Objekte in realer Form; > 2m*2m Grundfläche Dachform/-struktur: reale Form Punktgenauigkeit (Lage/Höhe): 0,5m / 0,5m
	**LoD 4 - Innenraummodell** Begehbare Architekturmodellen Erfassungsgeneralisierung: reale Form; Abbildung konstruktiver Elemente und Öffnungen Dachform/-struktur: reale Form Punktgenauigkeit (Lage/Höhe): 0,2m / 0,2m

Tabelle 1: LoD Stufen aus [GKD+04]

3.5. CityGML

CityGML ist ein Extensible Markup Language (XML) basiertes Geography Markup Language (GML) Anwendungsschema zum Austausch und zur Speicherung von 3D Stadtmodellen. Es gehört seit dem 13. August 2008 zum Open Geospatial Consortium (OGC) Standard und ist aktuell[1] als Version 2.0.0 verfügbar. CityGML wird von der SIG 3D entwickelt und gepflegt.
In diesem Standard werden alle zur Modellierung von 3D Stadtmodellen notwendigen Objekte abgebildet. Dazu gehören vor allem Gebäude und Stadtmöblierungen. Vegetation und Gelände, sowie Wasser- und Verkehrsflächen können ebenso modelliert werden. Jedes Objekt kann in verschiedenen LoDs abgebildet werden. Die in Abschnitt 3.4 beschriebenen Eigenschaften Geometrie, Topologie und Semantik werden von CityGML unterstützt. Dafür werden andere Standards des OGC und der International Organization for Standardization (ISO) Normen 191XX verwendet. Dazu zählen u.A. das Spatial Schema (ISO 19107) und das Simple Feature Model.

[1] Stand:04.07.2012

4. Datenbanken

Sowohl der CityServer3D als auch die 3D City Database sind auf einem Datenbanksystem (DBS) aufgebaut. In diesem Kapitel werden die Vorteile von Datenbanken sowie die speziellen Eigenschaften von Geodatenbanken erläutert. Des Weiteren wird ein Überblick zu den beiden Datenbanklösungen CityServer3D und 3D City Database gegeben.

Ein DBS dient der effizienten und persistenten Speicherung großer Datenmengen. Persistenz bezeichnet in diesem Zusammenhang die Fähigkeit, Objekte und deren logische Verbindungen über einen langen Zeitraum bereit zu halten. Ohne Kenntnis der internen Speicherrealisierung bieten Datenbanken eine Datenbanksprache zur Verwaltung sowie Mechanismen zum Datenschutz, zur Datensicherheit und der Datenintegrität an.
Die Strukturierung und die Beziehungen der Daten zueinander werden durch das Datenbankmodell realisiert. Zu den bekanntesten Datenbankmodellen gehören das hierarchische, das Netzwerk-, das relationale, das objektrelationale und das objektorientierte Datenbankmodell. Weitere Datenbankmodelle sind sog. Not only Structured Query Language (NoSQL) Datenbanken. Dazu gehören beispielsweise Dokumentenorientierte oder Key-Value Datenbanken. Das relationale und objektorientierte Modell sowie deren Mischform haben sich als die heute am meisten verwendeten Datenbankmodelle etabliert.

4.1. Datenbankvorteile

Die Alternative zur Speicherung von Daten in einer Datenbank ist die Speicherung in einem Filesystem. Die Vor- und Nachteile von File- und Datenbanksystemen werden in Tabelle 2 kurz skizziert.

	Filesystem (FS)	Datenbanksystem (DBS)
Vorteile	Ordnerhierachie durch Betriebssystem unterstützt schnelle Datenspeicherung	Datenredundanz kontrollierbar Mehrbenutzerkontrolle semantische Abfragen Versionierung zentrale Verwaltung
Nachteile	kein semantischer Datenzugriff keine Integritätskontrolle keine Sicherheit bei undef. Zuständen	komplexer als FS Datenmodellierung entscheidend

Tabelle 2: Vor- / Nachteile von File- und Datenbanksystemen

Datenbanken sind speziell entwickelt worden, um komfortabel und nutzergerecht große Datenmengen effizient zu verwalten. Vor allem aufgrund der semantischen Anfragesprache und der Mehrbenutzerverwaltung eignen sich Datenbanken besser zur Speicherung von wichtigen und viel genutzten Daten.

4.2. Geodatenbanken

Da die Anforderungen zur Speicherung von Geodaten sehr speziell sind, muss das Datenbankmodell diese unterstützen. In dieser Arbeit sollen Datenbanklösungen für räumliche Daten untersucht werden. Der folgende Abschnitt beschreibt die speziellen Anforderungen und Eigenschaften von Geodatenbanksystemen.
Räumliche Datenbanksysteme bzw. Geodatenbanksysteme dienen der Speicherung von Geodaten und unterstützen die Bearbeitung räumlicher Anfragen. Aufgrund dieser Eigenschaften bieten sich Geodatenbanksysteme für die Speicherung von 3D Stadtmodellen an.
Geodatenbanksysteme zeichnen sich durch die Verwendung von Geoobjekten aus. Zu den speziellen Eigenschaften gehört vor allem ein Geometrieattribut, welches für geometrische Operationen verwendet wird. Dafür werden eigene Datentypen wie Punkte, Streckenzüge oder Polygone benötigt. Weiterhin muss die Geodatenbank Methoden zur Ausführung geometrischer Operationen unterstützen, um z.B. Objekte eines bestimmten Bereichs auszuwählen. Laut [Bri08] müssen Anfragen mit Raumbezug auf räumliche Basisoperationen zurückgeführt werden. Beispielsweise muss eine Rechteckanfrage alle Geoobjekte in dem spezifizierten Bereich bestimmen. Die Verarbeitung solcher und geometrischer Operationen muss effizient von Algorithmen und Datenstrukturen vom DBMS unterstützt werden. Weiterhin muss die Interoperabilität der gespeicherten Geodaten gewährleistet werden. Gemeint sind allgemein anerkannte Standards (z.B. GML), damit die Daten auch von anderen Applikationen genutzt werden können [Bri08].
Im Folgenden wird die Eignung relationaler und objektrelationaler Datenbankmodelle für Geodaten nach den o.g. Kriterien nach [Bri08] untersucht.

4.2.1. Eignung von relationalen Datenbanken

Um dreidimensionale Geometrien abzubilden, werden geometrische Grundformen wie Punkte, Streckenzüge, Ringe, Polygone und Multipolygone benötigt. Streckenzüge, Ringe, Polygone und Multipolygone können nicht als atomare Attribute in relationalen Datenbanken gespeichert werden. Um sie dennoch in einer relationalen Datenbank abzubilden, müssen diese in mehreren Tabellen aufgeteilt werden, die wiederum untereinander über Beziehungen verknüpft sind. So besteht ein Multipolygon bspw. aus mehreren Polygonen.
Die Abbildung von simplen Geometrien ist mit diesem Datenmodell zwar möglich, aber äußerst komplex und weist einige funktionale Nachteile auf. Zusammengehörende Informationen werden in diesem Datenmodell auf mehrere Tabellen verteilt. Laut [Bri08] kann z.B. ein Multipolygon abgebildet werden, indem eine Datenbankabfrage über mindestens drei Tabellen ausgeführt wird. Dabei wird eine topologische Abfrage über die geometrischen Primitiven Knoten, Kante und Masche durchgeführt, welche über Beziehungen miteinander verbunden sind. Da die gespeicherten Daten in relationalen Datenbanksystemen nicht nach räumlichen Kriterien verwaltet werden, können räumliche Basisanfragen weder effizient noch mit der zugehörigen Datenbanksprache Structured Query Language (SQL) ausgeführt werden [Bri08].

4.2.2. Eignung von objektrelationalen Datenbanken

Zur Verwaltung von Geodaten sind geometrische Datentypen und Funktionen erforderlich. Mithilfe des objektorientierten Ansatzes können diese sowohl in objektorientierten, als auch in objektrelationalen Datenbankmodellen definiert werden. Datensätze, die die jeweiligen räumlichen Anfragebedingungen erfüllen, müssen von dem DBS effizient ausgewählt werden. Um dies umzusetzen, müssen Algorithmen und Datenstrukturen in das DBMS eingefügt werden.
Diese Anforderungen werden von den meisten Herstellern von objektrelationalen Datenbanksystemen erfüllt, indem die entsprechenden Datentypen, Funktionen und Erweiterungskomponenten zur Verfügung gestellt werden. Nachteil ist, dass die Erweiterungskomponenten jeder Firma unterschiedlich konzipiert sind. Dadurch werden die Anwender an einen Hersteller gebunden.
Objektorientierte und objektrelationale Datenbanksysteme sind für die Verwaltung von Geodaten geeignet, da sie die objektorientierten Ansätze aus der Software Entwicklung übernehmen und dadurch eigene Datentypen, Klassen und Methoden verwendet werden können. Objektorientierte Datenbanksysteme konnten sich bislang jedoch nicht auf dem Markt etablieren. Laut [Bri08] liegt dies an der kompletten Umstellung eines laufenden Systems. Weiterhin sind viele Anwendungsprogramme speziell auf relationale Datenbanken eingestellt. Ein Umstieg ist für die meisten Firmen daher nicht rentabel. Die Entwicklung geht daher in Richtung objektrelationale Datenbanksysteme, da sie die Vorteile relationaler und objektorientierter Modelle vereint [Bri08].
Bekannte Hersteller objektrelationaler Geodatenbanken mit zugehörigen Geometrieerweiterungen sind Oracle (Oracle Spatial), PostgreSQL (PostGIS), IBM Informix (DataBlade) und DB2 (Spatial Extender).

4.3. CityServer3D

Der CityServer3D ist ein kommerzielles Produkt zur Verwaltung von zwei- und dreidimensionalen geographischen Daten. Es wird vom Fraunhofer Institut für Graphische Datenverarbeitung (IGD) entwickelt und gepflegt. Die CityServer3D Plattform besteht aus mehreren Komponenten, welche die Verwaltung, Visualisierung und Analyse von 3D Stadtmodellen übernehmen.
In dieser Arbeit wird auf die Datenhaltungskomponente eingegangen. Es handelt sich dabei um eine Geodatenbank mit einer Client-Server Architektur. Der Server übernimmt die Funktion der Datenspeicherung. Die Datenbanklösung ist sowohl für relationale als auch objektrelationale und dokumentenorientierte Datenbanken kompatibel. Es werden große Datenvolumen sowie die bekanntesten (Geo-)Datenformate unterstützt.
Der CityServer3D benutzt ein eigenes objektrelationales Datenbankschema und ist durch seine mehrschichtige Architektur unabhängig vom verwendeten DBMS. Mithilfe des Datenmodells, dem sogenannten Metamodell, werden die Geometriedaten der Dateiformate aufgelöst sowie einheitliche Standardoperationen für Geodaten implementiert. Dieses Metamodell bildet den Kern des vollständig in Java implementierten Systems und ist laut [Gre06] in großen Teilen konform zur OpenGIS Simple Feature Specification [Vre10].

Da der CityServer3D ein kommerzielles System ist, sind die Interna nicht einsehbar. Abbildung 6 aus [Use05] gibt einen Überblick der Anwendungen und Schnittstellen des CityServer3D:

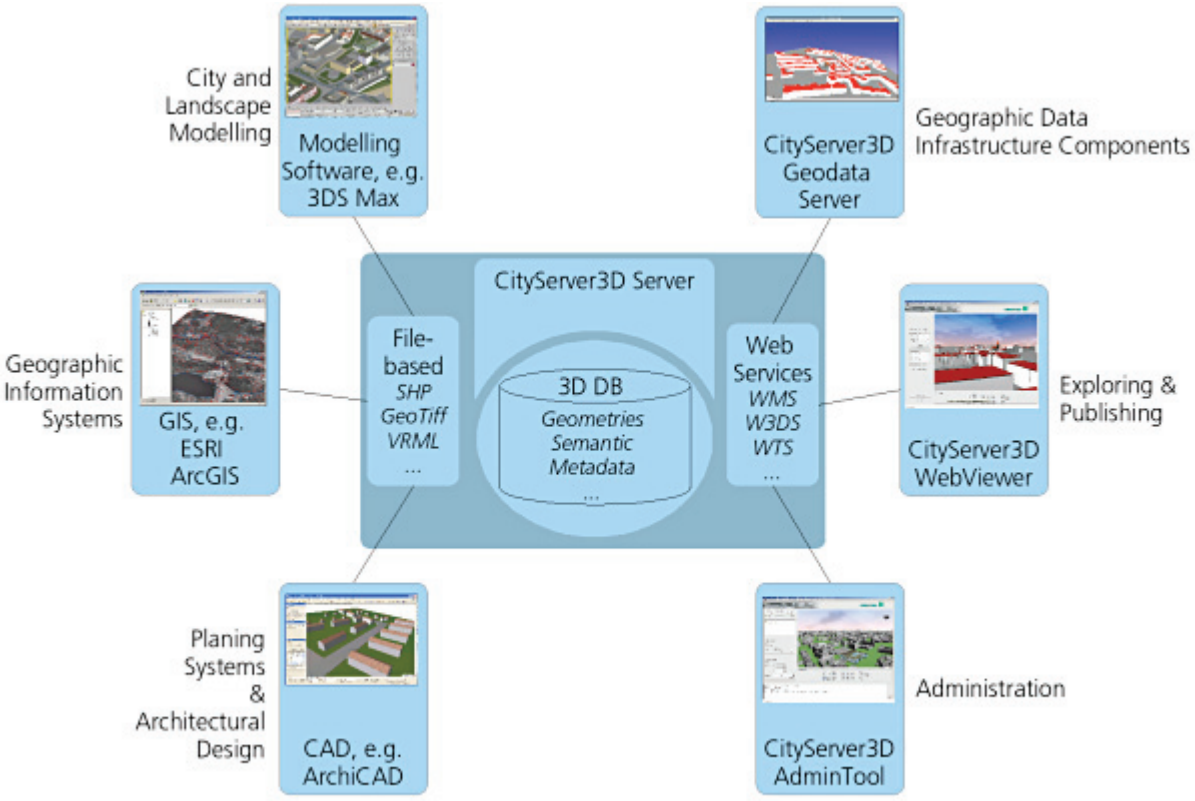


Abbildung 6: Aufbau CityServer3D aus [Use05]

Der Client (Admin Tool) übernimmt neben der Visualisierung der Daten auch den Im- und Export von Geodaten. Die Daten werden sowohl beim Im-, als auch beim Export in das Metamodell überführt. Die graphische Oberfläche erlaubt es, die Sach- und Metadaten strukturell zu verändern. Mithilfe eines integrierten regelbasierten Systems wird zusätzlich ein anwendungsbezogener Zugriff und ein einheitliches Datenmanagement ermöglicht. Zudem können Rasterdaten wie digitale Geländemodelle und Orthophotos abgebildet werden. Durch Nutzung des Metamodells als Schnittstelle ist diese Datenbanklösung für verschiedene Datenbanksysteme, wie z.B. MySQL, PostgreSQL, MongoDB oder Oracle kompatibel und demnach formatunabhängig. Eine Auflistung der Eigenschaften des gesamten Softwarepaketes CityServer3D befindet sich in der Tabelle 10 in Kapitel 5.5.

4.4. 3D City Database

Die 3D City Database ist eine frei verfügbare 3D Geodatenbank zur Speicherung, Repräsentation und Verwaltung von digitalen 3D Stadtmodellen. Die Datenbank wurde vom Institut für Geodäsie und Geoinformationstechnik (IGG) der Technischen Universität Berlin im Auftrag der Berliner Senatsverwaltung für Wirtschaft, Arbeit und Frauen und der Berlin Partner GmbH entwickelt.

4.4.1. Überblick

Das Hauptmerkmal der Geodatenbank ist die komplette Realisierung der CityGML Klassen zur Darstellung von 3D Stadtmodellobjekten bezüglich ihrer geometrischen, topologischen, semantischen und sichtbaren Oberflächeneigenschaften.
Das zugrunde liegende Datenbankmodell ist das objektrelationale Datenbankmanagementsystem (ORDBMS) von Oracle Spatial 10G oder höher. Durch die nahezu ausschließliche Nutzung von Oracle Spatial Datentypen werden projektbezogene Speziallösungen vermieden. Mithilfe des Geometriepaketes können räumliche 2D - 4D Datentypen abgedeckt werden. Der Oracle Spatial Datentyp wird auch von kommerziellen GIS mit Datenbankanbindung unterstützt. Die Oracle Erweiterung stellt zudem umfangreiche Methoden zur effizienten Verwaltung georeferenzierter Rasterdaten zur Verfügung. Die Versions- und Historienplanung wird mithilfe des Oracle Workspace Managers umgesetzt [KKN12].
Abbildung 7 aus [Her11] gibt einen Überblick zur 3D City Database.

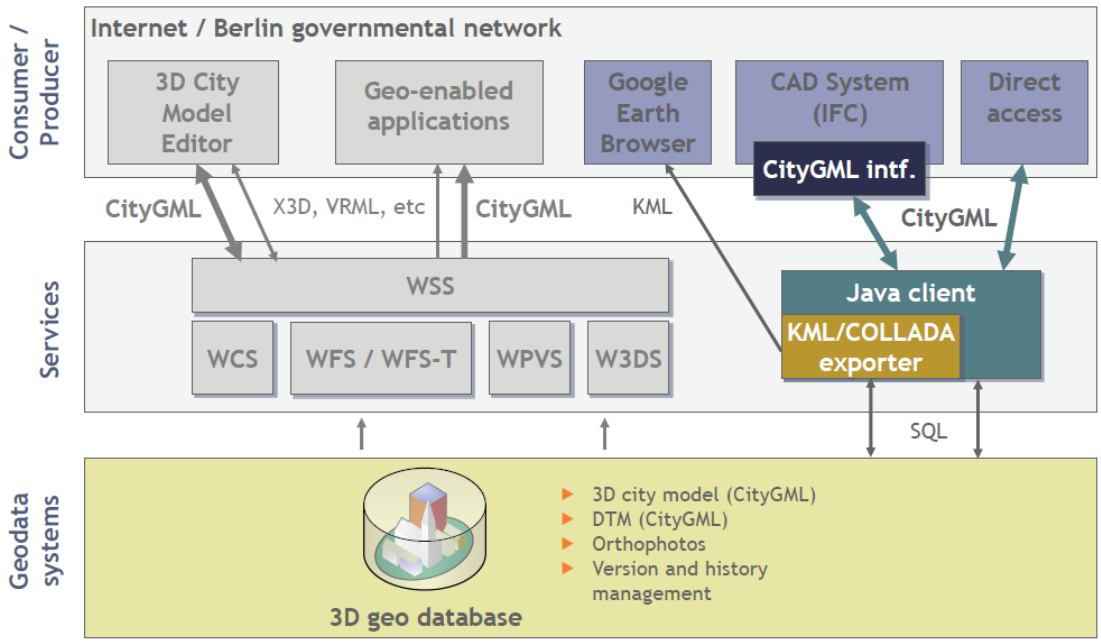


Abbildung 7: Übersicht 3D City Database aus [Her11]

Erkennbar ist die 3D Geodatenbank (Oracle Spatial) als Basis der Geodaten Infrastruktur (GDI). Als Service ist hier der Java Client dargestellt, welcher neben dem Import von CityGML konformen Dateien auch den Export in verschiedene Dateiformate übernimmt. Das Im- und Export Tool für Rasterdaten und Orthophotos ist hier nicht abgebildet, da es für die Vorgängerversion der 3D City Database an der Universität Bonn entwickelt wurde und nicht Teil des am IGG Berlin entwickelten Projekts ist. Die grauen Bereiche im Service/ Consumer/ Producer Bereich sind noch nicht implementiert. Allerdings können diese durch Nutzung standardisierter Schnittstellen als OGC konforme Webschnittstellen entwickelt werden.

4.4.2. Datenmodellierung

Im folgenden Abschnitt wird vereinfacht auf einen Teil der Datenmodellierung in der Geodatenbank eingegangen. Die Auswahl erfolgt im Hinblick auf die notwendigen Datenmodellierungen zur Fortführung von Gebäudemodellen. Um Gebäude in der 3D City Database zu speichern werden das *Core Model*, das *Building Model* und das *Appearance Model* benötigt. Wie bereits erwähnt, realisiert die 3D City Database die Klassen und Verbindungen des CityGML Standards. Im Anhang A befinden sich analog zu den hier gezeigten Entity Relationship (ER) Modellen die entsprechenden Unified Modeling Language (UML) Diagramme des CityGML Standards. In der semantischen Datenmodellierung dient ein ER Modell der Typisierung von Objekten und ihren Beziehungen zueinander. Abbildung 8 stellt das Core Model nach [KKN12] dar:

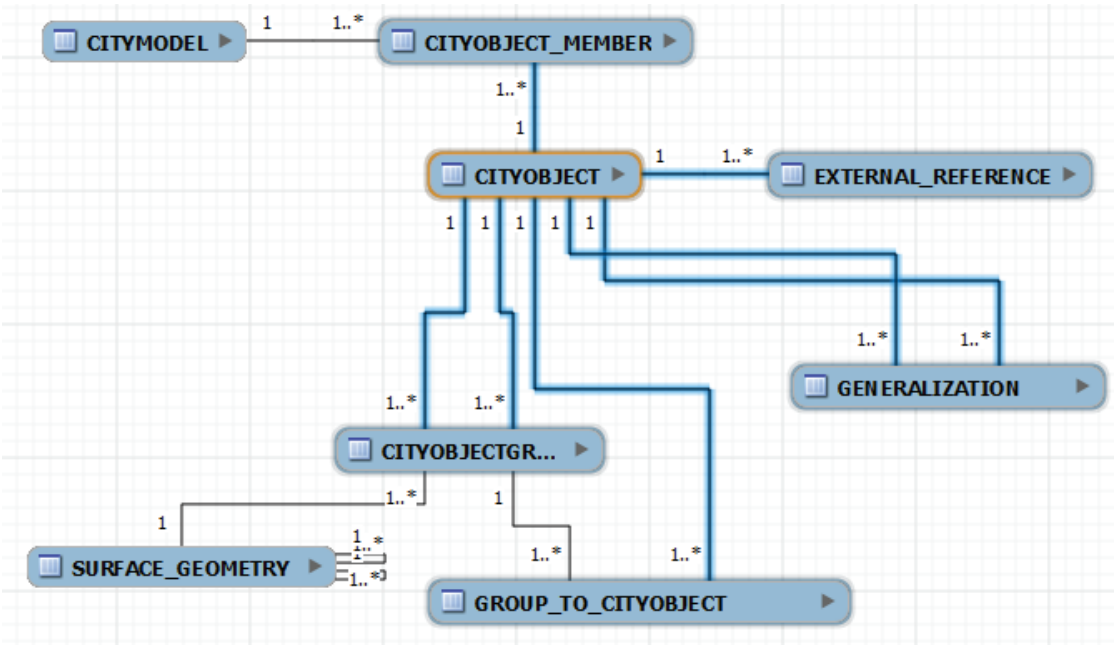


Abbildung 8: Core Model der 3D City Database nach [KKN12]

Die Basis aller thematischen Objekte ist durch den Entitätstyp *CityObject* realisiert. Alle in der Datenbank gespeicherten Objekte können auf diese Basisklasse zurückgeführt werden. 3D Stadtmodellobjekte zusammen mit ihren Unterklassen *Buildings*, *CityFurnitures*, *Appearances* etc. werden als Datensätze (Tupel) in der Tabelle *CityObject* gespeichert. Durch die Tabelle *Generalization* können dieselben Geoobjekte in unterschiedlichen LoDs abgebildet werden. Es können mehrere Tupel der *CityObject* Tabelle zu einem oder mehreren *CityModels* zusammengefasst werden. Durch eine N:M Beziehung kann dies auch umgekehrt geschehen. Weiterhin kann jedes *Cityobject* beliebig viele externe Referenzen besitzen. *CityObjects* können zu einer *CityObjectGroup* zusammengefügt werden, welches wiederum ein eigenes *CityObject* bildet. Angewendet wird dies z.B., um Räume einem bestimmten Stockwerk oder Gebäude zu einem Komplex zuzuordnen. Das *Core Model* legt die grundlegenden Eigenschaften eines beliebigen *Cityobjects* fest. Appearances entsprechen dem äußeren Erscheinungsbild (Theme) eines Stadtmodells. Um Informationen über das Aussehen einer Oberfläche zu speichern, wird das in der CityGML spezifizierte *Appearance Model* genutzt. Abbildung 9 nach [KKN12] zeigt die Umsetzung des *Appearance Models* in die objektrelationale Datenbankstruktur der 3D City Database:

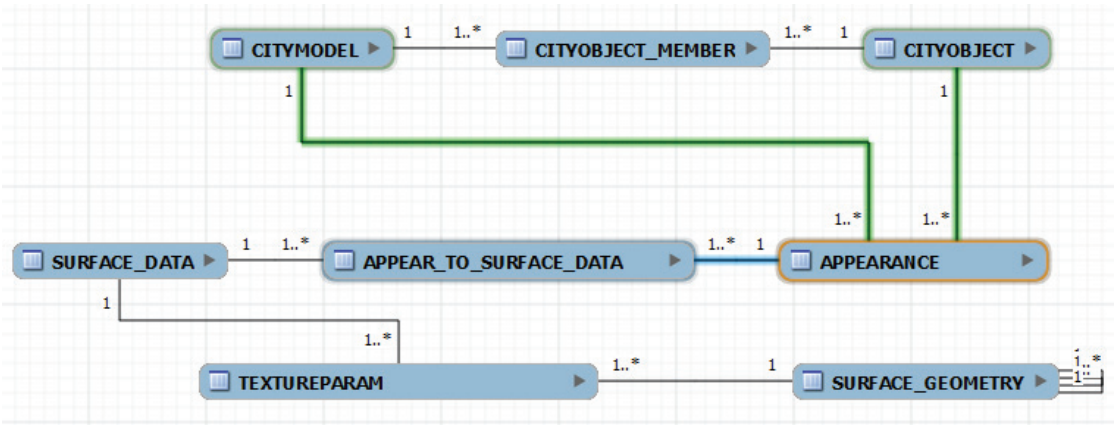


Abbildung 9: Appearance Model der 3D City Database nach [KKN12]

Es werden beliebig viele Appearances pro Stadtmodell unterstützt. Jedes LoD eines Geoobjekts kann individuelle Appearances besitzen. In der Abbildung ist erkennbar, dass jedes *CityObject* oder *CityModel* eigene Appearances besitzen kann. Dabei wird in der Tabelle *Appearance* nur ein Theme festgelegt (Name), welches über die Verknüpfungstabelle *Appear_To_Surface_Data* mit Eigenschaften (Farben, Texturen, Transparenz) in der *Surface_Data* Tabelle verlinkt wird. Die räumliche Ausdehnung der entsprechenden Flächen wird wiederum über die Relationstabelle *Textureparam* im Zusammenhang mit der *Surface_Geometry* Tabelle realisiert.

Im folgenden Abschnitt erfolgt ein kurzer Überblick zum Gebäudemodell, dem wichtigsten und umfangreichsten Modell für die Arbeit an der 3D City Database. In Abbildung 10 nach [KKN12] wird das vereinfachte ER Modell des *Building Models* mit Fokus auf den *Building* Entitätstyp dargestellt:

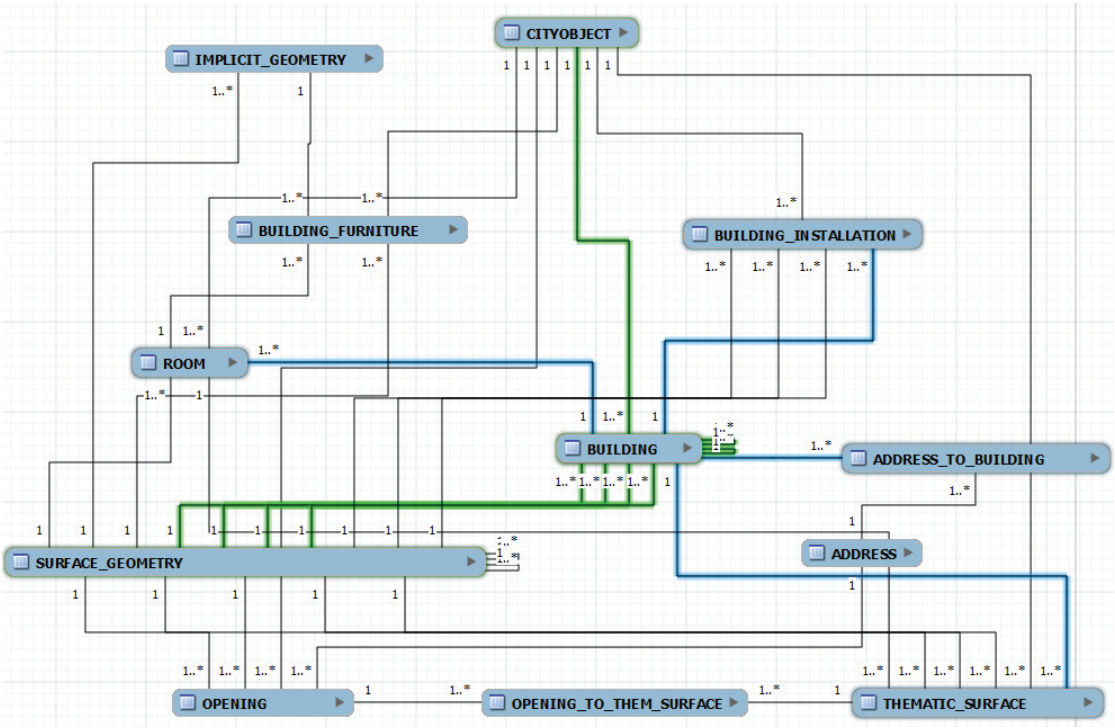


Abbildung 10: Building Model der 3DCityDB Fokus Building nach [KKN12]

Ein Gebäude kann in vier verschiedenen Genauigkeitsstufen als Tupel in der Tabelle *Building* abgebildet werden (LoD1 - LoD4). Mithilfe dieses Modells können sowohl einfache Gebäude mit nur einer Komponente, als auch komplexe Strukturen abgebildet werden. Die drei CityGML Klassen *AbstractBuilding*, *Building* und *BuildingPart* (vgl. Anhang A.3) werden in dem Entitätstyp *Building* vereinigt. Die rekursive Abbildung von Gebäuden (z.B. *Buildingparts*) erfolgt demzufolge innerhalb dieser Tabelle und wird durch zwei 1:N Beziehungen als Fremdschlüssel realisiert. Konkret ensteht dadurch eine Komponentenhierachie, welche jeweils zu dem Wurzelobjekt, bzw. zu seinen Aggregaten (abgeleitete Gebäude) verlinkt. Ein *Building* Objekt kann weiterhin mit den Entitätstypen *Room* (Räume), *Building_Installation* (Gebäudeinstallationen), *Address* (Adressen) und *Thematic_Surface* (z.B. Wand- und Dachflächen) und den daraus abgeleiteten Objekten wie *Building_Furniture* (Gebäudemöblierungen) oder *Opening* (Gebäudeöffnungen) verknüpft werden. Diese Entitätstypen (einschließling *Building*) bilden wiederum jeweils ein eigenes *CityObject*.
Der *Surface_Geometry* Entitätstyp beinhaltet in einem Standardanwendungsfall die meisten Informationen. In ihr werden sämtliche zu den o.g. Objekten gehörigen teilweise

sehr komplexen Geometrien abgebildet. Abbildung 11 zeigt, mit welchen Entitätstypen *Surface_Geometry* im *Building Model* verbunden ist.

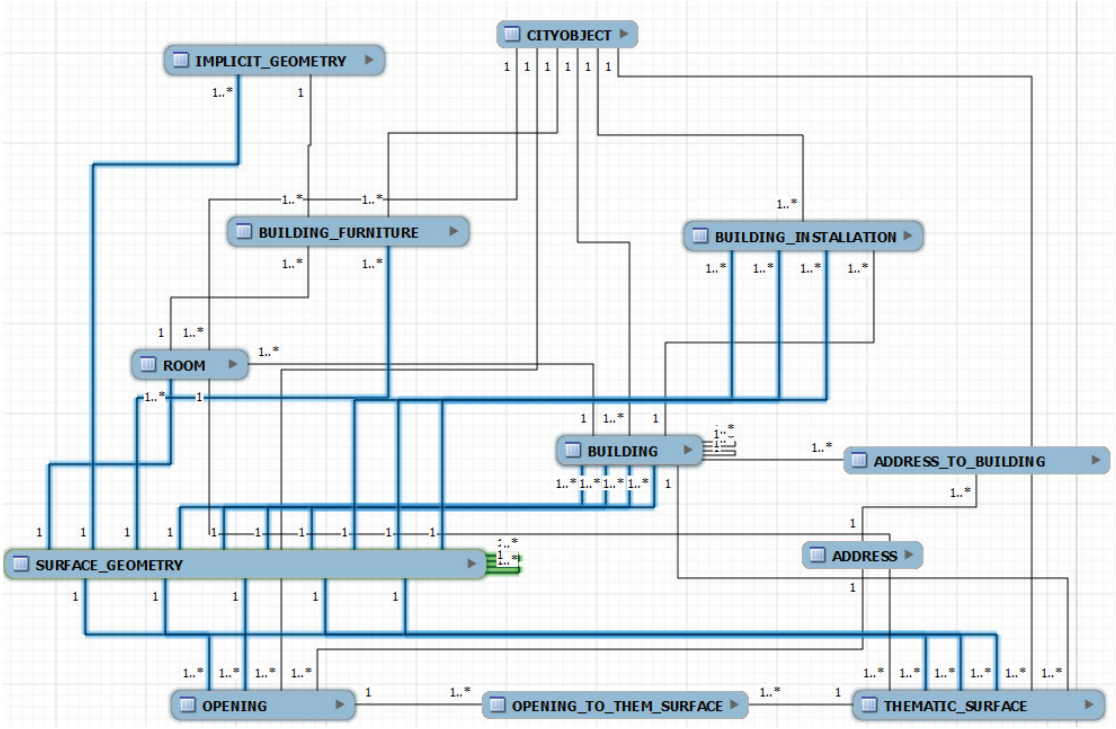


Abbildung 11: Building Model der 3DCityDB Fokus Geometry nach [KKN12]

Hier sind vor allem die 1:N Beziehungen des Entitätstyps erkennbar. Demzufolge können mehrere Objekte (*Building*, *Room*, etc.) lediglich einer einzigen Geometrie zugeordnet werden. Um dennoch komplexe Geometrien abzubilden, wird auch hier eine Komponentenhierachie durch Wurzelgeometrie und Aggregate durch Fremdschlüssel innerhalb des Entitätstyps realisiert. Dadurch können beliebig viele Geometrien zu einem Objekt gespeichert werden.
Neben den *Core*, *Appearance* und *Building Models* sind in der 3D City Database auch die restlichen CityGML Klassen als ER Modelle abgebildet. Dazu gehören z.B. das *CityFurniture*, *Transportation* oder *LandUse Model*. Eine vollständige Liste der CityGML Klassen inklusive deren Umsetzung ist in dem Dokument [KKN12] zu finden. Allen Modellen liegt das Geometrie - und Topologiemodell der GML3 Spezifikation zu Grunde, auf das hier nicht näher eingegangen wird.

5. Datenhaltung

In diesem Kapitel wird die Durchführung verschiedener Tests beschrieben. Da das CityGML Dateiformat ein Standard in der 3D Stadtmodellierung ist und von beiden Datenbanklösungen unterstützt wird, werden unterschiedliche CityGML Dateien im- und exportiert. Mithilfe von Orthophotos kann ein 3D Stadtmodell einen realitätsnäheren Eindruck vom Gebiet erhalten. Aufgrund dessen wird ebenso der Im- und Export eines Orthophotos getestet.
Diese beiden Datentypen entsprechen dem in der Analyse (Kap. 2) gezeigten Arbeitsprozess zur Nutzung von Geodaten. Beide Tests prüfen, ob die importierten Datensätze in der Datenbank bzw. nach dem Export korrekt abgebildet werden und keine Informationen verloren gehen.
Weiterhin zeigt dieses Kapitel einen Performance- und Featurevergleich, um den CityServer3D und die 3D City Database miteinander zu vergleichen und um daraus Empfehlungen ableiten zu können, welche Datenbanklösung für einen Anwender am besten geeignet ist.
Die Schnittstelle zum Im- und Export von Datensätzen wird in den Datenbanklösungen CityServer3D und 3D City Database durch spezielle Tools realisiert. In Abbildung 12 ist ein Überblick zu dieser Schnittstelle dargestellt:

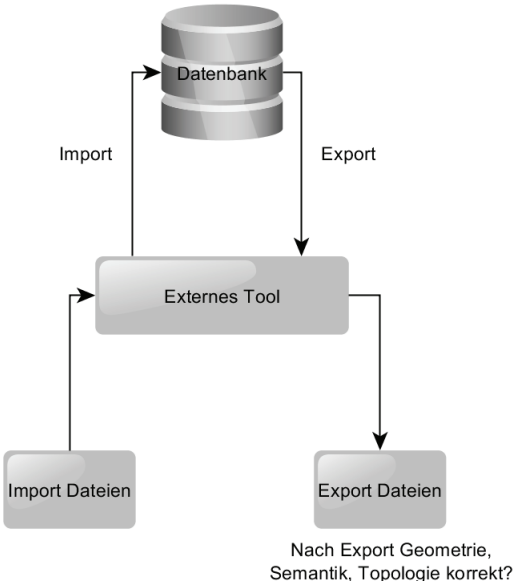


Abbildung 12: Schnittstelle Im- Export Tools

Der CityServer3D bietet das Admin Tool, ein umfangreiches Programm zum Im- und Export verschiedener Datenformate, auch Rasterdaten, in die Datenbank an. Das Tool dient der Visualisierung und Verwaltung von innerhalb sowie außerhalb der Datenbank befindlichen Daten. Mit dem Tool können zudem komplexe Regeln zur Bearbeitung von Datenbeständen definiert werden.
Die 3D City Database bietet als Schnittstelle zur Datenbank ein externes Open Source

Tool zum Im- und Export von CityGML Dateien an. Es ist Java basiert und besitzt sowohl ein Graphical User Interface (GUI) sowie ein Command Line Interface (CLI). Das Tool unterstützt den Im- und Export aller CityGML Dateien der Versionen 0.4.0 und 1.0.0. Als zusätzliches Export Dateiformat wird die Keyhole Markup Language (KML) angeboten. KML ist eine Auszeichnungssprache zur Beschreibung von Geodaten für Google Earth und Google Maps. Mit dem Tool können weiterhin zu importierende CityGML Dateien validiert werden. Zusätzlich kann aus zwei Objekten unterschiedlicher LoDs ein kombiniertes Objekt erzeugt werden, indem ein Objekt als Geometriegrundlage dient und mit den Eigenschaften des zweiten Objekts verknüpft wird.
Als zweites Tool zum Im- und Export von Orthophotos und Rasterdaten wird in [KKN12] auf ein ungetestetes Tool für die Vorgängerversion 1.0 der 3D City Database verwiesen. Die Funktionsfähigkeit dieses Programms mit der getesteten 3D City Database Version auf einer deutschen Oracle Umgebung wird in Abschnitt 5.3 näher beschrieben.
Mithilfe dieser Tools werden verschiedene Datensätze in die beiden Datenbanken importiert. Anschließend werden diese Daten aus der Datenbank exportiert und hinsichtlich Geometrie, Topologie und Semantik überprüft. Es wird erwartet, dass die exportierten Datensätze dieselben Eigenschaften wie die importierten Datensätze besitzen und keine Informationen durch den Datenbankim- und Export verloren gehen.

5.1. Getestete Datensätze

Die Auswahl der Datensätze erfolgt im Hinblick auf die Abdeckung der LoD Stufen 1-4, Texturen, verschiedene Hersteller und Anzahl der Objekte. Zu den getesteten Datensätzen gehören elf 3D Stadtmodelle im CityGML 1.0.0 Format. In Tabelle 3 sind diese mit ihrer Dateigröße und dem jeweiligen Detaillierungsgrad dargestellt.

Index	Beschreibung	Größe (KB)	Detaillierung
1	Kiel Altstadt	5.093	LoD 2
2	Kiel Blücherplatz	17.072	LoD 2
3	Kiel Brunswick	9.221	LoD 2
4	Berlin Beispieldaten mit Texturen	14.643	LoD 2
5	München Beispieldaten mit Texturen	23.244	LoD 3
6	Nürnberg mit Texturen	144.384	LoD 2
7	einzelnes Gebäude	20.377	LoD 4
8	Kiel komplett nach Fehlerbehebung	647.219	LoD 2
9	Erfurter Kreuz	5.978	LoD 1
10	München mit Landschaftsobjekten	8.600	LoD 3
11	Ingolstadt	4.100	LoD 2

Tabelle 3: Getestete Datensätze Im- und Export

3D Stadtmodelle können mit Orthophotos oder Rasterdaten unterlegt werden. Daher wird zusätzlich der Im- und Export eines georeferenzierten Orthophotos (350MB) in den Datenbanken getestet. Es gilt u.a., das laut [KKN12] ungetestete Tool für Rasterda-

ten und Orthophotos mit der installierten Version[2] der 3D City Database zu überprüfen. Die in der Tabelle 3 dargestellten Datensätze sowie das 350MB große Orthophoto werden in die Datenbanken importiert. Der Export erfolgt im CityGML, KML, Collaborative Design Activity (Collada) und im Geo Tagged Image File Format (GeoTIFF).

5.2. Im- / Export CityGML Datensätze

Da alle Datensätze im CityGML Standard erzeugt wurden, verlief der Import in beiden Datenbanklösungen fehlerfrei. Um Geometrie, Semantik und Topologie zu überprüfen, werden die Datensätze als CityGML und KML Dateien exportiert und anschließend visuell mithilfe des *tridicon CityDiscovererLight* der Firma GTA Geoinformatik GmbH und dem *LandXPlorer CityGMLViewer2009* der Firma Autdesk überprüft. Die Kontrolle der KML Dateien erfolgt mit *Google Earth* der Firma Google.

5.2.1. CityGML & KML Export

Der Export der Testdatensätze im CityServer3D erfolgt ohne Fehlerausschriften. Bei der visuellen Überprüfung der KML Dateien wird jedoch ein Fehler beobachtet. Es werden keine semantischen Eigenschaften (Wand- und Dachflächen) dargestellt, wie in Abbildung 13 erkennbar:

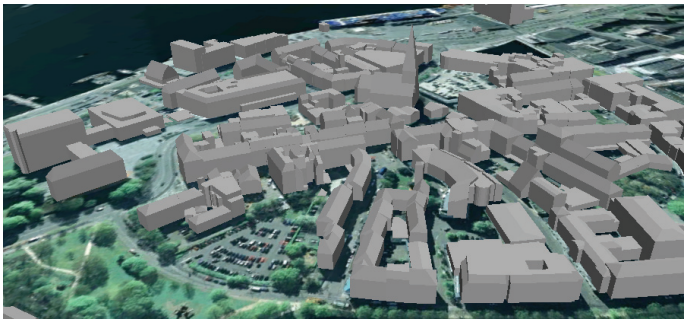


Abbildung 13: Kieler Altstadt KML Export CityServer3D fehlerhaft

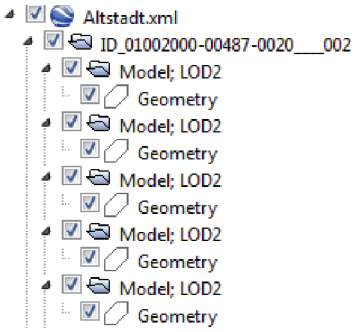


Abbildung 14: Kieler Altstadt KML Objektdefinition fehlerhaft

In den importierten und auch exportierten CityGML Dateien vom CityServer3D werden die Semantiken korrekt definiert. Die semantischen Eigenschaften müssten daher auch in den KML Dateien definiert sein. Stattdessen werden, wie in Abbildung 14 deutlich wird, lediglich Geometrien beschrieben.
In der 3D City Database liefert der KML Export bei den Datensätzen 1,2,3 und 11 teilweise *unknown geometry* und Oracle Fehlerausschriften. Diese Fehler können durch fehlerhafte Import-Datensätze (Geometrie falsch) oder fehlerhafte interne Funktionen entstehen.

[2] Version 2.0.5

Abbildung 15 zeigt den fehlerhaften KML Export der Kieler Altstadt:

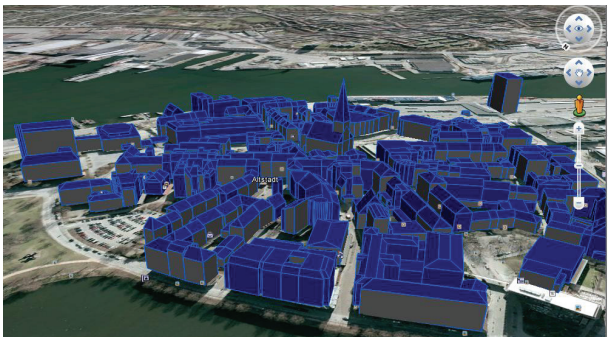


Abbildung 15: Kieler Altstadt KML Export fehlerhaft

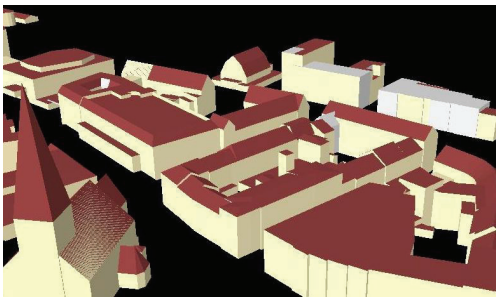


Abbildung 16: Kieler Altstadt CityGML fehlerhaft

Erkennbar sind einige Wandflächen (grau), die fälschlicher Weise als Dachflächen (blau) dargestellt werden. Um zu prüfen, ob der importierte Datensatz bereits Fehler besitzt, wird der zugehörige importierte CityGML Datensatz visuell mithilfe des LandXPlorers überprüft. Abbildung 16 zeigt den Datensatz 1 im CityGML Format.
Erkennbar ist ein verwandter Fehler wie beim KML Export (Abb. 15). Die Dachflächen werden teilweise nicht korrekt erkannt oder dargestellt. Die Darstellung erscheint, als sei eine zusätzliche Fläche über die ohnehin existierende Dachfläche modelliert. Diese zusätzliche Wandfläche ist in der Abbildung besonders gut auf dem Kirchendach erkennbar.
Dies lässt die Schlussfolgerung eines fehlerhaften importierten Datensatzes zu.
Ein 3D Stadtmodell in einem CityGML Datensatz ist generell durch eine beliebige Anzahl ineinander verschachtelter Knoten definiert. Ein einzelnes Gebäude ist beispielsweise ein Knoten und besitzt mehrere Unterknoten, welche die Eigenschaften des Objekts festlegen. Die sichtbaren Geometrien werden dabei in den Knoten *Solid* oder *boundedBy* festgelegt.
Bei der Untersuchung der Dateien wird eine doppelte Geometriezuweisung festgestellt. Die problematische Definition eines solchen Gebäudeobjekts via XLinks ist in Abbildung 17 dargestellt:

```
<bldg:lod2Solid>
    <gml:Solid>
        <gml:exterior>
            <gml:CompositeSurface>                                    Referenz auf*
                <gml:surfaceMember xlink:href="#ID_01002000-00952-0030____001_1"/>
                ...
</bldg:lod2Solid>
<bldg:boundedBy>
        ...
            <gml:surfaceMember>
                <gml:Polygon gml:id="ID_01002000-00952-0030____001_1">  *
                    <gml:exterior>
                        <gml:LinearRing>
                            <gml:pos>574231.442 6019936.663 11.263</gml:pos>
                            <gml:pos>574244.517 6019925.592 11.263</gml:pos>
```

Abbildung 17: XLink auf Objekt

Es wird deutlich, dass in den Datensätzen die Geometrien für Wand- und Dachflächen (*boundedBy*) gesetzt sind und anschließend mit XLinks nochmals als Körper (*Solids*) referenziert werden. Dadurch werden dem Objekt die Geometrien doppelt zugewiesen und es kommt zu der fehlerhaften Darstellung. Da mehrere Datensätze (für Kiel ca. 25) und mehrere hunderttausende Objekte korrigiert werden müssen, werden die fehlerhaften *Solid*-Knoten mithilfe eines eigenen Programmes aus den Dokumenten entfernt. Abbildung 18 zeigt die grundsätzliche Architektur des Programmes:

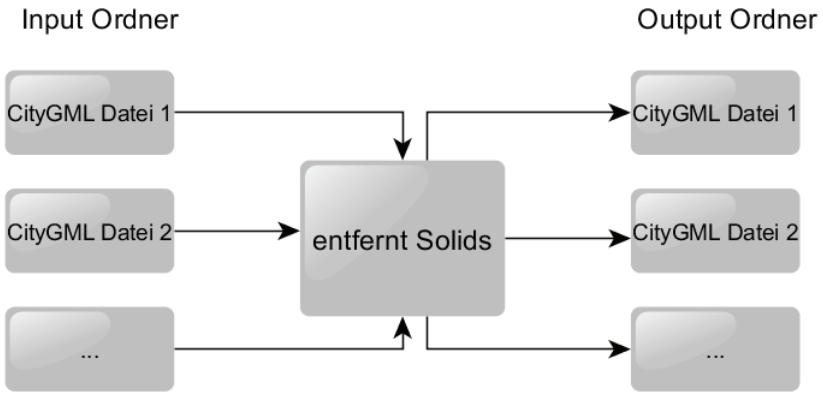


Abbildung 18: Programm zum Entfernen der Solid Objekte

Im CityServer3D muss dieses Programm nicht genutzt werden. Es erkennt diese falschen Referenzen und importiert diese nicht in die Datenbank. Durch einen Export der Datensätze werden die XLinks dadurch automatisch entfernt.
Der Export und die Visualisierung der fehlerhaften Datensätze 1,2,3 und 11 funktionieren nach dieser Fehlerbehebung durch das o.g. Programm in der 3D City Database nun problemlos. Erfolgreich getestet wurde dies mit dem Datensatz 8 „Kiel komplett“. Tabelle 4 gibt einen Überblick der entstandenen Fehler beider Datenbanken:

Datensatz	produzierter Fehler	Fehlerbeschreibung	Datenbank
1,2,3,11	unknown geometry, ORA-29913	Wand- und Dachflächen fehlerhaft	3D City Database
alle	keiner	KML Export Semantik fehlt	CityServer3D

Tabelle 4: Fehler CityGML & KML Export

5.2.2. XLinks Unterstützung

Um zu testen, ob die beiden Datenbanklösungen das XLinks Konzept unterstützen, wird ein eigenes Gebäude (aus 3 Gebäudeteilen) mit mehreren über XLinks referenzierte Flächen erstellt. Abbildung 19 stellt die gemeinsam genutzten Flächen (blau) dar:

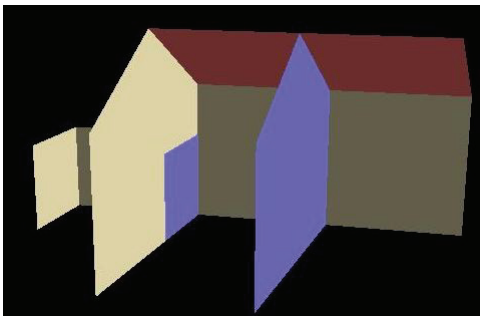


Abbildung 19: Eigenes Objekt zur Demonstration geteilter Flächen

Jede dieser markierten Flächen wird von zwei Gebäudeteilen genutzt. Die Koordinaten werden für ein Objekt festgelegt und vom zweiten über eine XLink Referenz genutzt. Da Geometrien z.T. sehr komplex sein können, ist die doppelte Definition nicht vorteilhaft. Ziel dieser Verlinkungen ist es daher, Speicherplatz in den CityGML Dateien zu sparen.

Um die XLinks Unterstützung zu prüfen, wird das Gebäude in beide Datenbanklösungen im- und exportiert.
Bei beiden Datenbanklösungen gehen nach dem CityGML Export keine Informationen verloren. Im CityServer3D befinden sich nach dem Export keine XLinks mehr in dem Dokument. Das Admin Tool löst die XLinks dennoch korrekt auf und speichert diese in der Datei.
Die 3D City Database hingegen importiert die referenzierten Geometrien in die Datenbank und markiert diese mit einem speziellen Attribut. Beim Export wird das Dokument wieder korrekt zusammen mit den XLink Definitionen ausgegeben. Die korrekte Nutzung der XLinks und genauere Arbeitsschritte sind in dem firmeninternen Dokument der GTA Geoinformatik [Kri12b] zusammengefasst.

5.3. Im- / Export eines Orthophotos

Orthophotos werden für einen realitätsnahen Eindruck des 3D Stadtmodellgebietes genutzt und müssen in einer Datenbank gespeichert und verwaltet werden können. In diesem Abschnitt wird der Im- und Export eines Orthophotos in den Datenbanklösungen CityServer3D und 3D City Database getestet.
Das Orthophoto lässt sich problemlos in den CityServer3D im- und exportieren.

Das Im- und Export Tool der 3D City Database kann einzelne Orthophotos / Rasterdaten importieren. Diese werden dann mithilfe einer gespeicherten Datenbankprozedur zu einem georeferenzierten Bild zusammengefügt und können anschließend mit dem Tool exportiert werden. Dieses Tool der 3D City Database ist nach ausgiebigen Tests mit der Version 2.05 unter einer deutschen Oracle Umgebung nicht funktionsfähig. Im folgenden Abschnitt werden die Fehler, Fehlerursachen sowie eine alternative Lösung zur Speicherung beschrieben. Ausführlichere Informationen dazu sind in dem firmeninternen Dokument der GTA Geoinformatik [Kri12a] nachzulesen. Arbeitsprozess 5 (Abbildung 20) verdeutlicht die Arbeitsschritte des Tests zum Im- und Export von Orthophotos in der 3D City Database:

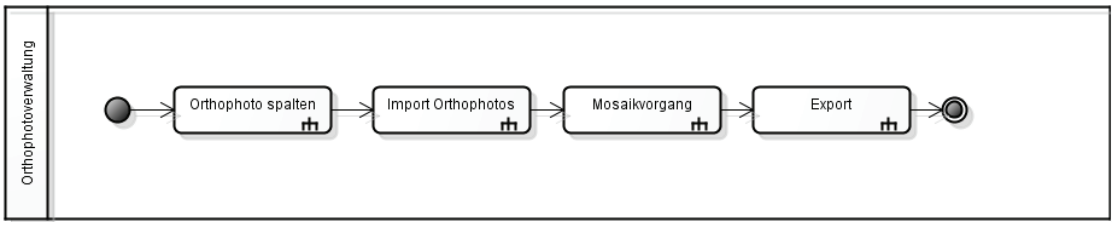


Abbildung 20: Orthophoto Im- und Export (Arbeitsprozess 5)

Um den Mosaikvorgang zu testen, müssen mehrere zusammengehörige Orthophotos importiert werden. Dazu wird das Orthophoto in vier gleich große Dateien geteilt. Zur Aufteilung wird eine in dem Tool mitgelieferte Funktion genutzt. Das Ergebnis sind vier TIFF Dateien mit zugehörigen .tfw Worlddateien, welche Koordinaten und Auflösung beinhalten. Bereits bei dieser Funktion sind Unstimmigkeiten in Form von fehlenden Zeilenumbrüchen in den Worlddateien vorhanden. Diese müssen manuell gesetzt werden, um die Dateien erfolgreich mit dem Tool importieren zu können. Das Tool meldet anschließend keine Fehler beim Import. Um den Mosaikvorgang zu starten muss die in der Datenbank gespeicherte Prozedur mittels *SQLPlus* (Kommandozeilen Interface zur Oracle Datenbank) aufgerufen werden:
Execute mosaicOrthophotosInitial (<name>, <type>, <origin>, <LOD>);
Die Mosaikfunktion soll diese vier Bilder zu einem georefenzierten Abbild zusammenfassen. Dieser Funktionsaufruf erzeugt jedoch ein Fehler, wie in Tabelle 5 unter Nr. 1 aufgeführt wird.

Nr.	Fehler	Beschreibung
1	ORA-06531	Nicht initialisierte Zusammenstellung referenziert
2	ORA-13419	cannot perform mosaick operation on specified column
3	JavaSQLException	No GeoRaster object exists at specified rasterid

Tabelle 5: Fehler Orthophoto Im- und Export

Die Ausschrift deutet auf eine Objektinitialisierung ohne Konstruktor hin. Um die Fehlerursache zu ermitteln, wird der Datenbankimport kontrolliert. Mithilfe der *sdo_geor.getSRS* Funktion werden Informationen zur räumlichen Referenzierung des in der Datenbank gespeicherten Bildobjekts ermittelt. Ergebnis ist die Erkenntnis, dass einige Einträge als *NULL* gekennzeichnet sind, welche von dem Importer Programm in der Datenbank gesetzt sein müssten. Die fehlenden Einträge werden aufgrund dessen manuell mithilfe der Funktion *sdo_geor_srs* gesetzt. Mit der erwähnten Abfrage der räumlichen Referenz ist nun erkennbar, dass einige Einträge neu gesetzt und einige vorher gesetzte Einträge *NULL* gesetzt sind. Dafür ist ein offizieller Fehler in Oracle Spatial bei räumlichen Operationen (Mosaik-Funktion) in der deutschen Oracle Umgebung verantwortlich [Xie09]. Die Datenbank wird auf die englische Umgebung konfiguriert und die *MosaicOrthophotosInitial* Prozedur nochmals aufgerufen. Daraus resultiert der in Tabelle 5 benannte nicht behebbare Fehler Nr. 2.
Laut dieser Fehlermeldung ist die für Orthophotos spezifizierte Tabelle nicht für die Mosaik-Funktion geeignet. Die Datenbanküberprüfung zeigt jedoch, dass die angegebene Tabelle laut [Ora03] für den Datentyp *SDO_Raster* geeignet ist. Daraufhin wird eine eigene Mosaik- und Pyramidenfunktion in der lt. Fehler Nr. 2 nicht dafür geeigneten Tabelle ausgeführt. Ergebnis ist ein erfolgreich zusammengefügtes *GeoRaster* Objekt in dieser Tabelle. Die Export Funktion des Tools ist, wie Fehler Nr. 3 in Tabelle 5 erkennbar, ebenso nicht funktionsfähig. Laut dieser Fehlermeldung existiert das manuell erzeugte *GeoRaster* Objekt in der angegeben Tabelle nicht.

Um das fehlerhafte Tool zu ersetzen und Orthophotos in der 3D City Database speichern zu können, werden prototypisch eigene Import-, Mosaik- und Exportprozeduren entwickelt. In Abbildung 21 ist der Ablauf dieser Prozesse dargestellt:

Abbildung 21: Import-, Mosaik- und Exportprozedur

Zunächst wird pro Orthophoto ein leeres *SDO_Raster* Objekt in der laut Fehler Nr. 2 nicht dafür geeigneten Tabelle erzeugt. In diese Objekte werden dann die Orthophotos importiert. Durch den Importvorgang werden auch alle Daten zur räumlichen Referenzierung erfolgreich gesetzt. Durch die eigene Mosaikfunktion wird ein Cityobject angelegt

und alle Orthophotos zusammengefügt. Nach Abschluss des Prozesses kann das neu zusammengefügte *SDO_GeoRaster* Objekt mittels einer eigenen Exportfunktion exportiert werden. Die Oracle Datenbank hat eine Exportgrößenbeschränkung von 67 MB. Demzufolge muss die Exportfunktion einen Ausschnitt umfassen. Alle Vorgänge wurden mit Oracle Spatial spezifischen Befehlen als SQL Prozeduren durchgeführt. Abbildung 22 zeigt das erfolgreich zusammengefügte Exportergebnis (Ausschnitt).

Abbildung 22: Eigener Export GeoRaster Objekt

5.4. Performance Tests

Dieses Kapitel stellt die Im- und Exportgeschwindigkeiten von einigen ausgewählten CityGML Dateien mit den mitgelieferten Tools des CityServer3D und der 3D City Database gegenüber. In diesen Tests wird sowohl der Client als auch der Server auf demselben Rechner betrieben. Die Ausgangssituation am lokalen Rechner wird durch eine 32Bit Windows XP Umgebung mit 2 GB RAM und einem Intel Core2Duo 1,86 GHz realisiert. Es wird die CityServer3D Version 2.5.1 mit der MySQL Datenbankversion 5.5 verwendet. Die 3D City Database Version 2.0.5 wird mit der Oracle Spatial Datenbankversion 11.2.0.1.0 genutzt.
Der Geschwindigkeitsvergleich wird durchgeführt, um die Effektivität der Tools miteinander zu vergleichen. Es wird erwartet, dass die Geschwindigkeit von der Größe der Dateien abhängt und der Export- schneller als der Importvorgang ist, da Geodatenbanken effektive interne Funktionen zum Export bereitstellen (s. Kap. 4.2) und der Zugriff auf ein Datenbanksystem schneller ist, als das Analysieren von Dateien in einem Filesystem. Weiterhin wird erwartet, dass das Admin Tool vom CityServer3D aufgrund der verhältnismäßig schwachen Hardware ein etwas schlechteres Ergebnis erzielen wird als das Im- und Exporter Tool der 3D City Database. Der Hauptgrund liegt darin, dass das Admin Tool die zu importierenden Daten zunächst in den Zwischenspeicher lädt, sie dort visualisiert, woraufhin sie dann bearbeitet werden können. Das Im- und Exporter Tool der 3D City Database hingegen bietet eine reine Importprozedur an, welche die Daten aus den Dateien liest und anschließend direkt in die Datenbank speichert.
Als Testdatensätze dienen 6 Datensätze. Tabelle 6 zeigt die Eigenschaften der Datensätze auf:

Datensatz	Beschreibung	Größe (MB)	Objekte	LoD
1	kleiner Datensatz Kiel	5	290	2
2	mittelgroßer Datensatz	250	19.360	2
B1	großer Datensatz, viele Objekte	7.895	1.055.951	1
B2	großer Datensatz, wenige Objekte	6.379	510.927	2&3
3	kleiner Datensatz, viele Objekte	18	3.535	1
4	kleiner Datensatz, wenige Objekte	15	233	2&3

Tabelle 6: Performance Test Datensätze

Die Datensätze 1 und 2 wurden aufgrund der Dateigrößen ausgewählt. Es handelt sich dabei um einen kleinen 5MB großen Datensatz mit 290 Gebäudeobjekten im LoD 2 und einen mittelgroßen 250MB großen Datensatz mit 19.360 Objekten, ebenfalls im LoD 2. Anhand dieser Datensätze wird lokal geprüft, welches Tool die Daten schneller verarbeiten kann.
Die Datensätze B1 und B2 können lokal nicht getestet werden. Es handelt sich um Testdatensätze der IGG Berlin, welche mit der 3D City Database getestet wurden. Es sind 3D Stadtmodelle von Köln und Berlin. Durchgeführt wurde ein veröffentlichter Performance Test mit zwei großen Datensätzen (7.895MB und 6.379MB) in unterschiedlichen Detaillierungsgraden (1 und 2&3). Der Test wurde auf einem Intel Xeon QuadCore auf

einer 64 Bit Windows 7 Umgebung mit 64GB RAM ausgeführt. Der Server läuft auf zwei Intel Xeon QuadCore Prozessoren auf einer Linux Umgebung mit 32 GB RAM und der Oracle 10.2.0.4 Version. Das Hauptmerkmal dieser Datensätze ist, dass sich die Dateigröße des zweiten Datensatzes (B2) um ca. 20% von dem ersten Datensatz (B1) unterscheidet, er aber lediglich die Hälfte an Objekten besitzt. Da dieser Datensatz lediglich in der 3D City Database getestet wurde, werden zwei ähnliche Datensätze für den CityServer3D gewählt: die Datensätze 3 und 4. Diese werden im CityServer3D getestet und weisen trotz ihrer geringen Dateigröße strukturell dieselben Eigenschaften wie die Datensätze B1 und B2 auf.
Wie bereits erwähnt, werden die Datensätze 1 und 2 lokal getestet und gegenübergestellt. Die Ergebnisse der Datensätze B1 und B2 sind aus [Her11] übernommen und nur für die 3D City Database gültig. Die Datensätze 3 und 4 werden lokal für den CityServer3D getestet, um dessen Ergebnisse mit B1 und B2 vergleichen zu können.
Die Angabe der folgenden Performance Ergebnisse entspricht dem arithmetischen Mittel aus drei Messungen. Tabelle 7 zeigt die Ergebnisse der Datensätze 1 und 2 im CityServer3D und der 3D City Database:

		CityServer3D	3D City Database
Import	Datensatz 1	0 Minuten 46 Sekunden	0 Minuten 18 Sekunden
	Datensatz 2	48 Minuten 36 Sekunden	37 Minuten 37 Sekunden
Export	Datensatz 1	0 Minuten 6 Sekunden	0 Minuten 9 Sekunden
	Datensatz 2	2 Minuten 55 Sekunden	2 Minuten 30 Sekunden

Tabelle 7: Performance Test Datensätze 1 & 2

Abbildung 23 zeigt das Ergebnis der Datensätze 1 und 2 als Diagramm:

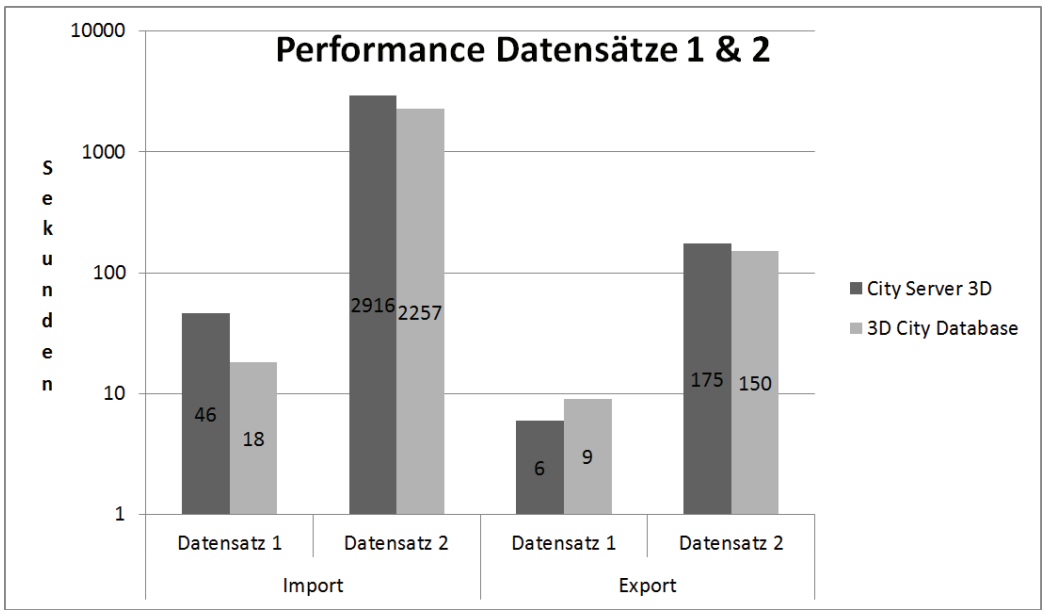


Abbildung 23: Performance Test Datensätze 1 & 2

Es wird deutlich, dass die Importprozeduren trotz relativ kleiner Datenmengen viel Zeit benötigen. Wie erwartet ist der Importvorgang im CityServer3D gegenüber dem der 3D City Database langsamer. Dies hängt, wie bereits erwähnt, mit der Funktionsweise des Admin Tools zusammen: Die Daten werden zunächst in den Zwischenspeicher geladen und visualisiert. Es können Manipulationen der Daten vorgenommen und die zu importierenden Objekte ausgewählt werden. Die Visualisierung mit der verwendeten Hardware benötigt sehr viel Zeit und das Ergebnis fällt dadurch schlechter aus.
Es bestätigt sich dementsprechend die Vermutung, dass der Importvorgang der 3D City Database aufgrund der fehlenden Visualisierung und Manipulationsmöglichkeiten effektiver die Speicherung in die Datenbank vornimmt.
Weiterhin wird ersichtlich, dass beide Datensätze in beiden Datenbanklösungen um ein Vielfaches schneller exportiert werden, als sie importiert werden. Die Hauptursache liegt darin, dass die Dateien beim Import zunächst analysiert und in ihre Bestandteile aufgelöst werden, ehe die Daten in der Datenbank gespeichert werden können. Exportvorgänge werden von den Datenbanklösungen effektiv durch interne Prozeduren unterstützt und aufgrund der effektiven Verwaltung der Daten in den Datenbanken schnell durchgeführt.

Der Geschwindigkeitstest der Datensätze B1 und B2 ist, wie bereits erwähnt, aus [Her11] entnommen und zeigt die Geschwindigkeit des CityGML Im- und Exports der 3D City Database unter sehr guten hardwaretechnischen Bedingungen. Tabelle 24 zeigt die Ergebnisse der Datensätze B1 und B2:

3D City Database	Import	Export
Datensatz B1	37 Minuten	12 Minuten
Datensatz B2	20 Minuten	13 Minuten

Tabelle 8: Performance Test Datensätze B1 & B2

Abbildung 24 zeigt das Ergebnis der Datensätze B1 und B2 als Diagramm:

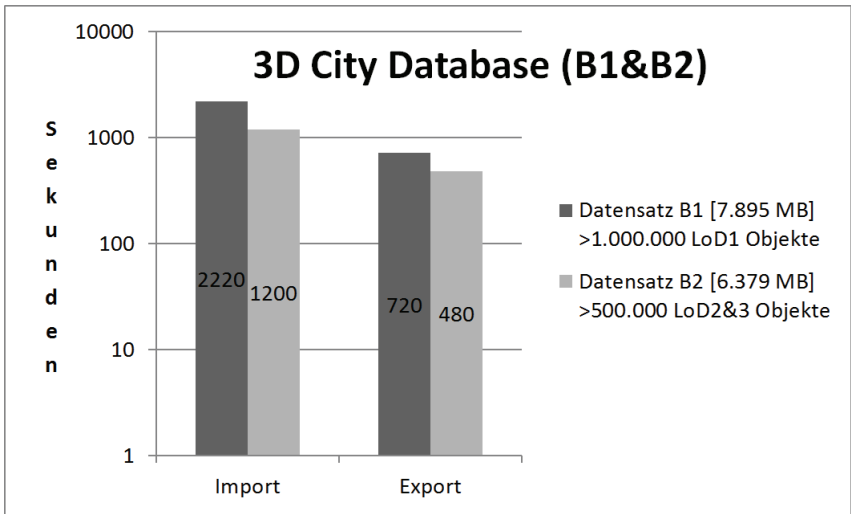


Abbildung 24: Performance Test Datensätze B1 & B2

Der Vergleich der beiden Datensätze lässt den Schluss zu, dass bei sehr guter Hardwareausstattung der Im- und Export in der 3D City Database größtenteils von der Anzahl der Objekte abhängig ist. Demzufolge hängt die Performance auch indirekt vom LoD ab, da bei gleicher Dateigröße in einem höheren Detaillierungsgrad weniger Objekte abgebildet werden können. Erkennbar wird dies besonders durch die Importzeiten der Datensätze B1 und B2. Datensatz B1 ist mit seinen 7.895 MB nur geringfügig größer als Datensatz B2. Dennoch besitzt er doppelt so viele Objekte, was durch die unterschiedlichen Detaillierungsgrade erklärbar ist, und benötigt für den Import fast doppelt soviel Zeit. Dass der Export der Datensätze schneller als der Import ist, wird auch bei diesem Performancetest ersichtlich.
Um zu überprüfen, wie sich der CityServer3D bei der Gegenüberstellung von vielen und wenigen Objekten bei ungefähr gleich großen Dateien verhält, werden die Datensätze 3 und 4 mit dem Admin Tool des CityServer3D im- und exportiert. Die Ergebnisse sind in der Tabelle 9 dargestellt:

CityServer3D	Import (Sekunden)	Export (Sekunden)
Datensatz 3 [18 MB]	4 Minuten 21 Sekunden	0 Minuten 22 Sekunden
Datensatz 4 [15 MB]	3 Minuten 11 Sekunden	0 Minuten 19 Sekunden

Tabelle 9: Performance Test Datensätze 3 & 4

Abbildung 25 zeigt das Ergebnis der Datensätze 3 und 4 als Diagramm:

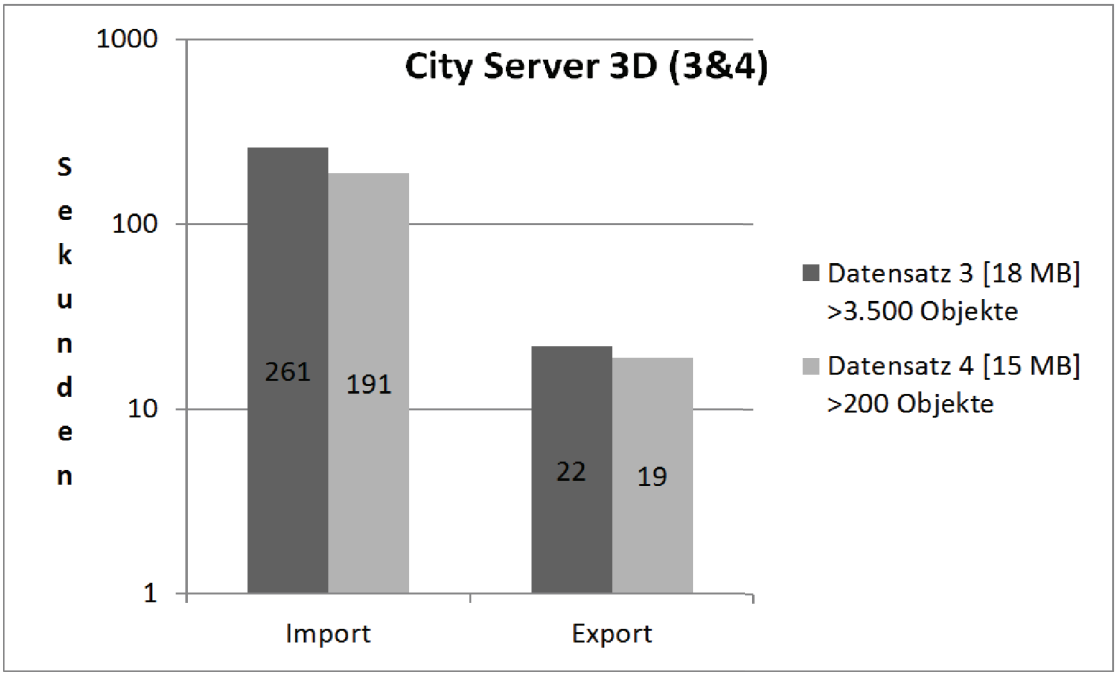


Abbildung 25: Performance Test Datensätze 3 & 4

Auch beim CityServer3D ist ein ähnliches Ergebnis wie bei den Datensätzen B1 und B2 in der 3D City Database zu beobachten. Datensatz 3 benötigt trotz geringfügig größerer Dateigröße etwa 35% mehr Importzeit als der Datensatz 4. Dass die Unterschiede nicht ganz so signifikant wie bei den Datensätzen B1 und B2 ausfallen, lässt sich mit der bereits erwähnten schwächeren Hardware und der insgesamt geringeren Dateigröße erklären. Dennoch wird auch beim CityServer3D ebenso wie bei der 3D City Database eine starke Abhängigkeit von der Objektanzahl und damit auch indirekt vom LoD festgestellt.

5.5. Feature Vergleich

Dieser Abschnitt stellt die Eigenschaften des CityServer3D und der 3D City Database gegenüber. Die vorliegende Tabelle ist an die Feature Matrix aus [Fra12] angelehnt, wird mit den Eigenschaften aus [KKN12] ergänzt und mit den Erkenntnissen der bisherigen Ergebnisse korrigiert. Tabelle 10 stellt die Eigenschaften der beiden Datenbanklösungen gegenüber:

		CityServer3D	3D City Database
Unterstützte Plattformen	Windows	✓	✓
	Unix / Linux	✓	✓
	Mac OS X	✓	-
Unterstützte Datenbanken	MySQL	✓	-
	MongoDB	✓	-
	PostgreSQL	✓	✓
	Oracle	✓	✓
	Microsoft SQL Server	✓	-
Architektur Eigenschaften	gr. Datenvolumen (>1TB)	✓	✓
	viele Clients (>50)	✓	✓
	verteilte Installation	✓	✓
Bereitgestellte Dienste	OGC W3DS	✓	(-)
	OGC WPVS	✓	(-)
	OGC WFS	✓	(-)
Importformate	CityGML	✓	✓
	GeoTiff	✓	(-)
	Shapefiles, VRML u.A.	✓	-
	Eigene Dateiformate	-	✓
Exportformate	CityGML	✓	✓
	KML/KMZ	✓	✓
	Shapefiles, VRML u.A.	✓	-
	Eigene Dateiformate	-	✓
Qualitäts-sicherung	Topologie	✓	✓
	Semantik	✓	✓
	Geometrie	✓	✓
weitere Eigenschaften	Versionierung	✓	✓
	integr. Visualisierung	✓	-
	Datenbearbeitung möglich	✓	(-)
	filterbarer Import	✓	✓
	integr. Regelmanagement	✓	-
	Objektzusammenführung	✓	✓
	CRS Konvertierung	✓	✓
	Nutzer- / Rollenmanagement	✓	✓
	Datenverschlüsselung	✓	✓
	Open Source	-	✓

Tabelle 10: Feature Vergleich nach [Fra12]

Aus der Tabelle ist ersichtlich, dass der CityServer3D eine bessere Kompatibilität und einen größeren Funktionsumfang bietet als die 3D City Database. Während die 3D City Database lediglich mit einer Oracle Datenbank und aktuell[3] auch mit PostgreSQL kompatibel ist, kann der CityServer3D aufgrund des Metamodells auf verschiedenen, auch dokumentenbasierten Datenbanken installiert werden. Beide Datenbanklösungen unterstützen große Datenvolumen und sind als Client-Server Architekturen konzipiert.

Als Im- und Exportformate bietet der CityServer3D eine breite Palette zusätzlicher Dateiformate an, während die 3D City Database lediglich CityGML und KML Dateien unterstützt. Da die 3D City Database eine komplette Umsetzung des CityGML Standards ist, werden zusätzliche Dateiformate nur in Form eigener Dateiformate unterstützt, welche der CityServer3D nicht unterstützt. Aus CityGML Dateien können spezielle Datenformate abgeleitet werden, wodurch Speziallösungen in der 3D City Database vermieden werden.

Bei den Anforderungen an Geometrie, Semantik und Topologie in CityGML Dateien können beide Datenbanklösungen überzeugen. Sowohl der CityServer3D als auch die 3D City Database können XLinks verarbeiten. Beide Datenbanklösungen unterstützen die Versionierung der Daten, können Objekte zusammenfügen, Koordinatentransformationen durchführen und bieten neben der Datenverschlüsselung auch ein Nutzer- und Rollenmanagement an. Das CityServer3D Paket bietet eine integrierte Visualisierung und Bearbeitung der Daten an, während in der 3D City Database die CityGML Dateien mit externen Programmen manipuliert bzw. visualisiert werden müssen. Ein weiterer Vorteil des CityServer3D ist das integrierte regelbasierte System zur Verwaltung der Datenbank, mit dem der in Abschnitt 6.1 erläuterte Fortführungsprozess durchgeführt wird.

[3] seit 19.07.2012

6. Fortführung

Wenn ein Gebiet mit einem höheren Detaillierungsgrad ausgestattet werden soll, oder der vorhandene Datenbestand veraltet ist, muss eine Neuerfassung durchgeführt werden. Dadurch entstehen neue Datenbestände, welche im System die alten ersetzen müssen. Dies entspricht dem in der Analyse geschilderten Arbeitsprozess „Ersetzen alter Datenbestände". Eine solche Aktualisierung soll in dieser Arbeit programmiertechnisch mit dem CityServer3D und der 3D City Database umgesetzt werden.
Als Testdatensätze werden zwei von der Firma GTA Geoinformatik GmbH bereitgestellte Datensätze genutzt. Es handelt sich dabei um ein nicht texturiertes 3D Stadtmodell (Datensatz 1) und einen veredelten Stadtteil des ersten Datensatzes (Datensatz 2). Ziel ist es, die Objekte gleicher ID des ersten Datensatzes mit denen des zweiten Datensatzes auszutauschen und demzufolge ein kombiniertes Stadtmodell zu erzeugen.

6.1. CityServer3D

Das Admin Tool des CityServer3D bietet eine umfangreiche, integrierte, regelbasierte Sprache zur Bearbeitung von in der Datenbank befindlichen Daten an. Es handelt sich dabei um Drools Expert, eine von der JBoss Inc. entwickelte deklarative und regelbasierte Programmierumgebung. Es ist ein Open Source System und Bestandteil eines Business-Rule-Management-Systems. Das Ziel eines solchen Systems ist die Trennung der Geschäftslogik von der Programmierlogik [JBo12]. Dadurch können auf Grundlage der internen Objekte Arbeitsprozesse umgesetzt werden, ohne den Programmcode ändern zu müssen.
Konkret bedeutet dies, dass zwei Codepassagen mithilfe der internen Java Objekte geschrieben werden müssen: eine Regel- und eine Konsequenz- Anweisung. Die Regel-Anweisung beinhaltet die Bedingung, unter welchen Aspekten Daten ausgewählt werden sollen. Die Konsequenz Anweisung behandelt die ausgewählten Daten. Mittels dieser regelbasierten Sprache wird im CityServer3D ein simpler Fortführungsprozess umgesetzt.

Abbildung 26 zeigt die Architektur des Fortführungsprozesses:

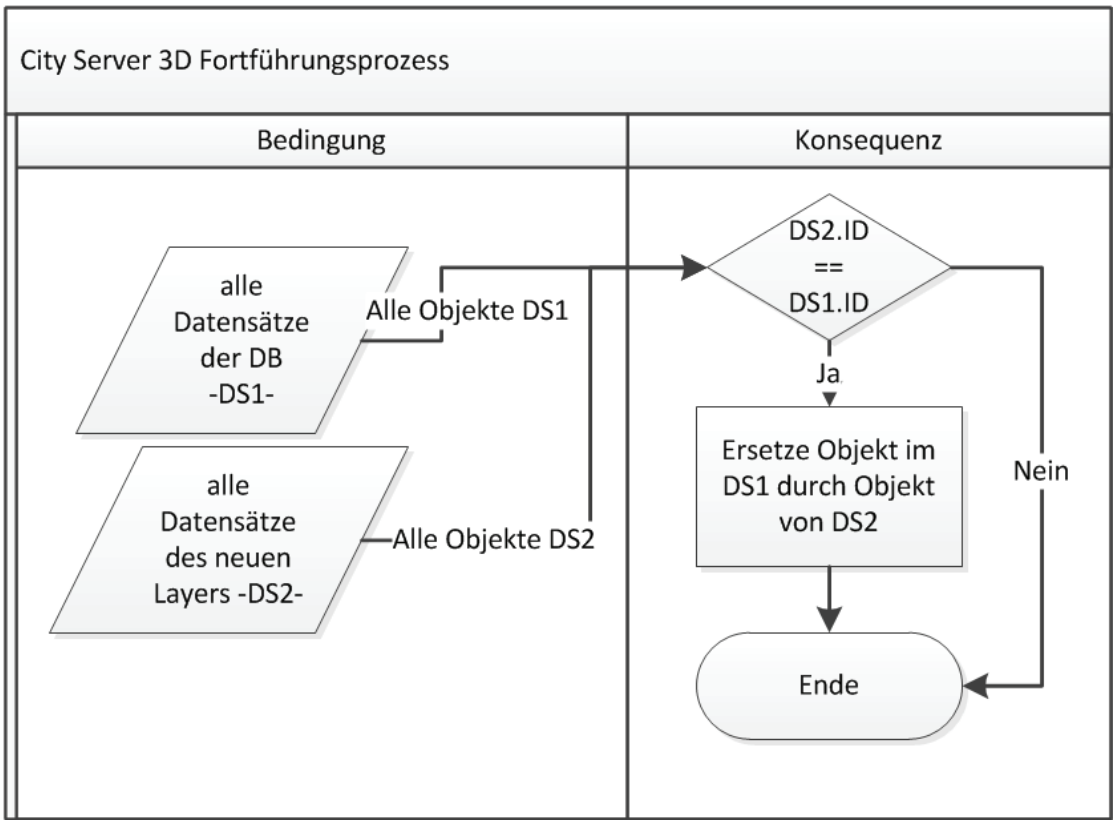


Abbildung 26: Fortführungsprozess CityServer3D

In der Bedingungsanweisung werden zwei Layer selektiert: das in der Datenbank befindliche Modell und das neue Fortführungsmodell. Von beiden Layern werden jeweils alle Stammobjekte ausgewählt. Damit sind alle Objekte gemeint, welche direkt dem Layer zugeordnet werden. Unterobjekte, wie z.B. Gebäudeteile werden mit den Stammobjekten identifiziert. Die zugehörige Regel wird hier aufgezeigt:

Bedingung:
stadtmodell : Layer (name=="Stadtmodell.xml") from entry-point world
fs : Feature (parent==stadtmodell) from entry-point world
fortfuehrung: Layer (name=="Fortfuehrung.xml") from entry-point world
ff : Feature (parent==fortfuehrung) from entry-point world

In der Konsequenzanweisung werden zunächst die IDs der Objekte beider Datensätze miteinander verglichen. Ist dies der Fall, wird das alte Objekt mit dem Neuen überschrieben und alte Objekte gelöscht. An dieser Stelle kann eine Versionierung durch Auslagern der alten Objekte in ein anderes Layer erfolgen. Der Prozess wird durch die regelbasierte Sprache automatisch für alle Objekte durchgeführt. Anschließend ist die Fortführung

abgeschlossen. Die verwendeten Konsequenzanweisungen werden hier zitiert:

```
Konsequenz:
if(ff.name==fs.name){
newf = new Feature();
newf = ff;
newf.children = ff.children;
Metadata.copy(ff,newf);
newf.parent=stadtmodell;
stadtmodell.children.remove(fs);
scene.delete(fs);
scene.delete(ff);
scene.add(newf);
stadtmodell.addChild(newf);
}
```

Erkennbar ist, dass die Umsetzung eines simplen Fortführungsprozesses mithilfe der regelbasierten Sprache ohne hohen Aufwand umgesetzt werden kann.
Zum Funktionstest wird der Datensatz 1 in die Datenbank importiert und der zweite Datensatz als zusätzliches Layer in den Zwischenspeicher geladen. Abbildung 27 zeigt die überlappten Layer vor dem Fortführungsprozess im Admin Tool des CityServer3D:

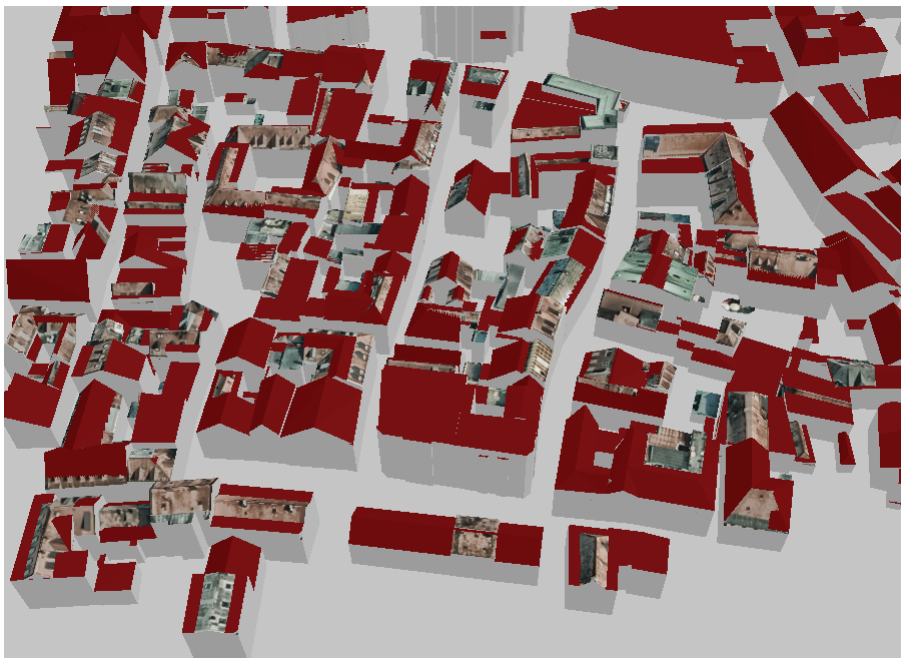


Abbildung 27: Überlappte Datensätze 1 & 2 CityServer3D

Es ist erkennbar, dass Geometrieveränderungen stattgefunden haben und die neuen Ob-

jekte Dachtexturen besitzen. Weiterhin ist ersichtlich, dass nicht alle Gebäude im Stadtmodell zu erneuern sind. Durch Ausführen der oben beschriebenen Regel wird der Fortführungsprozess vollzogen. In Datensatz 1 werden die Objekte mit identischer ID im Datensatz 2 ausgetauscht. Das visualisierte Ergebnis ist in Abbildung 28 dargestellt:

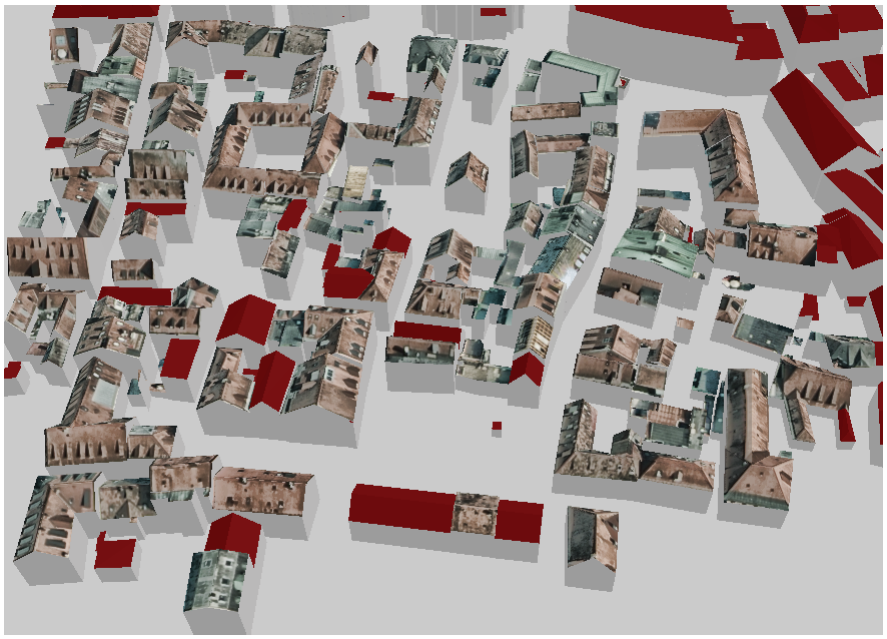


Abbildung 28: Fortführung CityServer3D

Das Layer des zweiten Datensatzes (Fortführungsdatensatz) wird aus der Ansicht entfernt, ehe die aktuelle Szene in der Datenbank gespeichert wird. Die Überprüfung zeigt, dass Objekte mit identischen IDs komplett ersetzt wurden. Es haben sowohl Geometrie- als auch Semantikveränderungen stattgefunden. Der Fortführungsprozess ist damit erfolgreich abgeschlossen.

6.2. 3D City Database

In der 3D City Database werden Fortführungsprozesse für Gebäudeobjekte bisher nicht unterstützt. Um dennoch einen solchen Arbeitsprozess abzubilden, muss ein neues Tool entwickelt oder das Open Source Tool zum Im- und Export von CityGML Dateien der 3D City Database erweitert werden. In dieser Arbeit wird ein neues Tool entwickelt, da zu Beginn dieser Arbeit noch keine Schnittstelle für Plugins in dem Im- und Export Tool existierte und die Anpassung des fremden Tools mehr Zeit benötigen würde, als ein neues Programm zu entwickeln. Im folgenden Abschnitt werden die Anforderungen an das Tool herausgearbeitet, die Umsetzung beschrieben und ein Funktionstest vorgenommen.

6.2.1. Konzeption

Die 3D City Database ist auf einem objektrelationalen Datenbanksystem aufgesetzt. Das Tool muss eine Verbindung zu der Datenbank aufbauen und die Daten manipulieren können. Demzufolge muss für einen Fortführungsprozess für Gebäudeobjekte das Datenbankmodell verstanden werden. Dazu gehören vor allem die in dem Kapitel 4.4 aufgeführten ER Modelle (Abb. 8, Abb. 9 und Abb. 10). Die Identifikation der Gebäudeobjekte erfolgt über eindeutige GMLIDs. Das Tool soll diese aus einer externen Textdatei oder aus dem Fortführungsdatensatz automatisch ermitteln. Der Löschvorgang in der Datenbank erfolgt auf Grundlage der o.g. ER Modelle. Um den fehlerfreien Import durchzuführen, kann ebenso eine eigene Importprozedur erstellt werden. Da die Importfunktion allerdings bereits durch das Im- und Exporter Tool der 3D City Database abgedeckt ist, wird dieser Vorgang über die CLI Schnittstelle des externen Tools ausgeführt. Um die CLI Schnittstelle zu nutzen wird eine XML Konfigurationsdatei benötigt, welche die Verbindungsinformationen, den Importmodus und falls erforderlich, die GMLIDs der zu importierenden Objekte beinhaltet. Die Konfigurationsdatei stammt von dem Im- und Exporter Tool der 3D City Database und wird durch das Tool manipuliert. Weiterhin benötigt die Schnittstelle den Pfad zu den zu importierenden Dateien. Abgerundet wird das Programm durch eine GUI als Benutzerschnittstelle, in der u.a. Verbindungsinformationen in der o.g. Konfigurationsdatei gesetzt und gelöscht, sowie Funktionen ausgewählt und Pfade gesetzt werden können.

Die Auswahl der Programmiersprache ist an die Anforderungen des Tools geknüpft. Demzufolge muss die Programmiersprache effektive Funktionen zur Bearbeitung von XML Dateien und zur Manipulation der Oracle Datenbank zur Verfügung stellen. Es müssen Prozesse gestartet werden können, um die CLI Schnittstelle nutzen zu können. Weiterhin sollten die Vorteile einer objektorientierten Programmiersprache, wie Vererbung und Kapselung sowie die Definition von Fehlverhalten ausgenutzt werden können. Die Auswahl ist daher auf die bekannten Programmiersprachen Java, C++ und C# begrenzt. Für das Programm wird die Programmiersprache C# genutzt, da dies die Standardprogrammiersprache der Firma GTA Geoinformatik GmbH ist, sie alle Anforderungen erfüllt, GUIs schnell und effektiv erzeugt werden können und sie eng mit der im Studium behandelten Sprache Java verwandt ist.

Mit dem Tool werden drei Anwendungsfälle abgedeckt:
Nur Einfügen: Mithilfe des im CLI Modus gestarteten Im- und Exporter Tool der 3D City Database, einer GMLID Liste und einem Pfad zu den Importdateien können Objekte problemlos in die Datenbank über das Tool importiert werden.
Nur Löschen: Eine Liste mit GMLIDs wird benötigt. Anschließend werden mittels SQL Datenbankabfragen gestellt um die zum jeweiligen Objekt gehörenden IDs zu ermitteln. Daraufhin werden die Entitäten, sofern keine weiteren Abhängigkeiten zu anderen Objekten mehr bestehen, aus der Datenbank gelöscht.
Fortführungsprozess: Dies stellt eine Kombination der beiden Anwendungsfälle dar. Es wird dabei eine GMLID Liste an das Löschpaket übergeben und eine Liste der tat-

sächlich gelöschten Objekte zurückgegeben. Da die Liste auch Fehler enthalten kann, wird durch die Rückgabeliste diese Fehlermöglichkeit behoben. Diese GMLID Liste wird anschließend an die CLI Schnittstelle des Im- und Exporter Tools der 3D City Database übergeben. Nach Beendigung des Importprozesses ist der Fortführungsprozess abgeschlossen.

6.2.2. Entwurf

Mithilfe der herausgearbeiteten Anforderungen und Lösungsideen wird nun die Entwicklungs- und Systemarchitektur festgelegt. Die Entwicklung erfolgt in einzelnen Paketen. Diese können unabhängig voneinander entwickelt werden. Durch Festlegen notwendiger Parameter können die einzelnen Pakete als zusätzliche Schnittstellen behandelt werden und am Projektende zusammengefügt und mit einem GUI versehen werden. Abbildung 29 zeigt die grobe Systemarchitektur des zu entwickelnden Tools. Dargestellt sind die geplanten Pakete, Schnittstellen und deren Verbindungen:

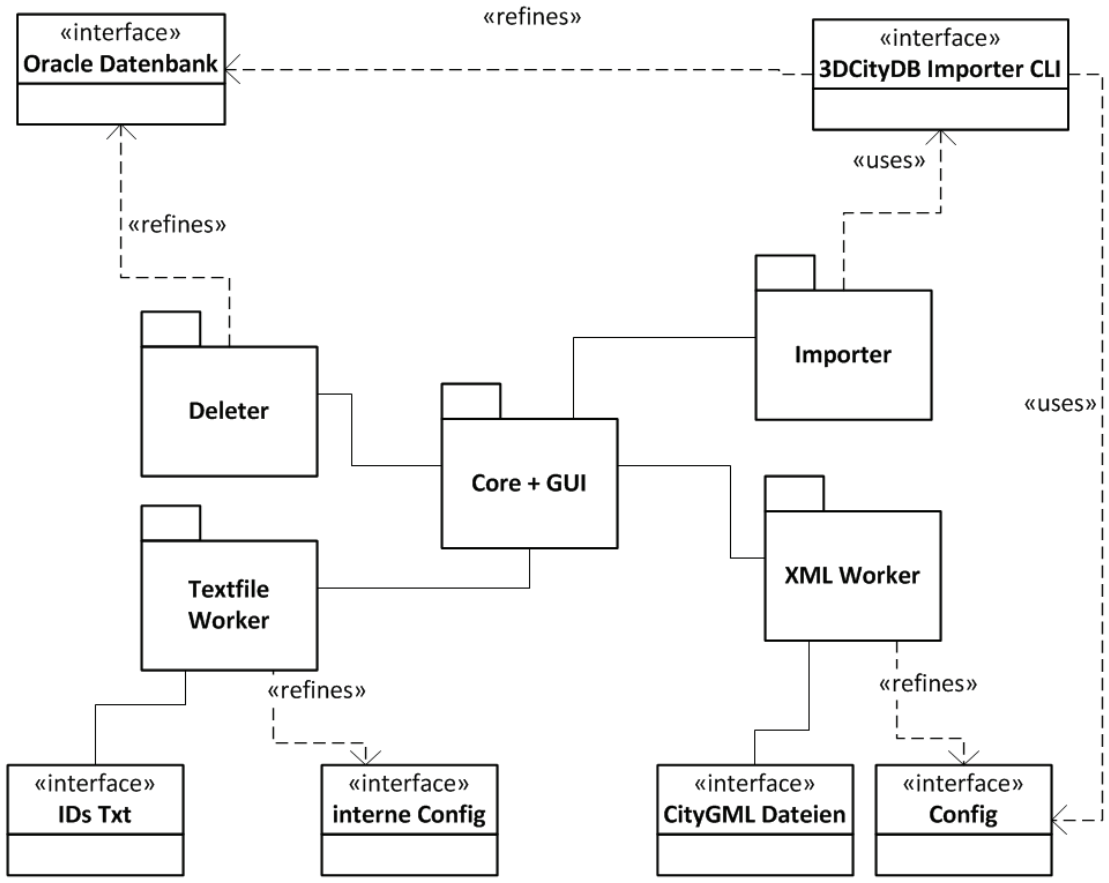


Abbildung 29: grobe Systemarchitektur Fortführungstool 3D City Database

Als gegebene Schnittstellen dienen die Oracle Spatial Datenbank, das CLI des Im- und

Exporter Tools, Textdateien und XML Dateien. Das *Importer-Paket* benötigt die Pfade zu den Import-Dateien, zum Im- und Exporter Programm der 3D City Database und zur (modifizierten) Konfigurationsdatei. Über einen externen Prozess wird dann das CLI des Im- und Exporter Tools mit den o.g. Parametern gestartet. Das Tool kommuniziert anschließend automatisch mit der Datenbank und importiert die gewählten Objekte.

Das *XML-Worker-Paket* übernimmt die XML Dateiarbeit. Seine Hauptaufgabe besteht im Auslesen und Ändern der Konfigurationsdatei. Weiterhin muss es GMLIDs der Gebäudeobjekte aus XML-basierten CityGML auslesen und als Liste wiedergeben.

Das *Textfile-Worker-Paket* übernimmt die normale Dateiarbeit. Es soll die interne Konfigurationsdatei erstellen, auslesen und verändern können. Innerhalb dieser Konfigurationsdatei werden die Pfade zu den anderen Dateien gespeichert. Dies erleichtert die spätere Bedienung, indem die Pfade bei Programmstart bereits bekannt sind.

Das *Delete-Paket* ist das umfangreichste Paket und bildet den Kern dieser Arbeit. Es benötigt eine Liste mit GMLIDs sowie die Verbindungsinformationen aus der Konfigurationsdatei. Das Paket soll eine Verbindung zur Datenbank aufbauen und die GMLIDs in der übergebenen Liste nacheinander abarbeiten. Die Bearbeitung erfolgt demzufolge pro Objekt.
Das *Delete-Paket* selektiert datenbankeindeutige IDs für jeden abgedeckten Entitätstyp für jedes Objekt. Es muss zudem Methoden bereitstellen, um die Mehrfachnutzung von Entitäten zu überprüfen, damit diese wieder aus den Listen entfernt werden können. Dazu gehören beispielsweise Texturen, welche von mehreren Objekten genutzt werden können. Es entstehen dadurch viele mit eindeutigen IDs gefüllte Listen, welche z.T. sehr groß werden können (z.B. Surface_Geometry). Da jede verwendete Liste theoretisch bis zu ca. 2 Milliarden Einträge verwalten kann, muss die Anzahl der Listen nicht überprüft und erweitert werden. Stattdessen müssen Methoden bereitgestellt werden, um sehr lange SQL Befehle in mehrere SQL Statements aufzulösen. Abschließend muss dieses Paket eine Liste mit den GMLIDs der erfolgreich gelöschten Objekte zurückgeben.

6.2.3. Umsetzung

Mithilfe des Entwurfes erfolgt die Umsetzung des Tools. Allgemein ist ein modularisierter Programmaufbau entstanden. Berücksichtigt werden alle wichtigen Programmierprinzipien wie beispielsweise Robustheit und Benutzbarkeit. Der Fokus liegt allerdings bei den Prinzipien Korrektheit und Erweiterbarkeit. Das Programm muss demzufolge alle Funktionen fehlerfrei durchführen und leicht erweitert werden können. Dies setzt eine korrekte fachliche Kenntnis der Funktionen und die Nutzung von Vererbungsprinzipien voraus. Abbildung 30 zeigt die Schnittstellen, Pakete, Klassen und Verbindungen in einem UML Diagramm:

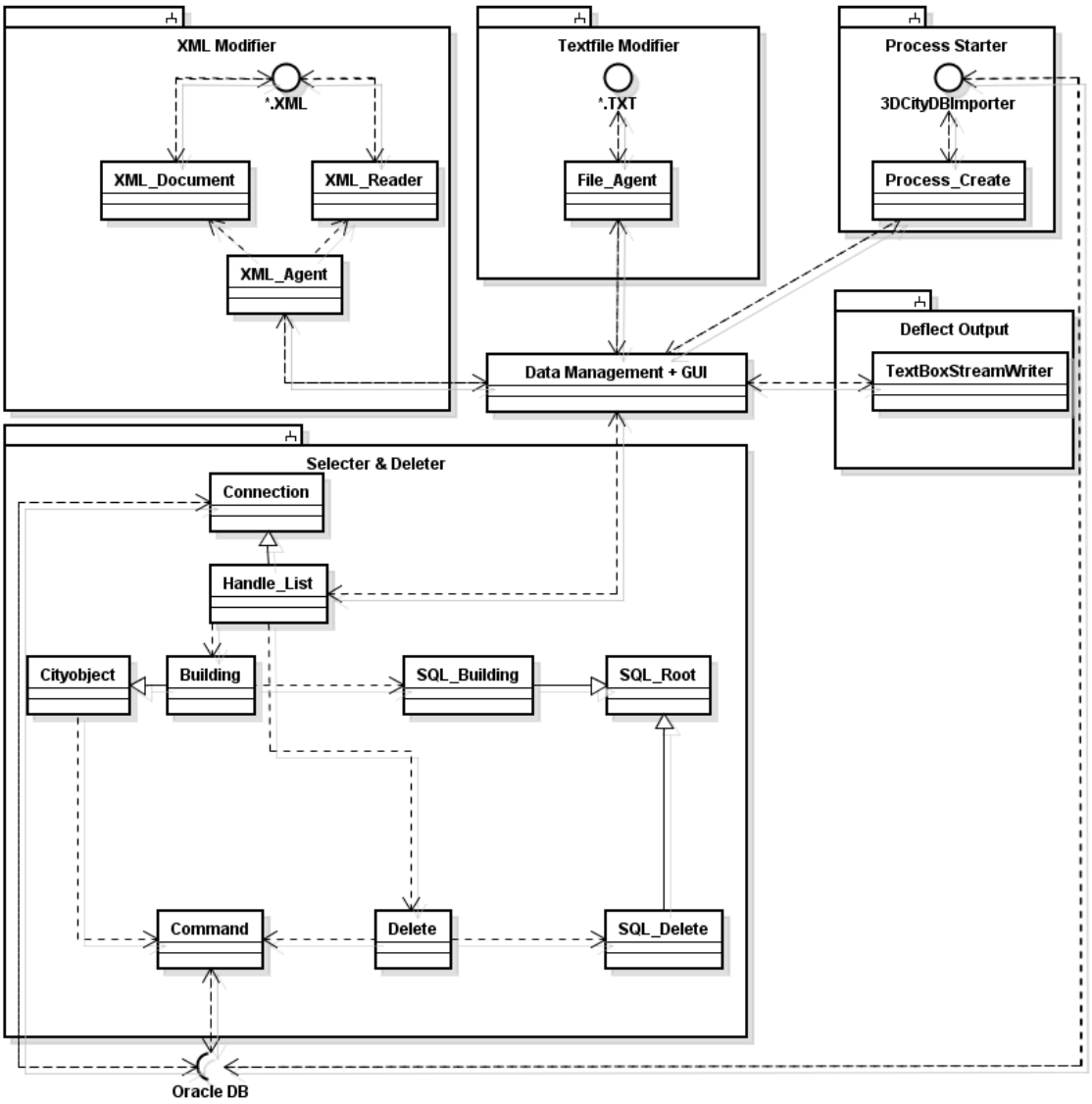


Abbildung 30: UML Diagramm Fortführungstool 3D City Database

Es ist ersichtlich, dass die im Entwurf geschilderten Pakete erhalten bleiben. Durch Klassen und Methoden werden die Funktionen umgesetzt. Erweitert wird das Konzept durch das *Deflect-Output-Paket*. Dieses beinhaltet die *TextBoxStreamWriter* Klasse und kommuniziert mit dem *Data Management (DM)* und der GUI. Sie beinhaltet einen StreamWriter und überlädt den Output der Konsole. Konkret bedeutet dies, dass jede Textausgabe im gesamten Programm über diesen StreamWriter in eine TextBox in der GUI umgeleitet wird.
Die Agent Klassen der *XML-* und *Textfile-Modifier* Klassen setzen die im Entwurf geschilderten Eigenschaften um. Sie werden durch das *DM* von der GUI angesteuert und geben angeforderte Informationen wie Verbindungsdetails und IDs zurück. Das *DM* verwaltet

zentral Listen mit Pfaden, Verbindungsinformationen, IDs etc. Die GUI wird ebenso von dieser Klasse gesteuert. Die Listen werden dabei stets aktuell gehalten. Werden beispielsweise Verbindungsinformationen in der XML Konfigurationsdatei vom *XML_Agent* durch die GUI geändert, werden die Informationen im *DM* aktualisiert. Wenn ein Prozess auf die gespeicherten Informationen zugreift, während Änderungen vorgenommen werden, kann ein inkonsistenter Zustand entstehen. Um diese inkonsistenten Zustände zwischen den in den Dateien gespeicherten Informationen und den im *DM* verwalteten Listen zu vermeiden, wird die Bearbeitung während des Lösch- oder Importvorgangs deaktiviert. Sobald keine aktiven Prozesse mehr auf die Listen zugreifen, wird die Bearbeitung wieder freigegeben.
Die Funktionen des *Importer-Pakets* (*Process Starter*) werden durch die Klasse *Process_Create* umgesetzt. Sie übernimmt, wie im Entwurf bereits geschildert, die Kommunikation mit dem Im- und Exporter Tool der 3D City Database. Der Import erfolgt mit den internen Methoden des Importers. Um die vom externen Tool gesendeten Meldungen wie Anzahl importierter Objekte und Fehlermeldungen auszugeben, wird der Prozessoutput über Prozessevents gefangen und an die TextBox über den *TextBoxStreamWriter* weitergegeben. Über ein weiteres Event wird dem *DM* mitgeteilt, dass der Prozess beendet ist.
Das Löschen alter Objekte übernimmt das *Selecter- und Deleter-Paket*. Dabei wird eine Liste mit GMLIDs an die *Handle_List* Klasse übergeben. Diese Klasse erbt wiederum die Eigenschaften der *Connection* Klasse, welche beim Instanziieren der *Handle_List* Klasse automatisch eine provisorische Verbindung zur Oracle Datenbank aufbaut und die Existenz der in der Liste übergebenen GMLIDs überprüft. Das Resultat ist eine Liste mit den tatsächlich in der Datenbank befindlichen GMLIDs. Anschließend erfolgt an dieser Stelle eine Typunterscheidung, um welchen Objekttypen es sich handelt. In dem Tool werden nur Gebäudeobjekte unterstützt. Für andere Objekttypen muss die Unterscheidung an dieser Stelle erfolgen.
Daraufhin wird die Liste Objekt für Objekt abgearbeitet. Dabei wird für jedes Objekt eine Verbindung zur Datenbank hergestellt, welche für den gesamten Selektions- und Löschvorgang gültig ist. Wenn es sich um ein Gebäudeobjekt (Building) handelt, wird die Verbindung und die GMLID an die *Building* Klasse weitergereicht. Diese erbt von der Basisklasse *Cityobject*, welche allgemeine Methoden besitzt und die Kommunikation mit der *Command* Klasse übernimmt. Die *Command* Klasse sendet SQL Strings an die Oracle Datenbank und gibt die Datenbankantworten zurück.
Innerhalb der *Building* Klasse erfolgt die Selektion auf Grundlage der beschriebenen ER Modelle (Abbildungen 8, 9 und 10). Die SQL Strings zur Selektion werden mithilfe der in der *Cityobject* Klasse gespeicherten Methode *Universal_Getter* erzeugt. Die grundlegenden SQL Befehle wie z.B. „SELECT ID FROM THEMATIC_SURFACE WHERE (ROOM_ID = ' “ werden in *SQL_Building* gespeichert. Diese Grundanweisung und eine Liste mit IDs wird an die von der *SQL_Root* Klasse geerbten Methode *Create_String* übergeben. Mithilfe dieser Methode werden die SQL Strings erzeugt. Die Besonderheit dieser ist, dass lange SQL Befehle vermieden werden, indem pro 100 IDs ein separater String erstellt wird. Die Teilung ist notwendig, da die Datenbank zu lange SQL Anweisungen nicht ausführen kann. Diese Strings werden in einer Liste gespeichert

und der *Building* Klasse zurückgegeben. Mithilfe der *Universal_Getter* Methode werden die Befehle einzeln an die *Command* Klasse und damit auch an die Oracle Datenbank übergeben. Die Ergebnisse werden dann wiederum in einer Liste zusammengefasst und zurückgegeben. Nach Abschluss aller Selektionsvorgänge ist pro genutzten Entitätstypen jeweils eine Liste entstanden. Diese Listen werden zu einem Dictionary zusammengefasst und der *Delete* Klasse übergeben. Auch diese erzeugt SQL Löschanweisungen mithilfe der von *SQL_Root* geerbten *Create_String* Methode. Unter Berücksichtigung der Abhängigkeiten der in der Datenbank gespeicherten Entitäten (nach ER Modell) werden diese SQL Befehle durch die *Command* Klasse an die Oracle Datenbank weitergeleitet. Nach Abarbeitung aller Listen im Dictionary ist der Löschvorgang abgeschlossen.
Zusätzlich wird über Events eine Statusbar verwaltet, welche den Fortschritt des Löschprozesses auf der GUI anzeigt. Ein weiteres Event gibt aktuelle Informationen aus, welche Objekte gelöscht wurden. Durch Nutzung des Transaktionskonzeptes wird gesichert, dass Objekte nur gelöscht werden, wenn keine Fehler auftreten.
Der Arbeitsablauf eines Löschprozesses wird im Anhang B schemenhaft dargestellt. Abbildung 31 zeigt die Entwurfsansicht der GUI:

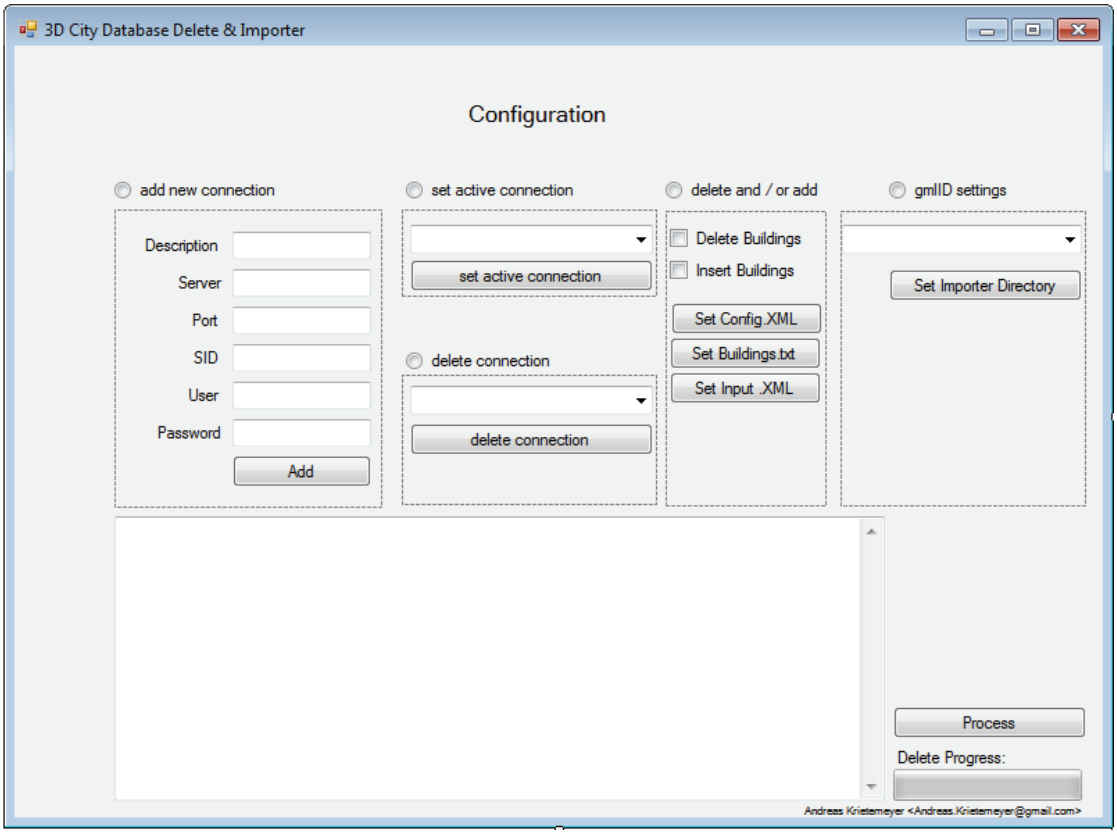


Abbildung 31: Entwufsansicht GUI Fortführungstool 3D City Database

6.2.4. Erweiterungsmöglichkeiten

Durch den modularen Programmaufbau kann das Tool mit der Umsetzung weiterer Datenmodelle wie z.B. Stadtmöblierungen und Vegetationen erweitert werden. Dazu muss eine Typunterscheidung in der *Handle_List* Klasse durchgeführt werden, welche mithilfe einer Datenbankabfrage zusammen mit einer Switch-Case Anweisung erfolgen kann. Die zusätzlichen Objektbehandlungen sollten als Schnittstellen behandelt werden, welche sich in das bestehende System integrieren lassen. Die Basisklassen *Cityobject* und *SQL_Root* sollten daher von den neuen Klassen geerbt werden, um auf die umfangreichen, erfolgreich getesteten Methoden zugreifen zu können.
Durch Definition kritischer Blöcke und Starten mehrerer Threads kann die Abarbeitung weiter optimiert werden. Zu beachten sind dabei die Beziehungen der Entitätstypen untereinander. Es müssen demzufolge sowohl bei der Selektion, als auch im Löschvorgang kritische Blöcke definiert werden. Mit dieser synchronen und asynchronen Abarbeitung wurde begonnen.
Verbessert werden sollte der Output des *Importer-Pakets*. Wenn das externe Tool innerhalb kürzester Zeit sehr viele Fehlermeldungen produziert (z.B. fehlende Texturen), ist die Abarbeitung des StreamWriters zu langsam. Dies hängt damit zusammen, dass jede Textausgabe des Prozesses über ein Event gefangen und über den StreamWriter ausgegeben werden muss. Dieses Verfahren benötigt verhältnismäßig viel Zeit. Besser wäre es, alle Meldungen in eine Log Datei zu speichern und nur die wichtigsten Meldungen auf die GUI auszugeben.

6.2.5. Funktionstest

Um die Funktionsfähigkeit des Tools zu testen, wird ein Fortführungsprozess durchgeführt. Als Eingangsdaten dienen die Datensätze 1 und 2 aus Kapitel 6.1. Mithilfe des eigenen Tools wird der Datensatz 1 (Stadtmodell ohne Texturen) in die Datenbank importiert. Dabei wird, wie bereits geschildert, das Importer Tool der 3D City Database über ein CLI aufgerufen. Abbildung 32 visualisiert den in der Datenbank importierten Datensatz 1.

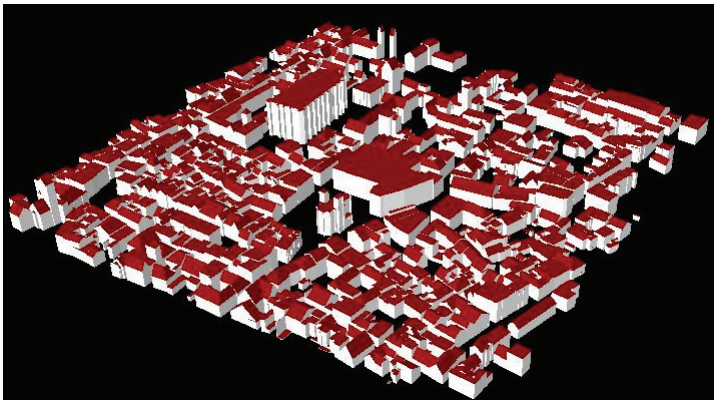


Abbildung 32: Visualisierung importierter Datensatz 1

Die Ausgangssituation ist demzufolge ein in der Datenbank befindliches 3D Stadtmodell, welches mithilfe des Datensatzes 2 veredelt werden soll. Das Tool wird anschließend durch die GUI für den Fortführungsprozess konfiguriert. Dabei liest das Tool die GMLIDs aus dem Datensatz 2 aus und löscht die dadurch identifizierten Objekte aus der Datenbank. Nach Abschluss des Prozesses werden die neuen Objekte automatisch durch das Importer Tool in die Datenbank importiert. Eine Log Kopie des TextBox Outputs während eines Fortführungsprozesses befindet sich im Anhang C. Ein Fortführungsprozess ist als Video auf der beiliegenden CD verfügbar. Abbildung 33 visualisiert das Ergebnis.

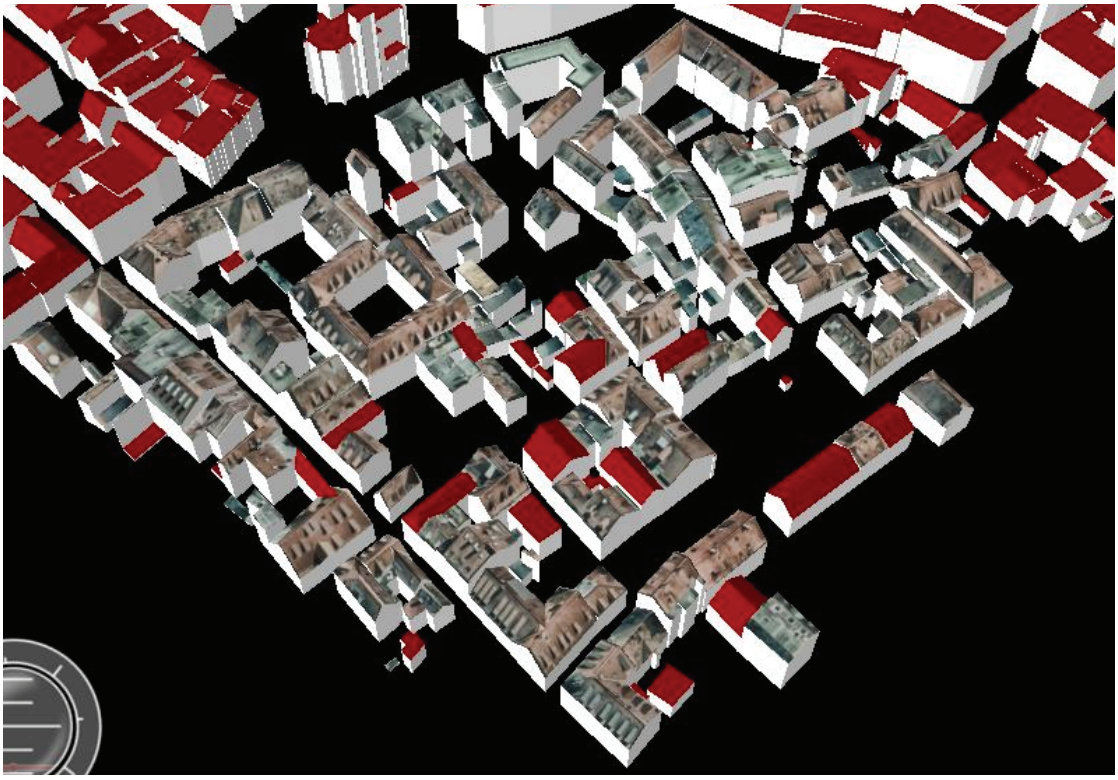


Abbildung 33: Visualisierung Fortführungsprozess 3D City Database

Erkennbar sind Geometrie- und Semantikänderungen. Es wurde dasselbe Ergebnis wie beim Fortführungsprozess im CityServer3D erreicht. Der Fortführungsprozess mit Gebäudeobjekten wurde auch in der 3D City Database erfolgreich umgesetzt.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurden die Datenbanklösungen CityServer3D und 3D City Database hinsichtlich ihrer Im- und Exportfunktionen unterschiedlicher Datenformate sowie einer Möglichkeit zur Datenfortführung untersucht. Dabei fanden Datenhaltungstests mit verschiedenen Beispielen statt. In diesem Zusammenhang wurden aufgetretene Probleme begründet und behoben. Es erfolgte die Erkenntnis der Funktionsunfähigkeit des Rasterdatenim- und Export Tools der 3D City Database unter einer deutschen Oracle Umgebung. Dazu wurde eine alternative Speicherungs- und Exportmöglichkeit entwickelt.
Weiterhin wurden Performance Tests mit markanten CityGML Dateien in beiden Datenbanklösungen ausgeführt und verglichen. Dabei stellte sich heraus, dass der CityServer3D aufgrund der schwachen Hardware und der integrierten Visualisierung tendenziell langsamer als die 3D City Database in der lokalen Umgebung ist. Darüber hinaus stellte sich heraus, dass die Im- und Exportzeit von der Dateigröße sowie der Objektanzahl abhängt.
Des Weiteren wurden die Eigenschaften der Datenbanklösungen miteinander verglichen. Der Funktionsumfang und die Kompatibilität mit verschiedenen Systemen ist im kommerziellen CityServer3D System der Open Source Variante, der 3D City Database, überlegen. Die 3D City Database überzeugt dafür durch eine komplette Realisierung aller CityGML Klassen.
Weiterhin wurde im CityServer3D mithilfe der integrierten regelbasierten Sprache eine simple Fortführungsmöglichkeit realisiert. Bei Kenntnis der Befehle können mithilfe dieser Sprache Arbeitsprozesse schnell und effizient umgesetzt werden. Abschließend wurde für die 3D City Database ein umfangreiches Tool zum Importieren, Löschen und Fortführen von Gebäudeobjekten in einer Oracle Datenbank umgesetzt.

Allgemein wird eine gute Hardwareausstattung und die Trennung zwischen Client und Server empfohlen, um Performanceeinbußen zu vermeiden. Die Nutzung des CityServer3D Pakets ist zu empfehlen, wenn ein komplett funktionsfähiges System zur Speicherung, Bearbeitung und Visualisierung verschiedener Datenformate benötigt wird, was nicht von speziellen (objektrelationalen) Datenbanksystemen abhängig sein soll. Wenn die gespeicherten Daten jedoch direkt genutzt werden sollen, wird der Nutzer an die Produkte des CityServer3D gebunden, da das Datenmodell unbekannt ist und stets die Schnittstelle des Metamodells genutzt werden muss. Die Datenbanklösung 3D City Database sollte verwendet werden, wenn ausschließlich CityGML Dateien verwaltet werden und die Bearbeitung, Nutzung und Visualisierung der Daten extern vorgenommen wird. Sie ist besonders für Nutzer mit Programmierkenntnissen geeignet, welche innovative eigene Anwendungen erstellen wollen. Vorteilhaft ist der Direktzugriff auf die Datenbank. Mit dem Datenmodell der 3D City Database als Grundlage können demzufolge komplett neue GIS entwickelt werden.

Der CityServer3D bietet verschiedene OGC Webdienste, wie dem Web 3D Service (W3DS), dem Web Perspective View Service (WPVS) und dem Web Feature Service (WFS) an.

Um diese Dienste auch in der 3D City Database zu nutzen, müssen die entsprechenden Module entwickelt werden. Diese Arbeit kann durch eine Untersuchung dieser Dienste im CityServer3D und die Entwicklung dieser Schnittstellen für die 3D City Database weitergeführt werden. Eine weitere Möglichkeit zur Weiterführung dieser Arbeit ist es, einen umfangreicheren Geschwindigkeitstest mit einer größeren Anzahl von Daten in beiden Datenbanklösungen zu testen. Im CityServer3D sollten auch die Geschwindigkeitsvorteile einer dokumentenorientierten Datenbank gegenüber einem objektrelationalen Datenbanksystem untersucht werden.
Bei beiden Datenbanklösungen sind Trends beobachtbar. Es werden immer weniger kommerzielle Datenbanksysteme verwendet. Diese Arbeit kann durch eine Untersuchung der neueren Eigenschaften der 3D City Database erweitert werden. Dazu zählt vor allem der Test in einer PostgreSQL Datenbank. Dies ist besonders interessant, da die 3D City Database zusammen mit dieser Datenbank komplett Open Source genutzt werden kann und keine Lizenz für das Datenbanksystem erworben werden muss, wie es bei der Oracle Datenbank der Fall wäre. In diesem Zusammenhang kann auch das Fortführungstool neben den bereits in Kapitel 6.2.4 beschriebenen Erweiterungsmöglichkeiten auch für die PostgreSQL Datenbank erweitert werden.
Für die 3D City Database sollte zudem ein Tool zum Im- und Export von Orthophotos nach den in Kapitel 5.3 beschriebenen Anweisungen entwickelt werden. Neben der beschriebenen Datenbankarbeit muss das Tool auch die exportierten Objekte zu einem zusammenhängenden Orthophoto zusammenfügen können.
Der CityServer3D und die 3D City Database werden von den Herstellern fortlaufend erweitert und verbessert, was weiterführende Untersuchungen ermöglicht.

Abkürzungsverzeichnis

GIS	Geoinformationssystem
ALK	Automatisierte Liegenschaftskarte
CSG	Constructive Solid Geometry
SIG 3D	Special Interest Group 3D
GDI DE	Gesellschaft für Geodateninfrastruktur Deutschland
CityGML	City Geography Markup Language
LoD	Level of Detail
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
GUI	Graphical User Interface
CLI	Command Line Interface
XML	Extensible Markup Language
GML	Geography Markup Language
KML	Keyhole Markup Language
Collada	Collaborative Design Activity
GeoTIFF	Geo Tagged Image File Format
OGC	Open Geospatial Consortium
W3DS	Web 3D Service
WPVS	Web Perspective View Service
WFS	Web Feature Service
ISO	International Organization for Standardization
SQL	Structured Query Language
ER	Entity Relationship
UML	Unified Modeling Language
NoSQL	Not only Structured Query Language
FS	Filesystem
IGD	Fraunhofer Institut für Graphische Datenverarbeitung
IGG	Institut für Geodäsie und Geoinformationstechnik
GDI	Geodaten Infrastruktur
DM	Data Management

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[ABH03] Jörg Albert, Dr. Matthias Bachmann, and Dr. Achim Hellmeier. Zielgruppen und Anwendungen für digitale Stadtmodelle und digitale Geländemodelle, Erhebungen im Rahmen der Arbeitsgruppe Anwendungen und Zielgruppen der SIG3D im Rahmen der Initiative GDI-NRW, 2003.

[AG 04] AG 3D Stadtmodelle des AK Kommunales Vermessungs- und Liegenschaftswesen des Städtetages NRW. 3D Stadtmodelle Orientierungshilfe, 2004.

[Bri08] Thomas Brinkhoff. *Geodatenbanksysteme in Theorie und Praxis.* Herber Wichmann Verlag, 2008.

[Fra12] Fraunhofer Institut für graphische Datenverarbeitung Darmstadt. Feature Matrix City Server 3D, 2012. http://www.cityserver3d.de/features/.

[GKCN08] Gerhard Gröger, Thomas H. Kolbe, Angela Czerwinski, and Claus Nagel. OpenGIS City Geography Markup Language (CityGML) Encoding Standard. Technical report, Open Geospatial Consortium Inc, 2008.

[GKD+04] Gerhard Gröger, Thomas Kolbe, R. Drees, A. Kohlhaas, H. Müller, Dr. F. Knospe, U. Gruber, and U. Krause. Das interoperable 3D Stadtmodell der SIG 3D der GDI NRW. Technical report, Initiative Geodaten Infrastruktur NRW, 2004.

[Gre06] Dominik Grether. Semantische Annotation und Deduktion in Geoinformationssystemen, 2006. Diplomarbeit, Abteilung Künstliche Intelligenz, Universität Ulm.

[Her11] Herreruela, Javier and Kolbe, Thomas H. and Nagel, Claus and König, Gerhard. CityGML DBMS storage, 3DCityDB implementation, 2011.

[JBo12] JBoss Inc. Drools Expert Documentation, 2012. http://www.jboss.org/drools/documentation.

[KB98] Alexander Köninger and Sigrid Bartel. 3D Gis for Urban Purposes, 1998.

[KKN12] Prof. Dr. Thomas H. Kolbe, Gerhard König, and Claus Nagel. *3D Geo Database for CityGML*, 2.0.6 edition, 2012.

[Kri12a] Andreas Krietemeyer. Features der 3D CityDB, 2012.

[Kri12b] Andreas Krietemeyer. XLinks und Semantik, 2012. Version 1.2.

[LL97] Hansjörg Luser and Günther Lorber. 3D Stadtmodell Graz, Anforderungen Ansprüche Anwendungen, 1997.

[Lor96] Günther Lorber. *Aufbau des 3D Stadtmodells Graz. Beiträge zum XII. Internationalen Kurs für Ingenieurvermessung.* Dümmler Verlag, 1996.

[Ora03] Oracle, America Inc. *GeoRaster Overview and Concepts*, 2003. http://docs.oracle.com/html/B10827_01/geor_intro.htm.

[Rhe09] Rheinmetall Defence Electronics GmbH. Einsatzpotential von 3D Stadtmodellen für Simulation und Training, 2009.

[SK] Alexandra Stadler and Thomas H. Kolbe. Spatio Semantic Coherence in the integration of 3D City Models.

[SK07] Alexandra Stadler and Thomas H. Kolbe. CityGML für die 3D Navigation, 2007.

[Spe] Special Interest Group 3D. SIG 3D. http://www.sig3d.org/.

[Sta07] Alexandra Stadler. Kohärenz von Geometrie und Semantik in der Modellierung von 3D Stadtmodellen. Technical report, TU Berlin, Institut für Geodäsie und Geoinformationstechnik, 2007.

[Tis09] Jan Tischer. Modellierung und Ausgleichung flächenparametrisierter 3D Gebäudemodelle mit der objektrelationalen Datenbank Oracle Spatial, 2009. Diplomarbeit, Fakultät Planen Bauen Umwelt, Technische Universität Berlin.

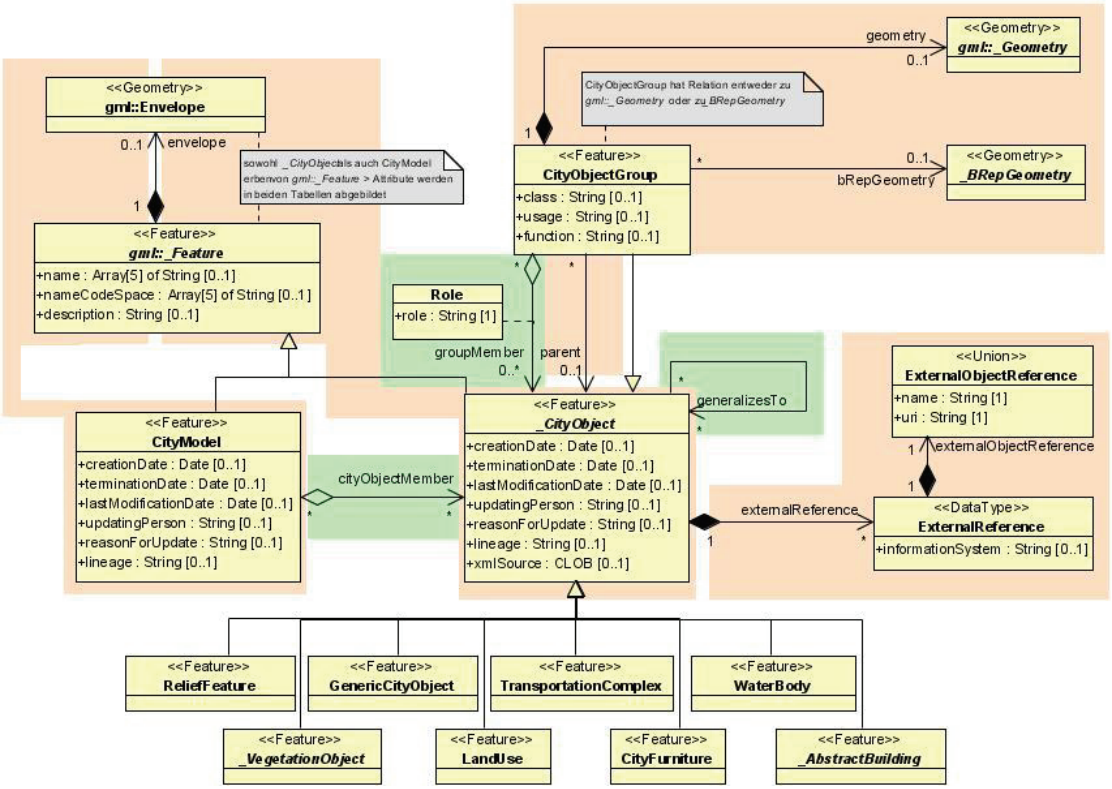
[Use05] Martin Use. 3D Routennavigation, 2005. Masterarbeit, Hochschule Furtwangen, Institut für graphische Informationssysteme.

[Vre10] Panagiotis A. Vretanos. OpenGIS Web Feature Service 2.0 Interface Standard. Technical report, Open Geospatial Consortium Inc., 2010.

[Xie09] Jeffrey Xie. Reported XDB bug, 2009. https://forums.oracle.com/forums/thread.jspa?messageID=3669428.

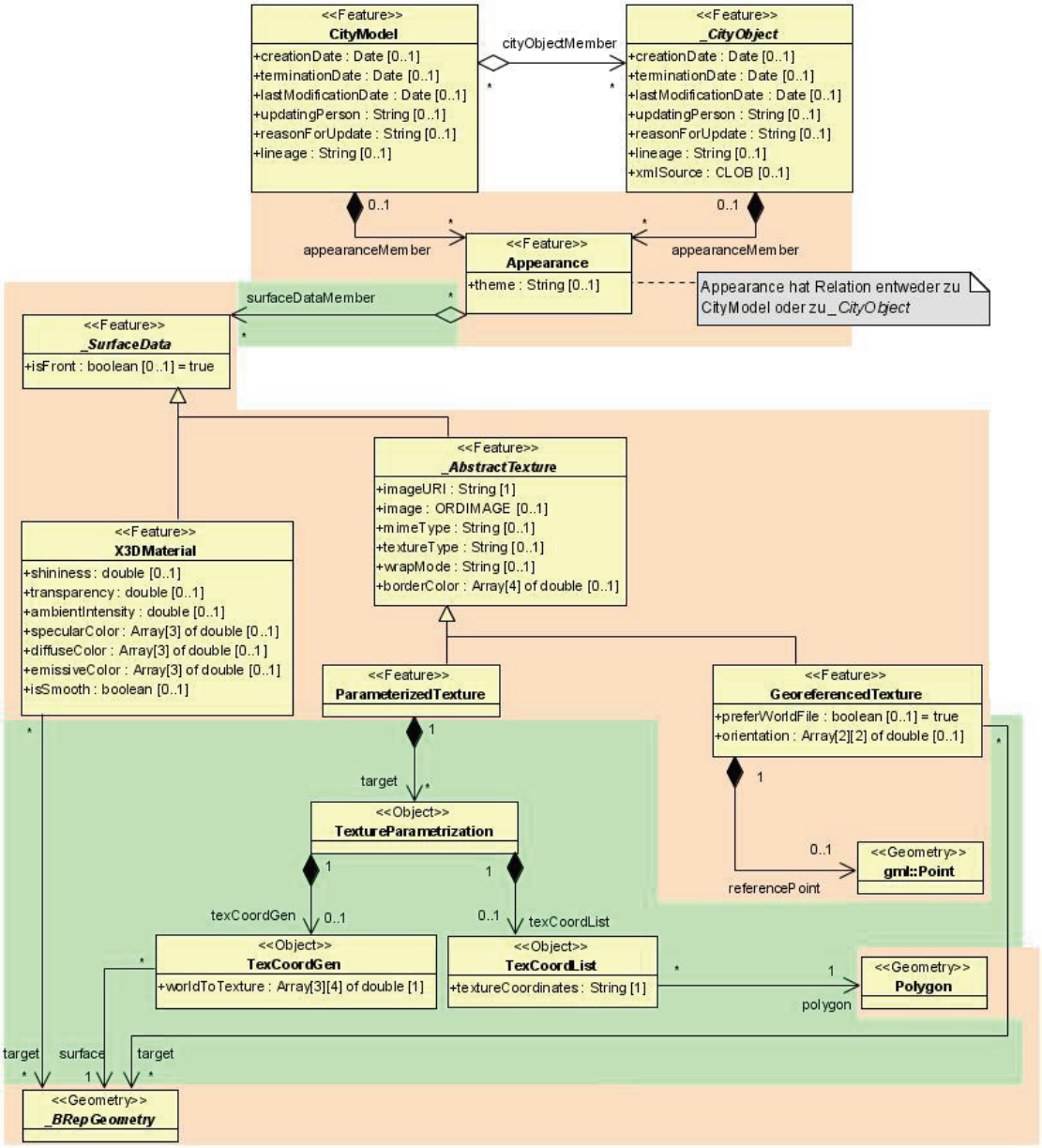
A. CityGML Klassen

A.1. Core Model



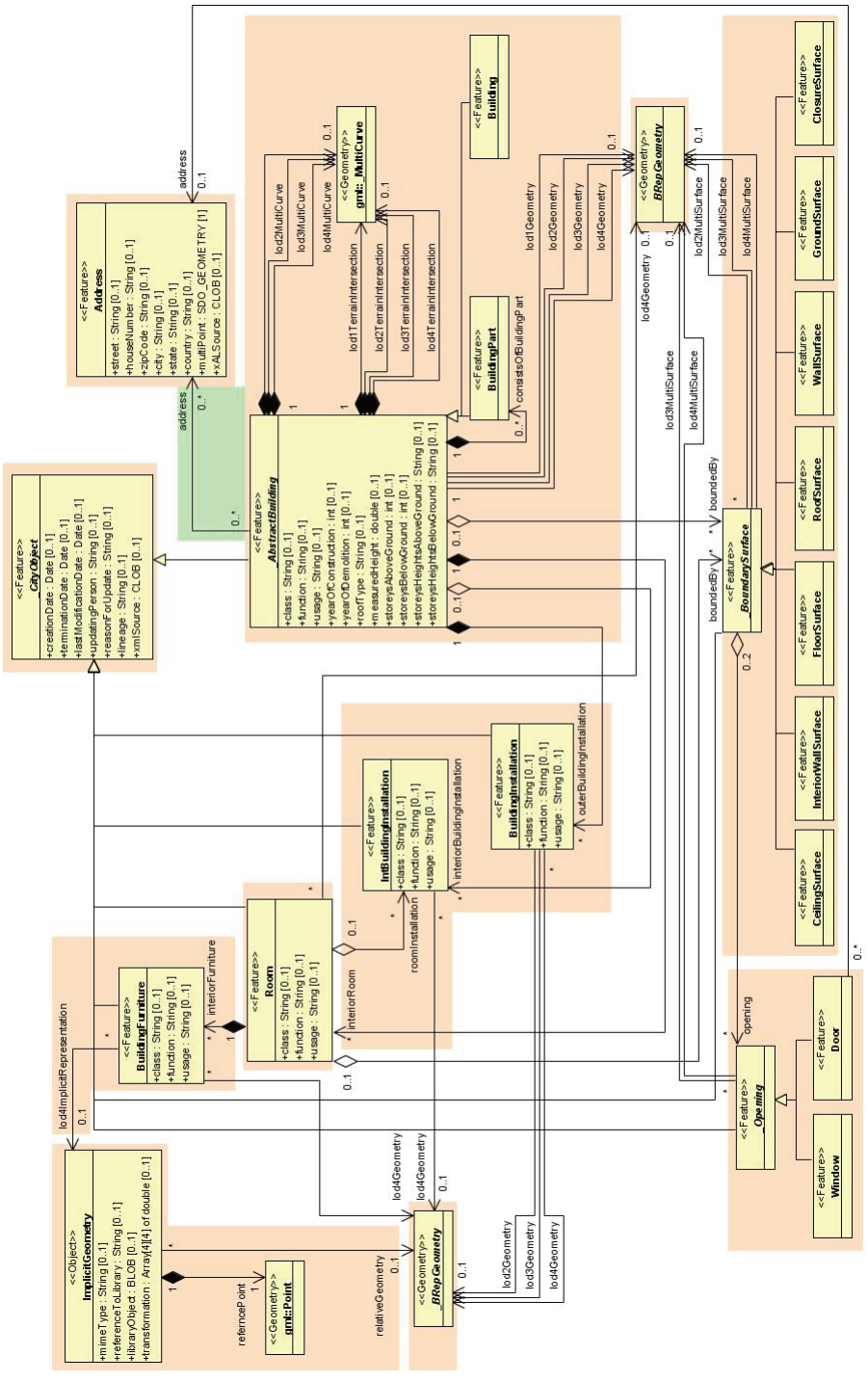
aus [KKN12]

A.2. Appearance Model



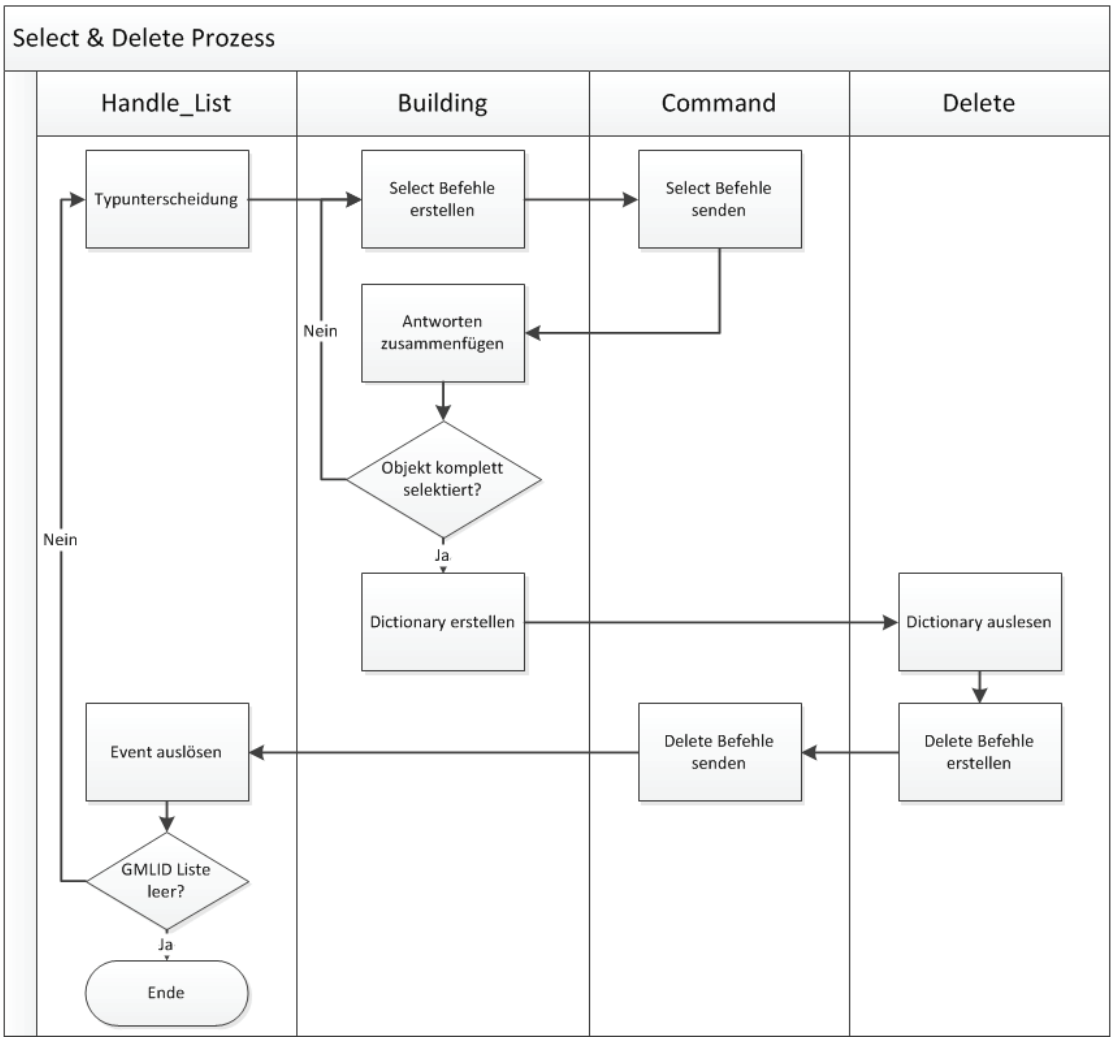
aus [KKN12]

A.3. Building Model



aus [KKN12]

B. Delete Prozess 3D City Database



C. Log Ausschnitt Fortführung 3D City Database

[07.08.2012 11:39:59] active connection is [User] Kriete orcl.kriete.lappy@localhost:1521
[07.08.2012 11:40:05] 115 IDs loaded
[07.08.2012 11:40:11] Processing started!
[07.08.2012 11:40:12] Delete progress started.
[07.08.2012 11:40:13] 6284658 finished
[07.08.2012 11:40:14] 6284667 finished
[07.08.2012 11:40:14] 6284672 finished
...
[07.08.2012 11:40:52] 6284748 finished
[07.08.2012 11:40:52] 6285677 finished
[07.08.2012 11:40:52] 6285694 finished
[07.08.2012 11:40:53] 6285702 finished
[07.08.2012 11:40:54] 6285724 finished
[07.08.2012 11:40:54] Delete Progress finished!
[07.08.2012 11:40:54] Insert progress started
[11:40:55 INFO] Starting 3D City Database Importer/Exporter, version "1.3-b1"
[11:40:55 INFO] Initializing application environment
[11:40:59 INFO] Loading project settings
[11:41:00 INFO] Connecting to database profile 'User'.
[11:41:00 INFO] Database connection established.
[11:41:00 INFO] Oracle Database 11g Release 11.1.0.0.0 - Production
[11:41:00 INFO] SRID: 32632
[11:41:00 INFO] gml:srsName: urn:ogc:def:crs,crs:EPSG:6.12:32632,crs:6.12:5783
[11:41:00 INFO] Versioning: OFF
[11:41:00 INFO] Initializing database import...
[11:41:00 INFO] Creating list of CityGML files to be imported...
[11:41:00 INFO] List of import files successfully created.
[11:41:00 INFO] 1 file(s) will be imported.
[11:41:02 INFO] Importing file: Fortfuehrung.xml
[11:41:07 INFO] Resolving XLink references.
[11:41:07 INFO] Resolving appearance XLinks...
[11:41:08 INFO] Importing texture images...
[11:41:29 INFO] Cleaning temporary cache.
[11:41:30 INFO] Imported CityGML features:
[11:41:30 INFO] Appearance: 1
[11:41:30 INFO] Building: 115
[11:41:30 INFO] BuildingPart: 176
[11:41:30 INFO] GroundSurface: 233
[11:41:30 INFO] RoofSurface: 233
[11:41:30 INFO] WallSurface: 233
[11:41:30 INFO] Processed geometry objects: 4376
[11:41:30 INFO] Database import successfully finished.