**1. Beschreibung des Parser-Moduls**

Im nachfolgenden Kapitel der Dokumentation werden wir auf die Funktionen und die Benutzung des Parsers eingehen. Das Modul deckt die folgenden Bereiche der vCity-Software ab:

* Import und parsen der CityGML-Datei
* Koordinatentransformation zu einem definierten Referenzwert
* Polygon-Triangulation
* Drei verschiedene Exportvarianten

**7. Datenexport**

**CSV**

Diese Option schreibt die Gebäude-IDs und deren berechneten Volumen in eine einfache CSV (**C**omma **S**eparated **V**alues) Datei.

Die Exportmethode verwendet dabei einen Java **FileWriter**, weil wir es unnötig fanden, hierfür ein extra Framework heran zu ziehen.

Building,Volume

DEBW\_LOD2\_1007722,112.82454681396484

**GML**

Die Software kann die eingelesene GML (**G**eography **M**arkup **L**anguage)-Datei wieder exportieren. Dabei wird dem geparsten Datenmodell das errechnete Volumen der Gebäude als double-Wert hinzugefügt und danach als GML-Datei ausgeschrieben.

Der Export verwendet das **CityGML4J** Framework, welches auch für den Import verwendet wird. Das Framework steht unter der GNU LGPL-Lizenz.

<gen:doubleAttribute name="Volume">

<gen:value>112.82454681396484</gen:value>

</gen:doubleAttribute>

**XML**

Die dritte Exportmöglichkeit ist der Export als XML (*e****X****tensible* ***M****arkup* ***L****anguage*)-Datei. Dieser Export ist für eine Weiterverwendung mit dem externen Programm “**INSEL**” gedacht.

Das XML-Datenformat ist dabei nach den Anforderungen eines Kundenreviews entworfen worden. Es ist zu beachten, dass Koordinaten und ähnliches nicht in diesem Format exportiert werden.

Die DTD (**D**ocument **T**ype **D**efinition) des XML-Dokuments ist wie folgt definiert:

<!ELEMENT City (SkyModel, Building+)>

<!ELEMENT SkyModel (SplitAzimuth, SplitHeight)>

<!ELEMENT SplitAzimuth (#PCDATA)>

<!ELEMENT SplitHeight (#PCDATA)>

<!ELEMENT Building (BoundarySurface)+>

<!ATTLIST Building

id CDATA #REQUIRED

>

<!ELEMENT Volume (#PCDATA)>

<!ATTLIST Volume

uom CDATA #REQUIRED

>

<!ELEMENT BoundarySurface (Polygon)+>

<!ATTLIST BoundarySurface

id CDATA #REQUIRED

type (GROUND|ROOF|WALL|OTHER) #REQUIRED

>

<!ELEMENT Polygon (Area, Shadow)>

<!ATTLIST Polygon

id CDATA #REQUIRED

>

<!ELEMENT Area (#PCDATA)>

<!ATTLIST Area

uom CDATA #REQUIRED

>

<!ELEMENT Shadow (#PCDATA)>

Zum erstellen des XML-Datenobjekts verwenden wir das **StAX** Framework, das unter der Apache v2.0 Lizenz vorliegt. Wir haben uns für dieses Framework entschieden, weil es Performant und leicht zu verwenden ist.

Wir erstellen XML-Objekte mit der Methode Element e = doc.createElement("Element"); . Attributs werte werden mit der Methode Attr a = doc.createAttribute("Attribute"); erstellt. Der Vorteil hierbei ist, dass wir zum Füllen des Datenobjekts einfach über jedes Gebäude gehen und die benötigten Daten auslesen können.

for (each Building)

add ID;

add Volume;

for (each Surface)

...

**8. Testfälle**

Um die Funktionalität des Imports und des Exports zu testen, haben wir eine JUnit-Testklasse geschrieben. Die Klasse enthält diverse Testfälle für den Import, das Parsen, die Triangulation, die Transformation und die Exportmöglichkeiten. Die größte Methode hier ist testReadAndParseAndValidateEinHaus(), welche einen kompletten Import der Gruenbuehl GML-Datei enthält, dabei wird jeder Schritt geprüft und validiert. Die drei Exportvarianten werden ebenfalls ausgeführt, jedoch müssen die geschriebenen Dateien bislang noch manuell überprüft werden.