Dokumentation zum

Shape 2 CityGML -Konverter

--- Version 1.0 ---

Autoren

Claus Nagel <claus.nagel@tu-berlin.de>
Alexandra Lorenz <lorenz@tu-berlin.de>
Jannes Bolling < jbo023@googlemail.com>

Institut für Geodäsie und Geoinformationstechnik
Technische Universität Berlin
http://www.gis.tu-berlin.de/

Berlin, 18. März 2011

1 Einleitung

1.1 Funktion der Software

Der Shape2CityGML-Konverter ermöglicht das Ableiten von 3D CityGML-Gebäudemodellen aus Shape Dateien mit 2D-Grundrissinformationen. Die Grundrisse werden anhand von Höhendaten, die in der Shape Datei attributiv vorliegen, extrudiert und in eine gültige CityGML LOD1-Repräsentation überführt. Neben der Umwandlung der Grundrisspolygone können weitere Shape Attribute auf thematische CityGML Attribute abgebildet werden. Der Shape2CityGML-Konverter benötigt für die Konvertierung eine vorkonfigurierte Projektdatei, in welcher die Umsetzung von Shape Attributen auf entsprechende CityGML Attribute gespeichert ist.

1.2 Systemvoraussetzungen

Der Shape2CityGML-Konverter ist vollständig in Java implementiert. Zur Ausführung der Anwendung muss daher eine Java-Laufzeitumgebung auf dem Computer installiert sein. Es wird empfohlen, die offizielle und kostenfreie Java Runtime Environment (JRE) von Oracle zu verwenden. Mit alternativen Laufzeitumgebungen kann es zu unerwartetem oder fehlerhaftem Programmverhalten kommen, wenn diese nicht vollständig kompatibel zur JRE sind.

Die JRE kann von der Internetseite http://www.java.com/ bezogen werden. Für die Ausführung des Shape2CityGML-Konverters wird die JRE mindestens in der **Version 6 Update 5** (oder höher) benötigt. Bitte stellen Sie sicher, dass die korrekte JRE für Ihr Betriebssystem verwendet wird. Weitere Hinweise für die Installation der JRE werden auf der genannten Internetseite bereitgestellt.

Der Shape2CityGML-Konverter kann auf unterschiedlichen Betriebssystemen ausgeführt werden, sofern zuvor die Java-Laufzeitumgebung erfolgreich eingerichtet wurde. Neben WindowsXP, Windows Vista und Windows 7 als Vertreter der Microsoft Windows-Familie wurde die Anwendung auf Apple MacOS X sowie auf verschiedenen Linux-Derivaten erfolgreich getestet.

Die Programmdateien benötigen ca. 14MB an Speicherplatz auf Ihrer Festplatte bzw. auf einem alternativen Speichermedium (z.B. USB-Speicher-Stick). Weitere Mindestanforderungen an die Hardware des Computers, auf welchem der Shape2CityGML-Konverter ausgeführt werden soll, bestehen nicht.

1.3 Installation

Der Shape2CityGML-Konverter liegt als gepacktes .zip-Archiv („shape2citygml.zip“) vor. Bitte entpacken Sie dieses Archiv an einen beliebigen Ort auf Ihrem Computer. Auf Windows-Rechnern (WindowsXP oder höher) kann das Entpacken von .zip-Archiven direkt mit Hilfe des Windows Explorers erfolgen. Alternativ können im Internet freie Packprogramme bezogen werden (z.B. 7-Zip für Windows, http://www.7-zip.org/). Auch MacOS X oder aktuelle Linux-Derivate stellen entsprechende Programme zum Entpacken von .zip-Archiven bereit.

Nach dem Entpacken der Archivdatei „shape2citygml.zip“ befindet sich im Hauptverzeichnis `shape2citygml` die folgende Datei- und Ordnerstruktur:

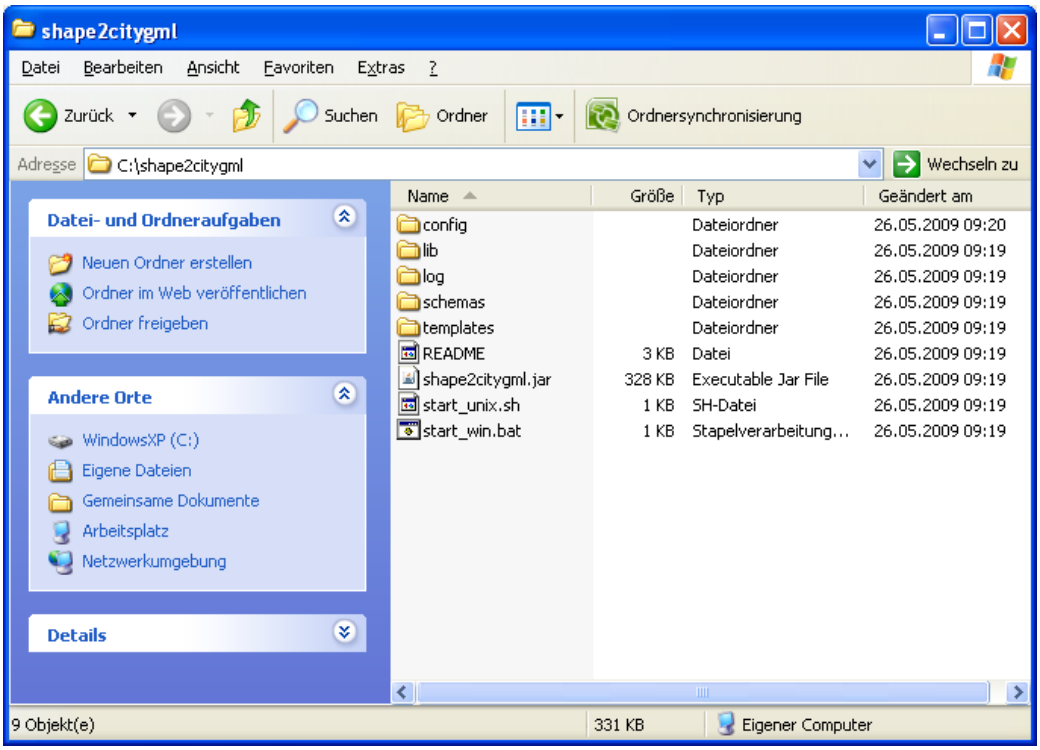


Abbildung 1: Programmdateien im Windows Explorer

Weitere Installationsschritte sind nicht erforderlich. Der Shape2CityGML-Konverter kann somit direkt nach dem Entpackvorgang gestartet werden. Falls erforderlich, können die Programmdateien zu jedem späteren Zeitpunkt an einen beliebigen anderen Ort auf Ihrem Computer kopiert werden. Bitte stellen Sie jedoch sicher, dass hierbei der gesamte Inhalt des Programmverzeichnisses `shape2citygml` inklusive aller Dateien und Unterordner kopiert wird.

Im Folgenden erfolgt ein Überblick über die Inhalte des Programmverzeichnisses:

- **Unterordner `config`**
  Im Unterordner `config` wird die Konfigurationsdatei `config.xml` des Shape2CityGML-Konverters gespeichert. Diese Datei enthält alle benutzerspezifischen Programmeinstellungen wie beispielsweise die Größe und Position des Programmfensters oder die Liste der zuletzt verwendeten Projektdateien. Alle Einstellungen können in der graphischen Benutzeroberfläche der Anwendung vorgenommen werden bzw. dort auf Standardeinstellungen zurückgesetzt werden. Ist die Datei `config.xml` noch nicht vorhanden bzw. wurde sie manuell gelöscht, erstellt der Shape2CityGML-Konverter beim nächsten Programmstart automatisch eine neue Konfigurationsdatei mit Standardeinstellungen.

- **Unterordner `lib`**
  Der Unterorder `lib` enthält alle für die Programmausführung erforderlichen externen Java-Programmbibliotheken. Diese Bibliotheken stammen ausschließlich von Open-Source-Softwareprojekten, deren Lizenz eine freie Verwendung auch in kommerziellen Softwareprodukten erlaubt. Im Einzelnen werden Bibliotheken nachfolgender Open-Source-Projekte eingesetzt:

  - **citygml4j**
    citygml4j ist eine freie Java-Softwarebibliothek, die am Institut für Geodäsie

und Geoinformationstechnik der Technischen Universität Berlin entwickelt wird. citygml4j ermöglicht das Lesen, Schreiben und Verarbeiten von CityGML-Dokumenten. Weitere Informationen zu citygml4j können auf der offiziellen Projektseite im Internet unter http://www.igg.tu-berlin.de/software/ bezogen werden. citygml4j steht unter der GNU Lesser General Public License (LGPL) der Free Software Foundation (FSF) in der Version 3 (http://www.gnu.org/licenses/lgpl-3.0.html).

- **args4j**
  args4j ist eine Open-Source-Bibliothek aus der java.net-Entwicklergemeinschaft. Sie ermöglicht das Parsen von Kommandozeilen-Parametern und wird für das Kommandozeilen-Benutzerinterface des Shape2CityGML-Konverters eingesetzt. Weitere Information zu args4j enthält die Projektseite unter https://args4j.dev.java.net/. args4j ist unter der MIT-Lizenz der Open Source Initiative (OSI) veröffentlicht (http://www.opensource.org/licenses/mit-license.php).
- **GeoTools**
  GeoTools ist eine Sammlung freier Java-Softwarebibliotheken, die umfangreiche Methoden für die Verarbeitung und Manipulation von Geodaten zur Verfügung stellt. Unter anderem implementieren die GeoTools freie Standards und Spezifikationen des Open Geospatial Consortiums (OGC). Das GeoTools-Projekt ist Mitglied in der Open Source Geospatial Foundation (OSGeo, http://www.osgeo.org/). Umfangreiche Informationen zu GeoTools können unter http://geotools.codehaus.org/ abgerufen werden. GeoTools steht unter der GNU Lesser General Public License (LGPL) der Free Software Foundation (FSF) in der Version 2.1 (http://www.gnu.org/licenses/lgpl-2.1.html).
- **named-regexp**
  named-regesp ist eine Open-Source-Bibliothek. Sie ermöglicht das nutzen von benannten Capturing Gruppen in Regulären Ausdrücken. Weitere Information zu named-regexp enthält die Projektseite unter http://code.google.com/p/named-regexp named-repexp ist unter der Apache-Lizenz Version 2.0 veröffentlicht (http://www.apache.org/licenses/LICENSE-2.0).

- **Unterordner `log`**
  Im Unterordner `log` werden die Log-Dateien des Shape2CityGML-Konverters abgelegt, sofern das Schreiben von Log-Dateien in den Benutzereinstellungen aktiviert wurde. Die Log-Dateien sind einfache Textdateien, die alle auftretenden Meldungen während der Programmausführung mitprotokollieren und speichern. Sie können daher für eine weitergehende Analyse nach der Programmausführung verwendet werden.
  Je nach eingestelltem Log-Level können die Log-Dateien in ihrer Dateigröße schnell anwachsen. Der Shape2CityGML-Konverter führt jedoch keine automatisierte Archivierung oder Löschung veralteter Log-Dateien durch. Dies muss durch den Benutzer manuell geschehen. Viele aktuelle Betriebssysteme unterstützen diesen Prozess durch entsprechende Programme und Dienste, welche eine automatisierte Archivierung von Log-Dateien im Hintergrund ermöglichen (sog. Log File Rotation).

- **Unterordner `schemas`**
  Der Unterordner `schemas` enthält die offizielle XML-Schemadatei `xAL.xsd` der eXtensible Address Language (xAL) des OASIS Consortiums in der Version 2.0. Diese XML-Schemadatei wird zur Validierung von xAL-Adressvorlagen benötigt, die vom Benutzer in den Shape2CityGML-Konvertierungsprozess eingebracht werden können (für ausführliche Informationen siehe Kapitel 4). Das xAL-Adressschema wird zur

Kodierung von Adressinformation in CityGML verwendet.

- **Unterordner `templates`**
  Eine Beispiel xAL-Adressvorlagen `PotsdamAddress.xml` wird im Unterordner `templates` mitgeliefert. Sie kann als Ausgangspunkt für projektspezifische Adressvorlagen verwendet oder auch unverändert in den Shape2CityGML-Konverter geladen werden.

- **Datei `README`**
  Die README-Datei enthält grundlegende Informationen über den Shape2CityGML-Konverter. Als einfache Textdatei kann sie mit jedem Texteditor eingelesen und angezeigt werden.

- **Datei `shape2citygml.jar`**
  `shape2citygml.jar` stellt die eigentliche ausführbare Programmdatei des Shape2CityGML-Konverters dar.

- **Dateien `start_win.bat` und `start_unix.sh`**
  `start_win.bat` und `start_unix.sh` sind ausführbare Dateien zum Starten des Shape2CityGML-Konverters. Sie können unverändert im Auslieferungszustand zum Programmstart verwendet werden, erlauben jedoch auch Anpassungen an systemspezifische Besonderheiten durch den Benutzer. Entsprechend ihrer Namensgebung wird die Datei `start_win.bat` bei Vertretern der Microsoft Windows-Familie (z.B. WindowsXP, Windows Vista, Windows 7) eingesetzt, während `start_unix.sh` für die Verwendung in UNIX/Linux-Derivaten vorgesehen ist (z.B. auch MacOS X). Bitte beachten Sie die Hinweise zum Starten des Shape2CityGML-Konverters im nachfolgenden Abschnitt.

1.4 Starten des Shape2CityGML-Konverters

Nach dem Entpacken der Programmdateien in ein beliebiges Verzeichnis auf Ihrem Computer stehen grundsätzlich drei Varianten zum Ausführen des Shape2CityGML-Konverters zur Verfügung.

1.4.1 Empfohlene Variante: Verwendung der mitgelieferten Starter-Dateien

Die **empfohlene Variante** zum Starten der graphischen Benutzeroberfläche des Shape2CityGML-Konverters ist die Verwendung der mitgelieferten Starter-Dateien `start_win.bat` bzw. `start_unix.sh`, die sich im extrahierten Programmverzeichnis `shape2citygml` befinden. In Windows-Umgebungen genügt hierzu bereits ein einfacher Doppelklick auf die Datei `start_win.bat`. Um die Starter-Dateien nicht nur aus dem Programmverzeichnis heraus verwenden zu können, können Verknüpfungen mit den Dateien an beliebigen anderen Orten des Computers angelegt werden. Beispielsweise kann so auf Windows-Systemen eine Verknüpfung mit der Datei `start_win.bat` auf dem Desktop oder aber in der Schnellstartleiste erstellt werden. Nicht ausreichend ist es jedoch, die Datei lediglich an diese Orte zu kopieren. In diesem Fall sind wichtige Pfadinformationen nicht korrekt gesetzt und die Anwendung kann nicht gestartet werden.

Die Starter-Dateien sind im Auslieferungszustand auf den meisten Systemen ohne Änderungen verwendbar. Sie können jedoch bei Bedarf an systemspezifische Besonderheiten angepasst werden. Dies sollte nur durch erfahrene Benutzer erfolgen. Eine mögliche Anpassung betrifft die Änderung des verfügbaren Hauptspeichers für die Java-Laufzeitumgebung. Bei der Verarbeitung großer Shape- bzw. CityGML-Dateien durch den Shape2CityGML-Konverter steigt die Speicherauslastung der Anwendung eventuell stark an. Sind die Grenzen des zugewiesenen Hauptspeichers für die Java-Laufzeitumgebung in einem solchen Fall zu restriktiv gesetzt, wird die Anwendung wegen Speichermangels mit

einer Fehlermeldung abbrechen. Im Folgenden wird die Anpassung des zugewiesenen Hauptspeichers am Beispiel der Datei `start_win.bat` dargestellt (alle Schritte sind analog auf die Datei `start_unix.sh` übertragbar).

Zunächst muss die Starter-Datei `start_win.bat` in einem beliebigen Texteditor geöffnet werden (siehe Abbildung 2).

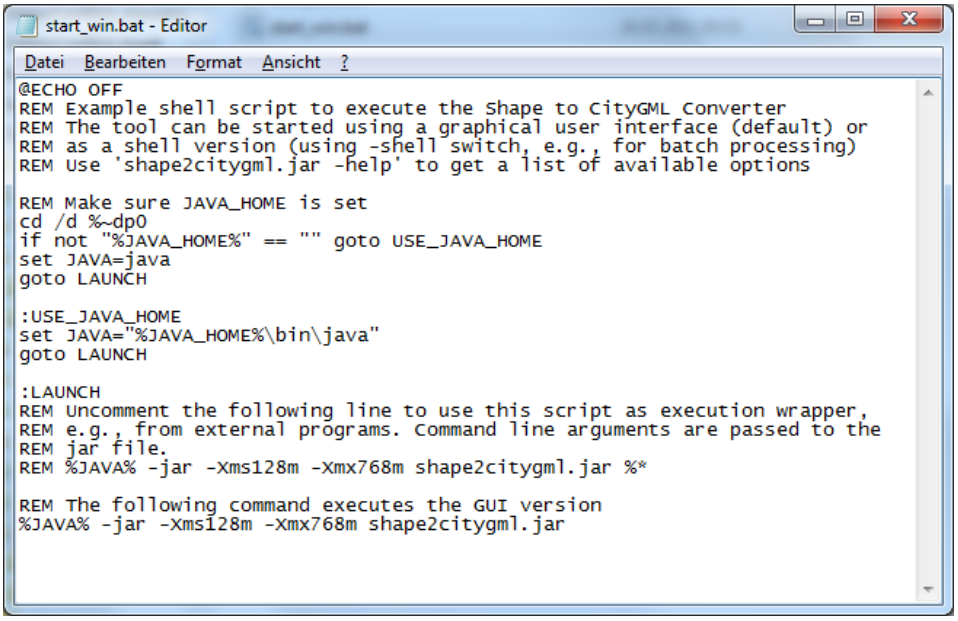


Abbildung 2: start_win.bat

Der eigentliche Befehl zum Starten des Shape2CityGML-Konverters steht in Zeile 24 der Datei `start_win.bat`:

```
%JAVA% -jar -Xms128m -Xmx768m shape2citygml.jar
```

In den vorausgehenden Zeilen 1 – 23 des Skripts wird die installierte Java-Laufzeitumgebung ermittelt, um die Java Virtual Machine (JVM) korrekt starten zu können. Die JVM ist der Teil der Java-Laufzeitumgebung, welcher für die Ausführung von Java-Programmen verantwortlich ist. Das zu startende Java-Programm wird in Zeile 24 durch die ausführbare Programmdatei `shape2citygml.jar` des Shape2CityGML-Konverters spezifiziert. Dass es sich hierbei um eine ausführbare Programmdatei handelt (erkennbar an der Dateiendung .jar) wird der JVM über den Schalter `–jar` mitgeteilt.

Der verfügbare Hauptspeicher wird der JVM durch die beiden Laufzeitparameter `–Xms<size>` bzw. `–Xmx<size>` übergeben. Der erste Parameter `–Xms<size>` bestimmt hierbei die Mindestgröße an Hauptspeicher, der von der JVM belegt werden kann, während `–Xmx<size>` die maximale Speicherauslastung festlegt. Im Auslieferungszustand der Starter-Dateien sind diese Werte auf 128MB bzw. 768MB festgelegt, was in den meisten Anwendungsszenarien des Shape2CityGML-Konvertes bereits ausreichend ist.

Um beispielsweise die maximale Speicherauslastung auf 1GB zu erhöhen, muss die Zeile 24 der Starter-Datei `start_win.bat` wie folgt abgeändert werden:

```
%JAVA% -jar -Xms128m –Xmx1024m shape2citygml.jar
```

Grundsätzlich können auch weitere Laufzeitparameter der JVM an dieser Stelle in das Skript eingefügt werden, wenn dies zum Starten des Shape2CityGML-Konverters aufgrund systemspezifischer Besonderheiten erforderlich ist. Weiterhin ist es möglich,

Kommandozeilen-Optionen des Shape2CityGML-Konverters selbst an den Start-Befehl anzuhängen (hinter die Angabe des zu startenden Java-Programms `shape2citygml.jar`). Letzteres wird im Kapitel 5 über die Bedienung des Kommandozeilen-Benutzerinterfaces vertieft dargestellt.

1.4.2 Starten über die Kommandozeile

Der Shape2CityGML-Konverter kann weiterhin direkt über die Kommandozeile ausgeführt werden. In Windows-Umgebungen muss hierzu eine MS-DOS-Eingabeaufforderung gestartet werden. In UNIX/Linux-Derivaten stehen entsprechende Kommandozeileninterpreter bereit (z.B. bash, csh, oder zsh). Für die grundlegende Bedienung der betriebssystemspezifischen Kommandozeileninterpreter wird an dieser Stelle auf die entsprechenden Dokumentationen verwiesen.

Nach dem Öffnen der Kommandozeilen-Umgebung muss zunächst in das Programmverzeichnis des Shape2CityGML-Konverters gewechselt werden. In diesem kann die Anwendung durch den nachfolgenden Befehl gestartet werden. Die Angabe `[-options]` am Ende des Befehls soll verdeutlichen, dass optional weitere Kommandozeilen-Parameter spezifiziert werden können.

```
java -jar shape2citygml.jar [-options]
```

Der angegebene Befehl setzt voraus, dass das Kommando `java` (bzw. `java.exe`) vom Kommandozeileninterpreter verstanden werden kann. Dies kann beispielsweise durch die Aufnahme von `java.exe` in die PATH-Umgebungsvariable erreicht werden. Grundsätzlich wird dies bereits während der Installation der JRE automatisch sichergestellt.

Das dargestellte Startkommando entspricht im Wesentlichen demjenigen, welches zum Start des Programms in den Starter-Dateien verwendet wird. Es wird daher dringend empfohlen, auch beim Start über die Kommandozeile die Hauptspeichergrenzen der JVM über die Laufzeitparameter `–Xms<size>` bzw. `–Xmx<size>` entsprechend den Ausführungen im vorangegangenen Abschnitt festzulegen. Unterbleibt diese Angabe, verwendet die JVM vordefinierte Standardeinstellungen. Diese variieren abhängig vom Betriebssystem und der verwendeten Version der JRE, sind aber in jedem Fall sehr restriktiv und können daher schnell zu einem Abbruch der Anwendung wegen Speichermangels führen.

Eine Übersicht der verfügbaren Kommandozeilen-Optionen zur Beeinflussung der Programmausführung des Shape2CityGML-Konverters kann über folgenden Befehl abgerufen werden:

```
java -jar shape2citygml.jar -help
```

Die Abbildung 3 zeigt das Ergebnis dieses Befehls beispielhaft innerhalb einer MS-DOS-Eingabeaufforderung:

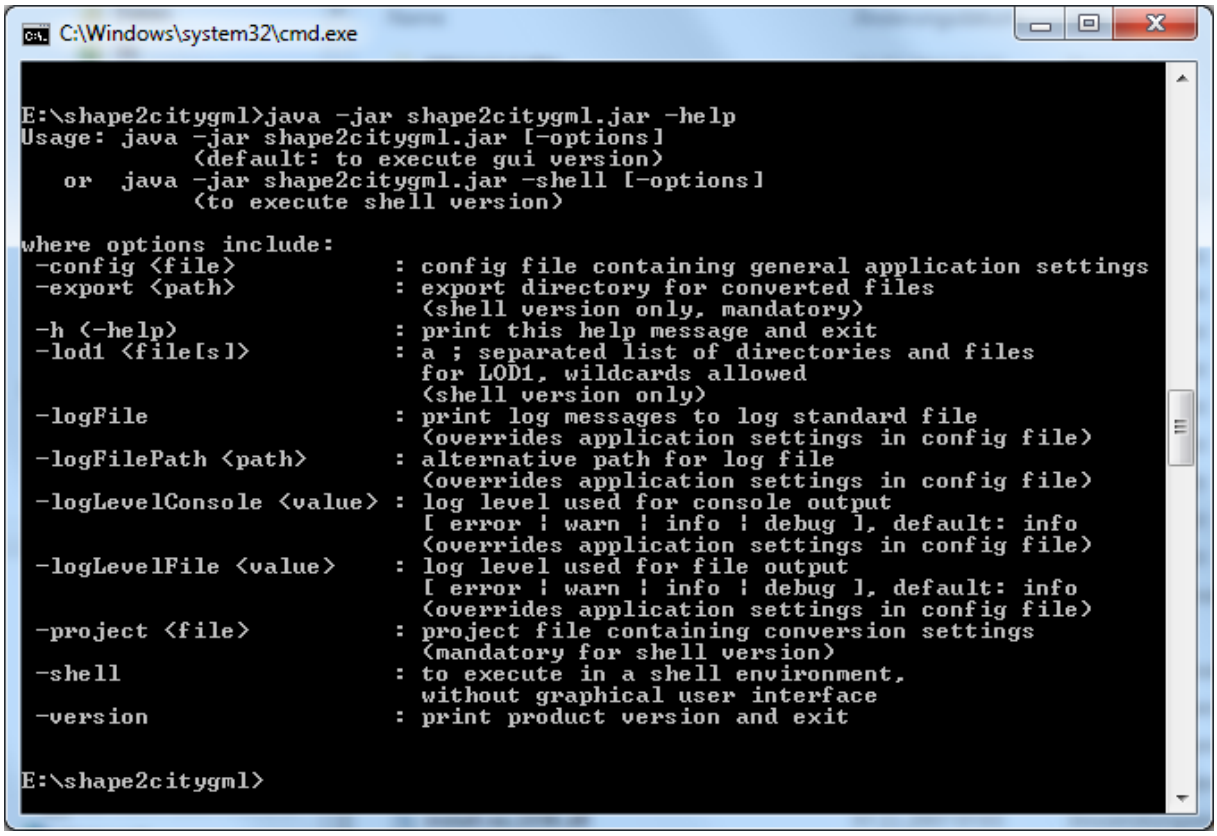


Abbildung 3: Kommandozeile help Befehl

Der Shape2CityGML-Konverter kann aus der Kommandozeile heraus wahlweise mit graphischer Benutzeroberfläche (Graphical User Interface, GUI) gestartet werden oder allein über sein Kommandozeilen-Interface (Command-Line Interface, CLI) gesteuert werden. Während das Starten der GUI die Standardeinstellung ist, muss die Benutzung des CLI mit der Option `-shell` erst angegeben werden. Wie die Abbildung 3 zeigt, sind einige der Kommandozeilen-Optionen nur für das Kommandozeilen-Interface verfügbar (gekennzeichnet mit dem Zusatz „`shell version only`“). Erfolgt ihre Angabe dennoch beim Start der GUI, werden sie vom Shape2CityGML-Konverter ignoriert und entfalten daher keine Wirkung auf die Programmausführung.

Die Verwendung des Kommandozeilen-Interfaces ermöglicht eine echte Stapel- bzw. Batchverarbeitung von Eingangsdatensätzen. Das Starten des Konvertierungsprozesses erfordert nicht mehr die Interaktion eines Benutzers mit der GUI, sondern kann als Befehl in beliebig komplexe Batch-Dateien eingebunden werden, welche sequentiell vom Betriebssystem verarbeitet werden (in UNIX/Linux werden Batch-Dateien als Shellskripte bezeichnet). Aktuelle Betriebssysteme erlauben die nicht-interaktive Ausführung von Batch-Dateien zu beliebigen Zeitpunkten und stellen entsprechende Scheduler-Dienste bereit. Schließlich erlaubt die Steuerung des Shape2CityGML-Konverters über das CLI auch die Einbindung der kompletten Funktionalität in Dritt-Anwendungen, z.B. die Steuerung des Programms über eine Web-Anwendung.

Das CLI des Shape2CityGML-Konverters wird eingehend in Kapitel 5 vorgestellt.

1.4.3 Verwendung der ausführbaren Programmdatei `shape2citygml.jar`

Die dritte Variante zum Starten des Shape2CityGML-Konverters besteht in der Verwendung der ausführbaren Programmdatei `shape2citygml.jar`. Die Datei befindet sich direkt im

Programmverzeichnis `shape2citygml` und kann beispielsweise in Windows-Umgebungen durch einen einfachen Doppelklick gestartet werden.

Allerdings wird davon abgeraten, den Shape2CityGML-Konverter auf diese Weise zu starten. Wie bereits in den vorangegangenen Abschnitten erläutert, startet die JVM ohne spezielle Angabe von Hauptspeichergrenzen mit sehr restriktiven Standardeinstellungen. Diese können bei dieser dritten Startvariante sehr schnell zu Programmabbrüchen wegen Speichermangels führen.

1.5 Deinstallation

Das Entfernen des Shape2CityGML-Konverters von Ihrem Computer erfordert keine spezielle Deinstallationsroutine. Es genügt vielmehr, das komplette Programmverzeichnis zu löschen. Eventuell sollten zuvor bestehende Log-Dateien im Unterorder `log` des Programmverzeichnisses gesichert werden, um bereits bearbeitete Projekte dokumentieren zu können. Darüber hinaus können die benutzerspezifischen Programmeinstellungen im Unterordner `config` gesichert werden, um sie für eine zukünftige Neuinstallation wiederverwenden zu können.

1.6 Java-Quelltexte des Shape2CityGML-Konverters

Der Quellcode des Shape2CityGML-Konverters liegt – wie auch die Programmdateien - als gepacktes .zip-Archiv vor. Nach dem Entpacken der Archivdatei „shape2citygml-src.zip“ enthält das Hauptverzeichnis `shape2citygml-src` die nachfolgende Datei- und Ordnerstruktur:

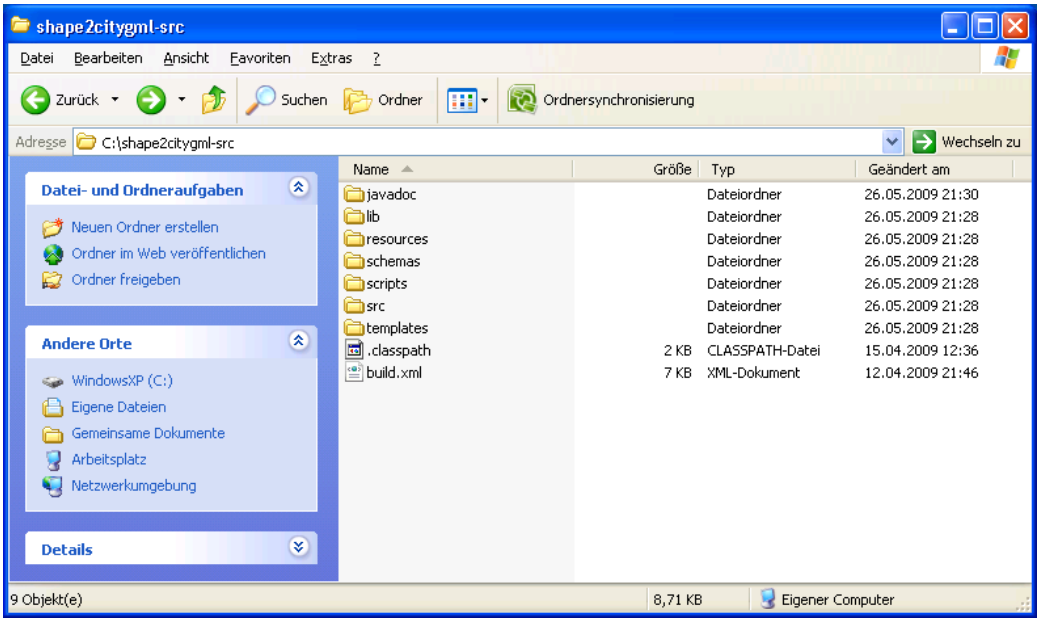


Abbildung 4: Java-Quellcode des Shape2CityGML-Konverters

Die einzelnen Dateien und Unterordner enthalten folgende Inhalte:

- **Unterordner `javadoc`**
  Der Unterordner `javadoc` enthält die gesamte Dokumentation des Quellcodes in Form von HTML-Dokumentationsdateien, welche direkt aus den Java-Quelltexten abgeleitet wurden. Die Generierung der HTML-Seiten erfolgte mit dem Standard-Software-Dokumentationswerkzeug Javadoc. Zum Öffnen der Dokumentationsseiten muss die Datei `index.html` im Unterordner `javadoc` in einem beliebigen Internetbrowser geladen werden. Die Abbildung 5 zeigt einen Ausschnitt der Hauptseite der Quelltext-Dokumentation.

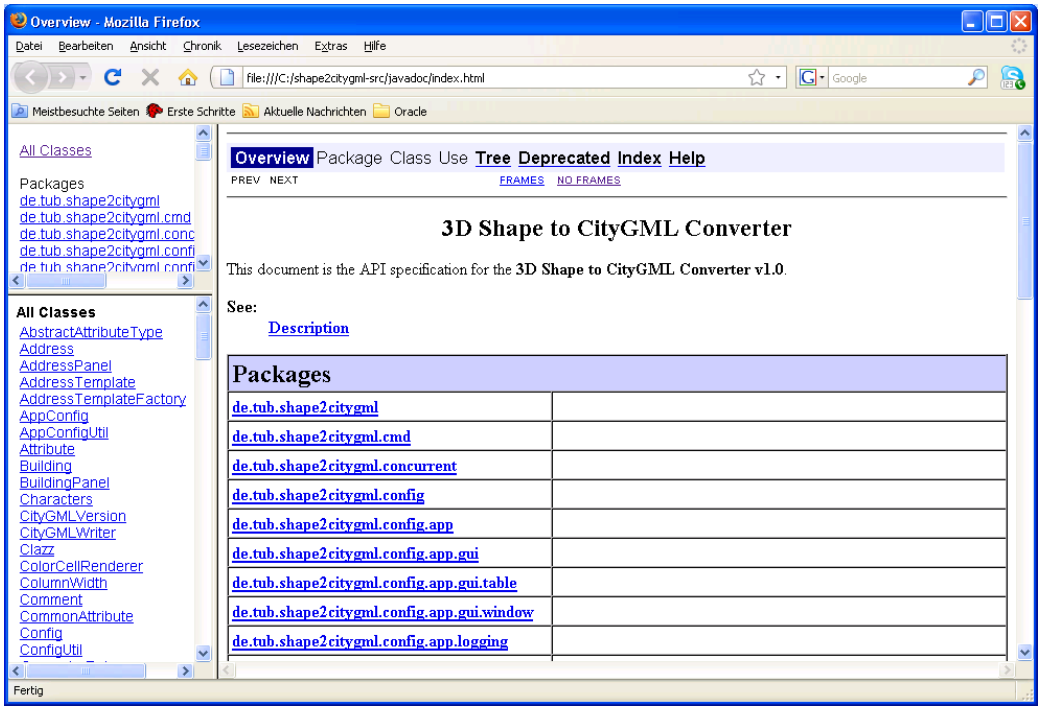


Abbildung 5: Quellcode Dokumentation

- **Unterordner `lib`**
  Wie bereits der entsprechende Unterordner des Programmverzeichnis enthält auch hier der Unterordner `lib` alle erforderlichen externen Java-Programmbibliotheken. Insofern kann auf die Ausführungen in Kapitel 1.3 verwiesen werden.

- **Unterordner `resources`**
  Der Unterordner `resources` enthält Ressource-Dateien, die zusätzlich zu den eigentlichen Java-Quelltexten für die Erstellung des Programmpakets bzw. für dessen Dokumentation erforderlich sind.

- **Unterordner `schemas`**
  Wie bereits der entsprechende Unterordner des Programmverzeichnis enthält auch hier der Unterordner `schemas` die XML-Schemadatei des xAL-Adressstandards. Insofern kann auf die Ausführungen in Kapitel 1.3 verwiesen werden.

- **Unterordner `scripts`**
  Im Unterordner `scripts` finden sich die Vorlagen für die beiden Starter-Dateien `start_win.bat` und `start_unix.sh`.

- **Unterordner `src`**
  Der Unterordner `src` enthält alle Java-Quelltextdateien des Shape2CityGML-Konverters. Die Klasse `de.tub.shape2citygml.Shape2CityGML` ist die zentrale Klasse des Shape2CityGML-Konverters. Sie ist für den Programmstart der Anwendung verantwortlich und steuert alle weiteren Aktionen.

- **Unterordner `templates`**
  Wie bereits der entsprechende Unterordner des Programmverzeichnis enthält auch hier der Unterordner `templates` zwei beispielhafte xAL-Adressvorlagen. Insofern

kann auf die Ausführungen in Kapitel 1.3 verwiesen werden.

- **Datei `.classpath`**
  Die Datei `.classpath` selbst ist kein Bestandteil des Quellcodes. Sie stellt vielmehr eine Informationsdatei dar, die von der freien Programmierumgebung Eclipse (http://www.eclipse.org/) ausgewertet werden kann. Damit wird der Import der Quelltextdateien in ein neues Java-Projekt innerhalb von Eclipse deutlich vereinfacht. Die `.classpath`-Datei sorgt beispielsweise dafür, dass die externen Programmbibliotheken automatisch eingebunden werden. Für die aktuelle Version 3.6 (Helios) von Eclipse erfordert das Importieren der Quelltextdateien zunächst das Anlegen eines neuen Java-Projektes über das Hauptmenü „File → New → Java Project“. Im sich anschließenden Dialog muss lediglich ein Projektname ausgewählt sowie das Hauptverzeichnis des extrahierten .zip-Archivs als Quellverzeichnis angegeben werden (siehe Abbildung 6).

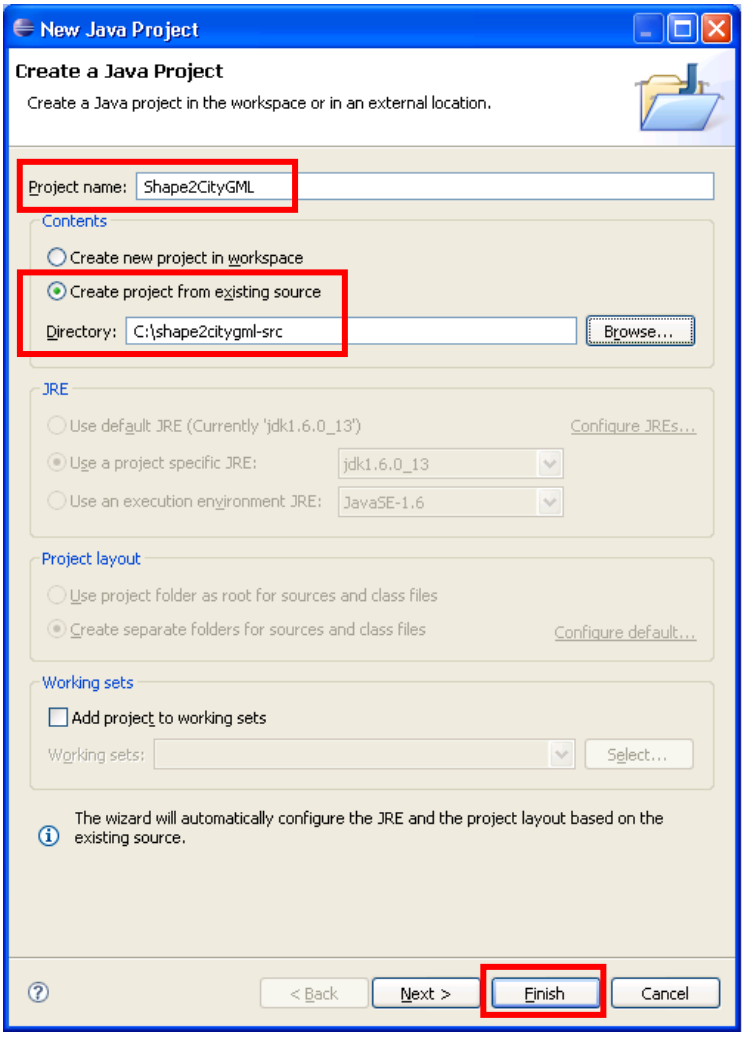


Abbildung 6: Eclipse Dialog: Neues Java Projekt

- **Datei `build.xml`**
  Die Datei `build.xml` ist ein Ant-Skript, welches den automatisierten Erstellungsvorgang (Build) des Shape2CityGML-Konverters aus den Quelltexten ermöglicht. Das Ergebnis dieses Erstellungsvorgangs ist die Datei- und Ordnerstruktur des Programmverzeichnisses, wie sie in Kapitel 1.3 vorgestellt wurde. Die `build.xml`-Datei wird von der Open-Source-Software Apache Ant der Apache Software Foundation interpretiert und ausgeführt. Dafür muss entweder Ant

auf dem Computer installiert werden (weitere Information hierzu siehe unter http://ant.apache.org/). Alternativ kann der Erstellungsprozess auch sehr einfach über das Programmierwerkzeug Eclipse gestartet werden. Neben den Regeln für die Erstellung der ausführbaren Dateien des Shape2CityGML-Konverters enthält die `build.xml`-Datei auch alle notwendigen Einstellungen zur automatisierten Ableitung der Javadoc-Dokumentationsseiten aus den Quelltexten.

2 Aufbau der graphischen Benutzeroberfläche

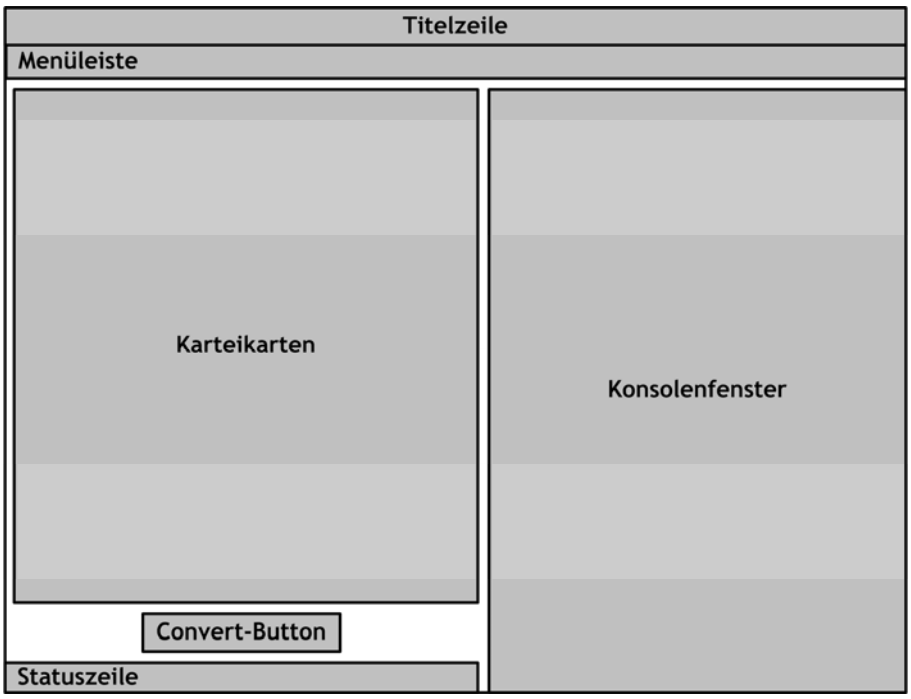


Abbildung 7: Graphische Benutzeroberfläche

Die graphische Benutzeroberfläche ist in sechs Bereiche aufgeteilt (Titelzeile, Menüleiste, Karteikarten, Convert-Button, Statuszeile, Konsolenfenster), die in den folgenden Unterkapiteln beschrieben werden.

2.1 Titelzeile

In der Titelzeile wird neben den üblichen Elementen (Applikationsicon, Name des Tools, Symbole zur Fensterminimierung, -maximierung und-schließung) auch der Name der aktuell geöffneten Projektdatei (.shp2gml-Datei) inklusive Dateipfad eingeblendet. In der Projektdatei werden alle projektspezifischen Einstellungen, die in der GUI vorgenommen werden, gespeichert. Dies sind insbesondere die Zuordnungen von Shape-Attributen der Eingangsdaten auf CityGML-Attribute. Erscheint hinter dem Dateinamen ein Sternchen, ist das ein Zeichen dafür, dass die Datei seit dem letzten Speichervorgang verändert wurde.

2.2 Menüleiste

In der Menüleiste sind 3 Menüs zu erkennen: File, Project und Help.

- **File** beinhaltet die Möglichkeit, das Programm zu schließen. Wurde die Projektdatei seit dem letzten Speichervorgang verändert, erscheint vor dem Schließen eine absichernde Anfrage, ob die Datei zuvor gespeichert werden soll.
- **Project** beinhaltet Möglichkeiten zum Öffnen bestehender Projektdateien (Open project…), zum Speichern eines Projektes unter dem bereits vergebenen Namen (Save project), unter einem alternativen Namen (Save project as…), zum Speichern der entsprechenden XSD Schemadatei (Save project XSD as…) und zum Öffnen zuletzt bearbeiteter Projektdateien (Last used projects).

- **Help** beinhaltet Hinweise auf die Autoren (siehe Abbildung 8), die Copyrightbesitzer und die verwendeten Bibliotheken.

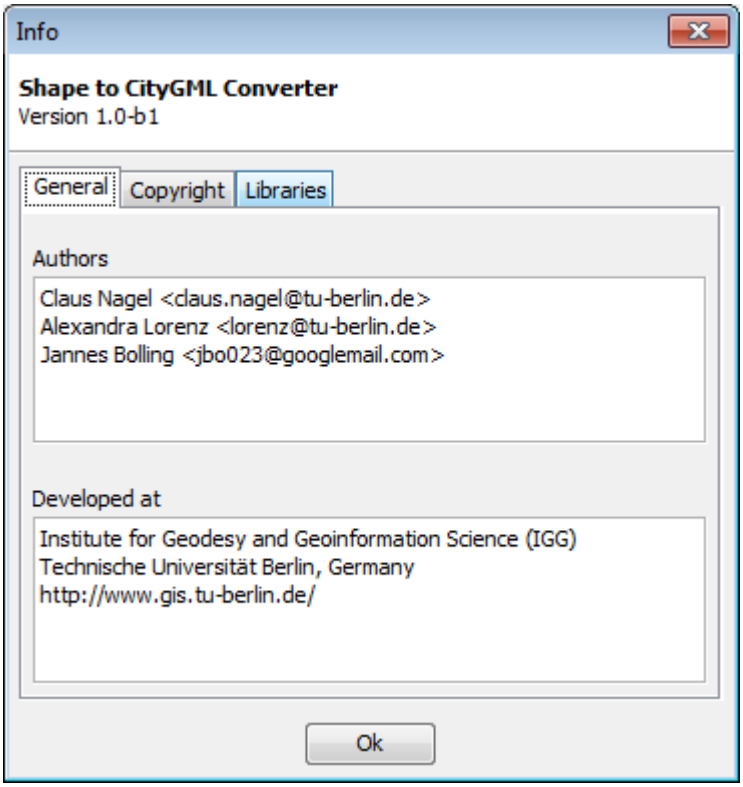


Abbildung 8: Info Dialog

2.3 Karteikarten

Die Hauptfunktionalität des Tools ist zu Zwecken der Übersichtlichkeit in Karteikarten eingeteilt. Die 3 Bereiche umfassen folgende Funktionalitäten, die in den nachfolgenden Unterkapiteln im Detail erläutert werden:

- **File**: Auswahl von Eingabedateien, Angabe eines Ausgabeverzeichnisses
- **Address**: Angabe einer Adressvorlage, Angabe einer CSV Mapping Datei für das Auflösen von Straßenschlüsseln.
- **Preferences**: Voreinstellungen für den Export und das Logging

2.3.1 Karteikarte „File“

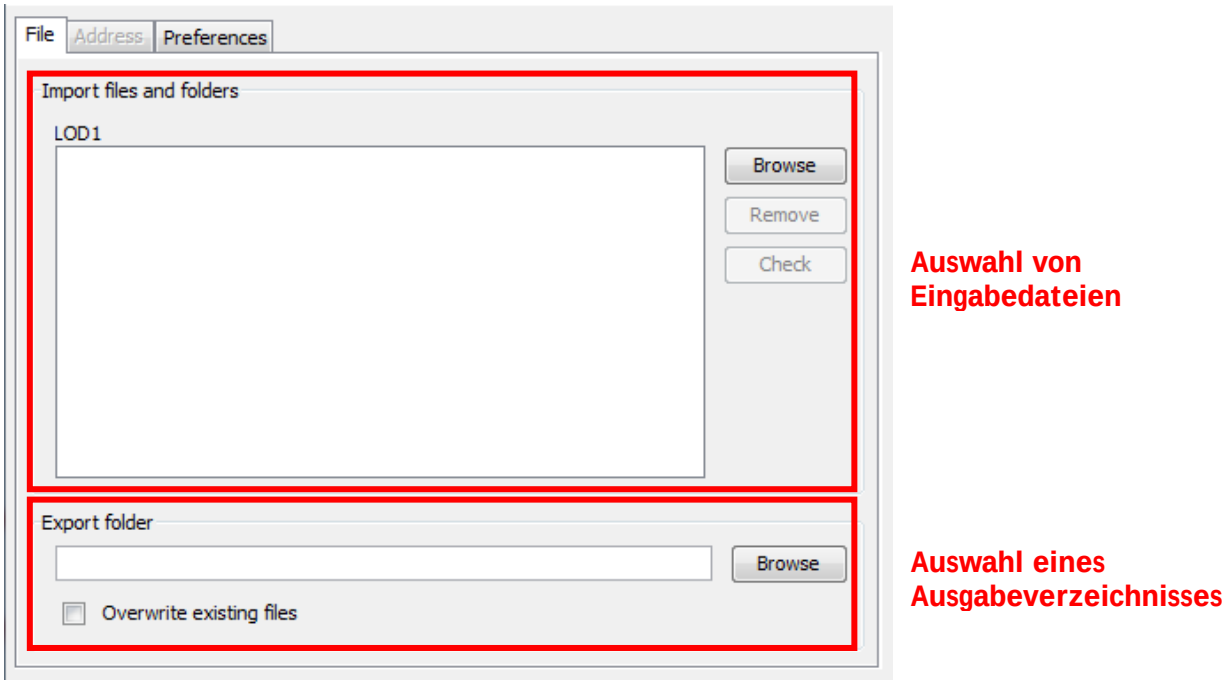


Abbildung 9: Karteikarte File

Die Karteikarte „File“ besteht aus zwei Bereichen:

- Auswahl von Eingabedateien,
- Auswahl eines Ausgabeverzeichnisses.

Die Auswahl von Dateien erfolgt in dem Bereich LOD1. Es können einzelne Dateien oder ganze Ordner ausgewählt werden (Mehrfachauswahl möglich). Bei der Auswahl eines Ordners werden alle Shapefiles in diesem Ordner sowie rekursiv in allen Unterordnern verwendet. Folgende Methoden für die Auswahl sind realisiert:

- Browse-Button, der ein programminternes Explorerfenster für die Datei-Navigation öffnet
- Drag & Drop Funktion, wodurch Dateien oder Ordner aus einem beliebigen Explorerfenster in die Eingabefelder gezogen werden können (per Drag-and-Drop). Grundsätzlich werden bei Drag & Drop alle Einträge des entsprechenden Eingabefelds durch die neue Dateiliste ersetzt. Wird allerdings während des Drag & Drop zusätzlich die Strg-Taste gedrückt, werden die bestehenden Einträge im Eingabefeld um die neue Dateiliste erweitert (gilt für Windows-Systeme, bei Verwendung anderer Betriebssysteme muss eventuell eine abweichende Tastenkombination verwendet werden).

Markierte Dateien im Eingabefeld können über „Remove“ wieder aus der Dateiliste entfernt werden. „Check“ wertet die Attribute aller Dateien im entsprechenden Eingabefeld aus. In der Konsole wird eine Statistik über die gefundenen Attribute ausgegeben.
Neben der Auswahl von Eingabedateien ist die Auswahl eines Ausgabeverzeichnisses Grundvoraussetzung für den erfolgreichen Konvertierungsvorgang. Die Checkbox „Overwrite existing files“ stellt sicher, dass nicht unabsichtlich bereits existierende Dateien überschrieben werden. Die Namen der Ausgabedateien richten sich direkt nach den Namen der Eingabedateien – sollen also dieselben Eingabedateien mit anderen Attributzuordnungen bzw. anderen Projektsettings nochmals konvertiert werden, müssen entweder die bereits herausgeschriebenen Dateien überschrieben oder (um das zu vermeiden) ein alternatives Ausgabeverzeichnis angegeben werden. Wenn die Checkbox

„Overwrite existing files“ nicht aktiviert ist werden schon vorhandene Dateien nicht überschrieben; das Programm bricht in diesem Fall ab.

2.3.2 Karteikarte „Address“

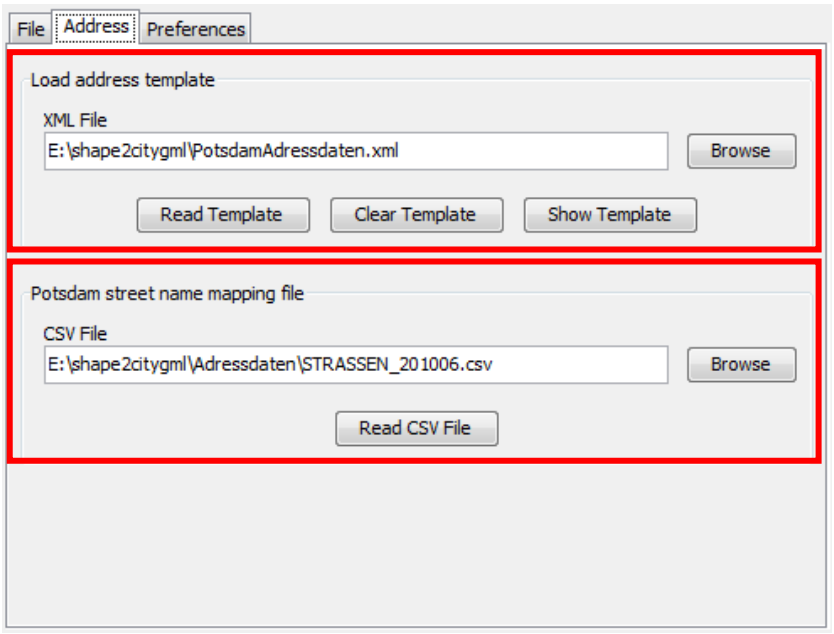


Abbildung 10: Karteikarte Address

In Kapitel 2.3.2 wurde bereits die graphische Benutzeroberfläche, die der Verwaltung von Adressvorlagen dient, vorgestellt.

Die Karteikarte „Address“ wird aktiv sobald man über den Menüpunkt „Open project…“ eine gültige Projektdatei geladen hat. Die Karteikarte „Address“ ist in zwei Bereiche unterteilt. Der obere Bereich „Load address template“ ermöglicht das Laden einer beliebigen xAL-Adressvorlage (Template). Mit dem „Browse“-Button bzw. per Drag & Drop kann ein Template ausgewählt werden. Über den Button „Read template“ wird die Adressvorlage vom Shape2CityGML-Konverter geladen und überprüft. Der Button „Show Template“ öffnet ein Pop-up Fenster in dem die Adressvorlage angezeigt wird.
Ein Template ermöglicht die Einbettung von vorstrukturierte Adressinformationen in die CityGML-Ausgabedatei. In den Adresstemplates werden so genannte „Tokens“ eingebettet, welche dann durch entsprechende Daten aus der Shape Datei ersetzt werden (siehe Kapitel 4.2Erstellen und Verwenden von Adressvorlagen (Templates)).
Der zweite Bereich ermöglicht das angeben einer Zuordnungs Datei um für Straßenschlüssel den entsprechenden Straßennamen herauszufinden. Als Eingangsdaten benötigt man hier eine Semikolon oder Komma getrennte CSV Datei mit den zwei Spalten Straßenschlüssel und Straßennamen. (siehe Kapitel 4.5)

2.3.3 Karteikarte „Preferences“

Die Karteikarte „Preferences“ gewährleistet die Angabe von projekt- und benutzerspezifischen Voreinstellungen. Im Baum auf der linken Seite kann zwischen Export und Logging gewählt werden. Export umfasst projektspezifische Einstellungen zum Dateiexport. Logging umfasst benutzerspezifische Einstellungen zum Logging von Programminformationen, welche in der Konsole und zusätzlich in Log-Dateien ausgegeben werden können. Der Unterschied zwischen Projekt- und benutzerspezifische besteht darin, dass projektspezifische Einstellungen immer nur für ein spezifisches Projekt gelten und daher beim Aufrufen eines neuen Projektes überschrieben werden. Benutzerspezifische Eingaben hingegen bleiben über Projektwechsel hinweg bestehen.

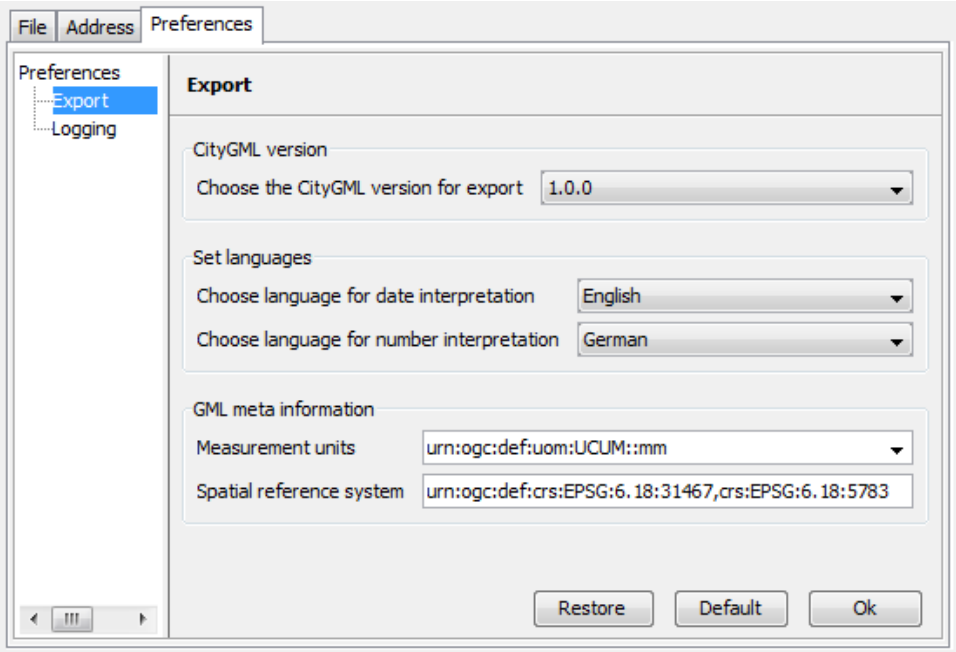


Abbildung 11: Karteikarte Preferences Export

Exporteinstellungen umfassen

- Wahl der auszugebenden CityGML-Version (0.4.0 oder 1.0.0)
- Wahl der Sprache für Datums- und Nummerninterpretation für das Auslesen der Attribut-Daten aus der Shape Datei – deutsche/englische Datumsinterpretation richtet sich an Datumsformate, in denen Monatsnamen in textueller Form vorkommen; deutsche/englische Nummerninterpretation gibt Auskunft darüber, wie Trennzeichen (Punkt oder Komma) zu interpretieren sind.
- Angabe von GML-Metainformationen (Maßeinheit und räumliches Referenzsystem) – die Angabe erfolgt über die Namensräume *urn*:*ogc*:*def*:*uom* bzw. *urn*:*ogc*:*def*:*crs*; da die Wahl der Maßeinheit mit hoher Wahrscheinlichkeit eine metrische Einheit betreffen wird, sind in der Combobox bereits Vorschläge für Standardangaben enthalten.

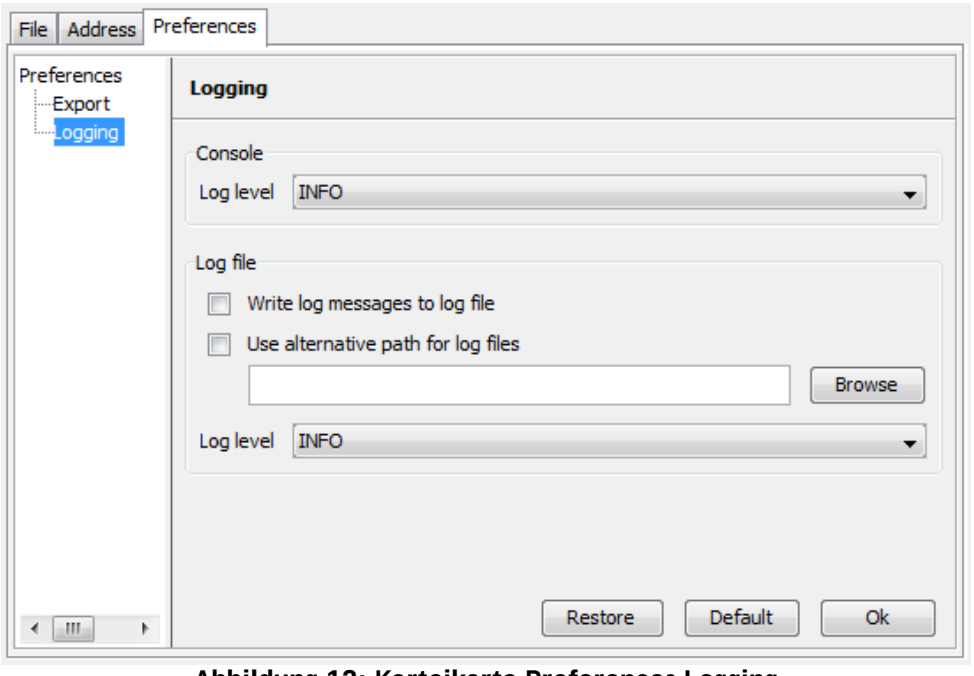


Abbildung 12: Karteikarte Preferences Logging

Loggingeinstellungen umfassen

- Wahl des Logging-Niveaus in der Konsole – je restriktiver das Niveau gewählt wird, umso weniger Programminformationen erscheinen in der Konsole. Es empfiehlt sich, ein mittleres Niveau (z.B. WARN oder INFO) auszuwählen. Am restriktivsten ist ERROR. Bei DEBUG werden alle Meldungen ausgegeben. Achtung: die Konsole wird nach 10.000 Meldungen automatisch gelöscht, um einen Programmabbruch wegen Hauptspeichermangels vorzubeugen. Die Wahl des DEBUG-Modus für die Konsole ist daher im Normalfall nicht empfehlenswert.
- Ausgabe eines zusätzlichen Logfiles. Für die Ausgabe in einem externen Logfile kann ein spezifischer Pfad angegeben werden – der Standardpfad ist der Unterordner *log* im Programmverzeichnis. Für das Logfile ist ein eigenes Logging-Niveau wählbar, das durchaus weniger restriktiv gewählt werden kann als das Niveau in der Konsole (z.B. INFO oder DEBUG).

Um die getroffenen Einstellungen zu speichern und damit zu aktivieren, ist es notwendig vor dem Wechseln des Panels diese mit „Ok“ zu bestätigen. Wird das Panel ohne Bestätigung der Eingaben gewechselt, erscheint eine Dialogbox, die nachfragt, ob die Einstellungen übernommen werden sollen. Alternativ zum Bestätigen mittels „Ok“ können die Einstellungen auch auf die letzte Eingabevariante („Restore“) oder auf die Standartwerte („Default“) zurückgesetzt werden.

2.4 Convert-Button

Bei Betätigung des Convert-Buttons wird der Konvertiervorgang gestartet. Wenn eine Projektdatei geladen wurde fängt der Konvertierungsvorgang an. Die entstehenden CityGML-Ausgabedateien werden in das angegebene Ausgabeverzeichnis gespeichert. Sind keine Eingabedateien oder kein Ausgabeverzeichnis ausgewählt, wird der User zur Angabe der entsprechenden Informationen aufgefordert.

2.5 Statuszeile

In der Statuszeile wird der aktuelle Status des Programms angezeigt. „Ready“ bedeutet, dass gerade keine Aktionen (wie z.B. die Konvertierung der Eingabedateien) ausgeführt werden.

2.6 Konsolenfenster

In der Konsole werden alle wesentlichen Informationen zum Programmablauf ausgegeben. Das Logging umfasst vier Detailstufen: ERROR, WARN, INFO und DEBUG. Je nach Einstellung im Preferences-Panel werden Informationen auf einem bestimmten Niveau ausgegeben. Ist des gewählte Niveau beispielsweise INFO, werden alle Meldungen des Typs ERROR, WARN und INFO ausgegeben. Bei dem Niveau ERROR werden nur Meldungen des Typs ERROR ausgegeben.

3 Auswerten der Eingangsdaten

Die Hauptfunktionalität des Shape2CityGML-Konverters besteht in der Überführung der in den eingelesenen Shape-Dateien vorhandenen Attribute in korrespondierende CityGML-Elemente sowie das Erstellen von LOD1 Gebäudegeometrien über Grundrisse und Höhendaten. Die entsprechenden Zuordnungen und Einstellungen werden in einer projekt.shp2gml Datei definiert. Die Erstellung dieser Projektdatei ist nicht Gegenstand dieser Software. Üblicherweise wird eine Projektdatei mitgeliefert.

3.1 Einlesen von Eingabedateien

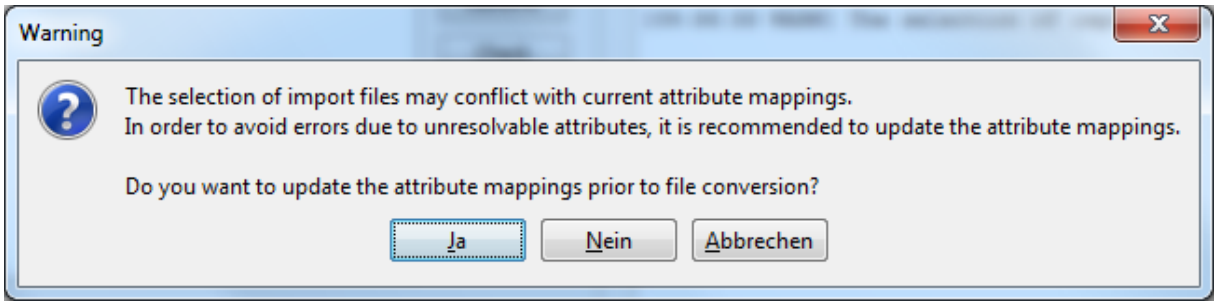
Beim Einlesen der angegebenen Shape Dateien wird geprüft, ob die Attribute der Shape-Datei mit den in der Projekt Datei angegebenen Attributen übereinstimmen.

Folgende drei Situationen können dabei auftreten:

- der Attributvektor, der aus den neuen Eingangsdateien entsteht, entspricht dem in der Projekt Datei definierten Attributvektor
- der neue Attributvektor enthält alle Attribute der des definierten Attributvektors sowie zusätzliche Attribute, die aufgrund der neuen Eingangsdateien einfließen
- der neue Attributvektor enthält weniger Attribute als der definierte Attributvektor

Generell bleiben alle Zuordnungen bestehen, welche Attribute betreffen, deren Kategorie, Name und Typ sich nicht verändert hat. Dies sind Attribute, die sowohl im definierten als auch im neuen Attributvektor enthalten sind. Die ersten beiden Fälle führen daher nicht zum Verlust bestehender Attributzuordnungen. Kritisch ist lediglich der letzte Fall. Attributzuordnungen für Attribute, die im neuen Attributvektor nicht mehr enthalten sind, gehen in diesem letzten Fall automatisch verloren.

Im dritten Fall wird das Programm beim Starten des Konvertierungsprozesses eine Warnmeldung.



Ein automatisches Update der Attributzuweisungen löscht alle Zuordnungen die von den eingelesenen Daten nicht mehr unterstützt werden. Falls man die Attributzuweisungen nicht erneuern lässt, kommt es im Laufe der Konvertierung zu Fehlermeldungen, sobald das Programm versucht, die nicht mehr vorhandenen Attribute auszulesen. In der Konsole werden in diesem Fall entsprechende Warnmeldungen ausgegeben.
Wenn man die Attributzuweisungen automatisch updaten lässt kann die Projektdatei über den Menüpunkt „Save project“ oder durch eine Abfrage beim Schließen des Programmes gespeichert werden.

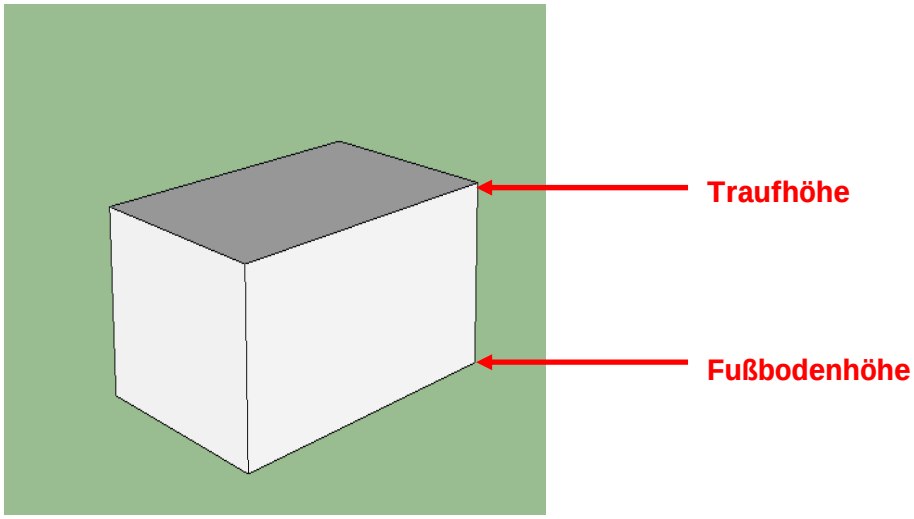
3.2 Konvertierung der Geometrie

Ziel des Shape2CityGML-Konverters ist die automatische Generierung von 3D LOD1 Gebäuden. Als Eingangsdaten werden hierfür folgende Informationen verwendet.

- 2D Polygon (Grundriss) (Geometrie des Shape Features)

- Absoluter Höhenwert der Traufhöhe des Gebäudes (Shape Attribut)
- Absoluter Höhenwert der Fußbodenhöhe des Gebäudes (Shape Attribut)

Der Konvertierungsprozess setzt nun den Grundriss als Bodenplatte des neuen LOD1 Gebäudes mit dem Höhenwert der angegebenen Fußbodenhöhe. Eine Kopie des Grundrisses mit geänderter Orientierung wird als Dachplatte mit der Traufhöhe als Höhenwert eingefügt. Zusätzlich werden für alle Wände entsprechende Polygone erstellt. Zusammen ergibt sich dann ein Volumenkörper (gml:Solid), welcher als LOD1 Geometrie dem CityGML Gebäude hinzugefügt wird.



4 Erstellen und Verwenden von Adressvorlagen (Templates)

Ein besonderes Feature des Shape2CityGML-Konverters ist die sehr flexible Einbindung von Gebäude-Adressinformationen in die resultierenden CityGML-Dokumente anhand von benutzerdefinierten Adressvorlagen (Adress-Templates).

Adressinformationen werden in CityGML mit Hilfe der eXtensible Address Language (xAL) kodiert, die von der Organization for the Advancement of Structured Information Standards (OASIS, http://www.oasis-open.org/) spezifiziert wird. Das Ziel von OASIS ist die Weiterentwicklung von E-Business- und Web-Service-Standards sowie die Schaffung internationaler Standards in diesem Bereich. xAL ist ein solcher internationaler Standard und liegt derzeit in der Version 2.0 vor, die auch für CityGML zum Einsatz kommt.

xAL ist eine XML-Anwendung zur Abbildung von internationalen Adressinformationen. Mit xAL steht eine flexible Beschreibungssprache für Adressinformationen bereit, die nicht beschränkt ist auf Eigenheiten der Adresskodierung in einzelnen Ländern, sondern beliebige internationale Adressformate unterstützt. Mit xAL ist es daher möglich, ebenso deutsche wie etwa japanische Adressdaten zu beschreiben. Dieser hohe Grad an Flexibilität führt allerdings dazu, dass bereits deutsche Adressdaten auf unterschiedlich komplexe, aber für sich genommen gültige xAL-Dokumente abgebildet werden können.

Um diese Flexibilität auch bei der Konvertierung von Shape-Dateien zu gewährleisten, arbeitet der Shape2CityGML-Konverter nicht mit einer intern festgelegten xAL-Adressstruktur, sondern erlaubt es dem Benutzer, beliebige xAL-Adressvorlagen für den Konvertierungsprozess vorzugeben. Der Umgang mit xAL-Adressvorlagen wird in den nächsten Abschnitten vorgestellt.

4.1 Erstellen von Adressvorlagen

Eine **Adressvorlage** muss ein **valides xAL-Dokument** sein. Das nachfolgende Beispiel zeigt eine solche gültige Adressvorlage.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0
../schemas/xAL/2.0/xAL.xsd">
  <Country>
    <CountryName>Germany</CountryName>
    <Locality Type="Town">
      <LocalityName>Potsdam</LocalityName>
      <Thoroughfare Type="Street">
        <ThoroughfareNumber>%Hausnummer%
%Hausnummer_Adressierungszusatz%</ThoroughfareNumber>
        <ThoroughfareName>%Strassenschluessel%</ThoroughfareName>
      </Thoroughfare>
      <PostalCode>
        <PostalCodeNumber>53115</PostalCodeNumber>
      </PostalCode>
    </Locality>
  </Country>
</AddressDetails>
```

Die Adressvorlage erfüllt zwei Hauptaufgaben:

1. Sie definiert diejenige **xAL-Elementstruktur**, die vom Shape2CityGML-Konverter 1:1 als Struktur der Gebäudeadresse in das CityGML-Dokument übernommen wird. Damit können beliebig komplexe und benutzerdefinierte xAL-Strukturen in den Konvertierungsprozess eingebunden werden.

2. Sie beinhaltet **Platzhalter** (Token), die während des Konvertierungsprozesses durch Daten aus den Shape-Eingangsdateien ersetzt werden. Damit werden die Adressvorlagen zur Laufzeit mit konkreten Adressdaten für ein Gebäude befüllt.(siehe Kapitel 4.2)

Um zu gewährleisten, dass als Ergebnis des Konvertierungsprozesses gültige CityGML-Dokumente vom Shape2CityGML-Konverter erzeugt werden, müssen benutzerdefinierte Adressvorlagen folgende Voraussetzungen erfüllen:

1. Adressvorlagen müssen **valide xAL-Dokumente** sein. Das bedeutet, dass sie im Einklang mit dem xAL-Schema-Dokument stehen müssen. Zur Prüfung der xAL-Validität wird der Einsatz von entsprechenden XML-Werkzeugen empfohlen. Spätestens beim Laden der Adressvorlage in den Shape2CityGML-Konverter prüft dieser, ob es sich um ein valides xAL-Dokument handelt oder nicht. Im letzteren Fall wird die Adressvorlage nicht akzeptiert.
2. Das **Wurzelelement** (Root-Element) der xAL-Struktur muss das xAL-Element **<AddressDetails>** sein (siehe obiges Beispiel einer gültigen Adressvorlage). Diese Vorgabe ergibt sich direkt aus der CityGML-Spezifikation für Adresselemente.

Der Shape2CityGML-Konverter wird mit einem Beispiel einer gültigen xAL-Adressvorlagen ausgeliefert (`PotsdamAdress.xml`). Diese befindet sich im Unterordner `templates` des Programmverzeichnisses und kann als Ausgangspunkt für eigene Templates dienen. Des Weiteren ist auch das offizielle xAL-Schemadokument in der Version 2.0 Bestandteil des Programmpakets. Es befindet sich im Unterordner `schemas` und sollte zur XML-Validierung der selbst erstellten Adressvorlagen verwendet werden.

Das folgende Listing zeigt ein minimales xAL-Fragment, das zur Erzeugung eigener Adressvorlagen verwendet werden kann.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AddressDetails xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0
  ../schemas/xAL/2.0/xAL.xsd"
  xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">

  <!-- Benutzerdefinierte Adressstruktur hier einfügen -->

</AddressDetails>
```

Im Wurzelelement `<AddressDetails>` müssen zwei Namensräume definiert sowie der Ort des xAL-Schemadokuments angegeben werden. Der Namensraum `urn:oasis:names:tc:ciq:xsdschema:xAL:2.0` ist der offizielle xAL-Namensraum und wird daher als Default-Namensraum für das gesamte Dokument festgelegt. Der zweite Namensraum `http://www.w3.org/2001/XMLSchema-instance` muss angegeben werden, um das Attribut `schemaLocation` aus diesem Namensraum verwenden zu können. Dieses Attribut wird sodann dazu genutzt, den Ort der xAL-Schemadatei anzugeben. Die `schemaLocation` wird immer als Wertepaar definiert: Der erste Bestandteil des Wertepaares ist der Namensraum, für welchen das Schema-Dokument gelten soll – im Beispiel muss daher der xAL-Namensraum wiederholt werden. Der zweite Bestandteil ist der Speicherort des Schema-Dokuments in Form einer gültigen URL. Im obigen Beispiel wird dieser Speicherort durch eine Pfadangabe definiert (`../schemas/xAL/2.0/xAL.xsd`), die relativ ist zum Speicherort der Adressvorlage selbst. Dieser Speicherort kann so übernommen werden, wenn die Adressvorlage im Unterorder `templates` des Programmverzeichnisses erstellt wird. Beide Werte müssen durch ein einzelnes Leerzeichen getrennt werden. Unterhalb des Wurzelements `<AddressDetails>` kann nun die benutzerdefinierte Adressstruktur aufgebaut werden.

4.2 Verwenden von Token

Um während des Konvertierungsprozesses die Adressstruktur mit Werten aus den Shape-Eingangsdaten füllen zu können, müssen bereits beim Erstellen der Adressvorlage entsprechende Platzhalter gesetzt werden. Folgende Token sind möglich.

- %Gemeindeschluessel%
- %Strassenschluessel%
- %Hausnummer%
- %Hausnummer_Adressierungszusatz%
- %Kennung%
- %Laufende_Nummer_des_Gebaeudes%

Token können an jeder beliebigen Stelle in der Adressvorlage eingesetzt werden, wo das xAL-Schema die Angabe von Adressdaten vorsieht, und somit typischerweise als Attribut- oder Elementwert. Es können auch mehrere Platzhalter direkt aufeinanderfolgen. Nicht erlaubt hingegen ist es, vordefinierte xAL-Strukturen durch Platzhalter zu ersetzen, etwa Attribut- oder Elementnamen.

Ob alle Platzhalter in einer Adressvorlage korrekt verwendet wurden lässt sich einfach dadurch feststellen, ob die Adressvorlage gegen das xAL-Schema validiert oder nicht. Ein entsprechendes XML-Wekzeug würde so das zweite Beispiel als nicht korrekt erkennen.

Überall dort, wo Platzhalter innerhalb der Adressvorlage verwendet werden können, kann auch statischer Text gesetzt werden. Statischer Text unterscheidet sich von Token dadurch, dass er nicht durch zwei %-Zeichen eingefasst wird. Er wird vom Shape2CityGML-Konverter nicht weiter interpretiert und während des Konvertierungsprozesses 1:1 in die Gebäudeadresse eingebunden.

4.3 Datenherkunft der Token

Die in Kapitel 4.2 aufgelisteten Token sind fest vorgegeben und setzen sich aus den folgenden Daten zusammen. Der Shape2CityGML-Konverter liest aus der Shape-Datei das Attribut „OBJEKTNAME“ aus, welches die Gebäudekennzeichnung nach der VALK Richtlinie enthält. In diesem Gebäudekennzeichen sind die mehrere Werte codiert. Die aufgelisteten Token entsprechen diesen codierten Werten.

Die Token sind folgenden Bestandteilen des Gebäudekennzeichens zugeordnet.

%Gemeindeschluessel%	Gemeindeschlüssel	8 Stellen
%Strassenschluessel%	Straßenschlüssel	5 Stellen
	Hausnummer	
%Hausnummer%	Ganze Zahl	4 Stellen
%Hausnummer_Adressierungszusatz%	Adressierungszusatz	4 Stellen
%Kennung%	Kennung	1 Stelle
%Laufende_Nummer_des_Gebaeudes%	Laufende Nummer des Gebäudes	2 Stellen
Summe		24 Stellen

Eine Sonderstellung hat hier der Token %Strassenschluessel%, welcher anhand einer Zuordnungsdatei in den entsprechenden Straßennamen umgewandelt wird. (siehe Kapitel 4.5)

Beispiel Gebäudekennzeichen:

12054000501161001 P02	(24 Stellen, in der Mitte 4 Leerzeichen)
%Gemeindeschluessel%	= 12054000
%Strassenschluessel%	= 50116

%Hausnummer% = 1001
%Hausnummer_Adressierungszusatz% = (4 Leerzeichen)
%Kennung% = P
%Laufende_Nummer_des_Gebaeudes% = 02

4.4 Verwenden von Adressvorlagen

Im Folgenden wird die Einbindung und Verwendung einer xAL-Adressvorlage in den Konvertierungsprozess des Shape2CityGML-Konverters dargestellt. Als Beispiel dient wiederum Vorlage aus dem vorigen Abschnitt:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0
../schemas/xAL/2.0/xAL.xsd">
  <Country>
    <CountryName>Germany</CountryName>
    <Locality Type="Town">
      <LocalityName>Potsdam</LocalityName>
      <Thoroughfare Type="Street">
        <ThoroughfareNumber>%Hausnummer%
%Hausnummer_Adressierungszusatz%</ThoroughfareNumber>
        <ThoroughfareName>%Strassenschluessel%</ThoroughfareName>
      </Thoroughfare>
      <PostalCode>
        <PostalCodeNumber>53115</PostalCodeNumber>
      </PostalCode>
    </Locality>
  </Country>
</AddressDetails>
```

Das Adress-Template verwendet die drei Token: `%Hausnummer%`, `%Hausnummer_Adressierungszusatz%` sowie `%Strassenschluessel%`.

In Kapitel 2.3.2 wurde bereits die graphische Benutzeroberfläche, die der Verwaltung von Adressvorlagen dient, vorgestellt. Das Laden einer xAL-Adressvorlage für ein Projekt erfolgt im Kartenreiter „Address“. Dort kann die Adressvorlage per „Browse“-Button bzw. per Drag & Drop ausgewählt werden. Über den Button „Read template“ wird die Adressvorlage vom Shape2CityGML-Konverter geladen und geprüft.

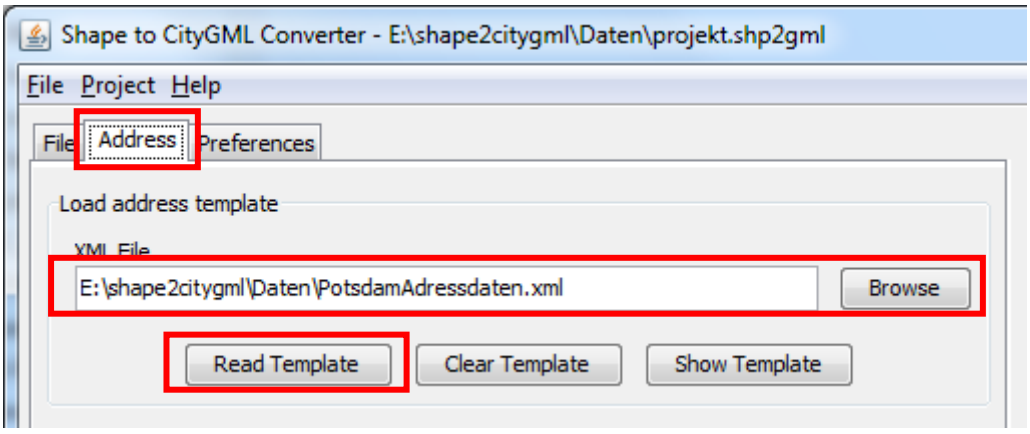


Abbildung 13: Laden eines Adresstemplates

Der Shape2CityGML-Konverter führt für jede Adressvorlage eine xAL-Validitätsprüfung durch. Der Fortgang dieser Prüfung wird im Konsolen-Fenster mitprotokolliert. Dort werden ebenso Fehlermeldungen angezeigt, die während der Prüfung auftreten. Diese Fehlermeldungen enthalten neben dem Fehlergrund auch immer die Angabe der Zeilen- und Spaltennummer, um den Ort des Fehlers innerhalb der Vorlagendatei schnell auffinden zu können. Invalide xAL-Adressvorlagen werden vom Shape2CityGML-Konverter nicht angenommen und müssen korrigiert werden.

Im Erfolgsfall wird die erkannte xAL-Adressstruktur am Ende des Prüfprozesses im Konsolen-Fenster dargestellt. Für die oben vorgestellte Beispielvorlage ergibt sich die in Abbildung 13: Laden eines AdresstemplatesAbbildung 14 gezeigte Konsolenausgabe:

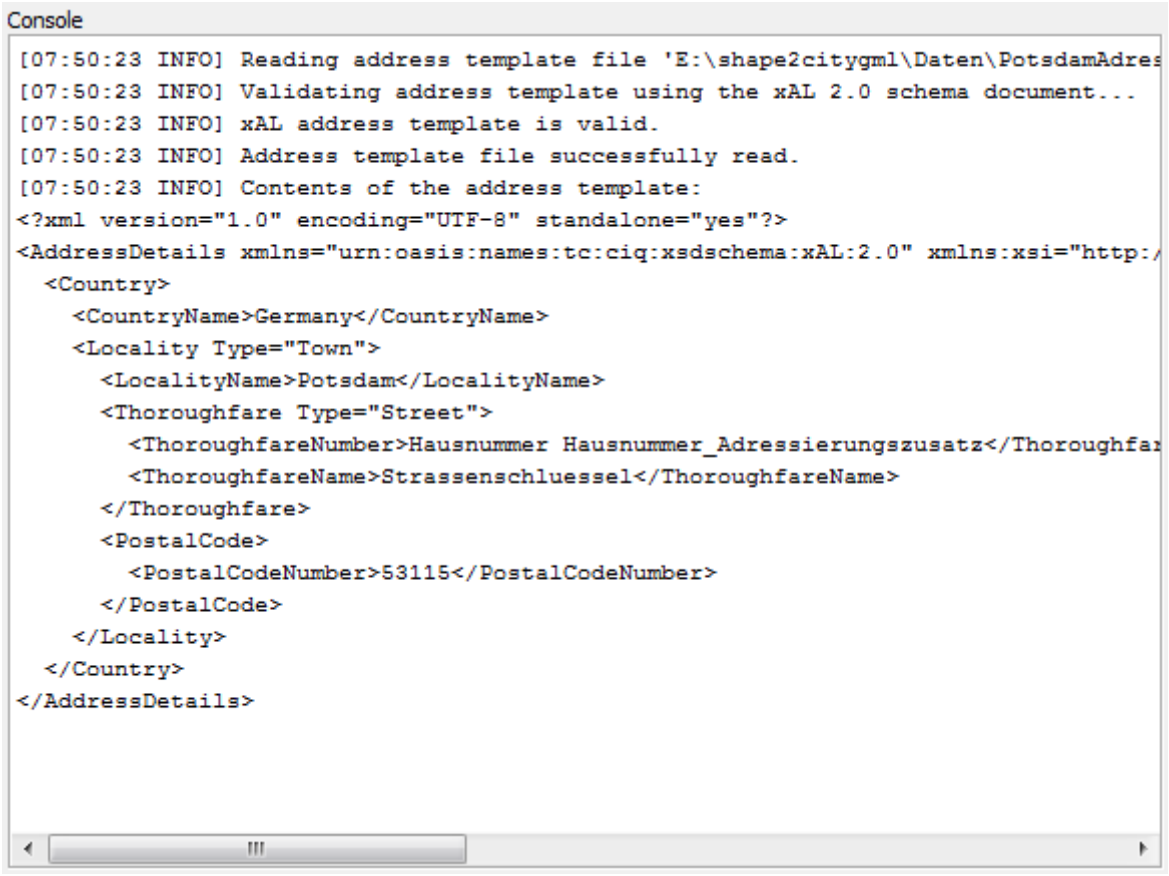


Abbildung 14: Addresstemplate Konsolenausgabe

4.5 Einlesen einer Straßenschlüssel Zuordnungsdatei

Damit der Adresstoken %Strassenschluessel% nicht nur, wie in dem ALK Gebäudekennzeichen vorgegeben, einen Straßenschlüssel enthält, sondern der ausgeschriebene Straßennamen verwendet wird, ermöglicht Shape2CityGML das Angeben einer Zuordnungsdatei. Diese Datei muss als Komma oder Semikolon getrennte CSV Datei vorliegen.

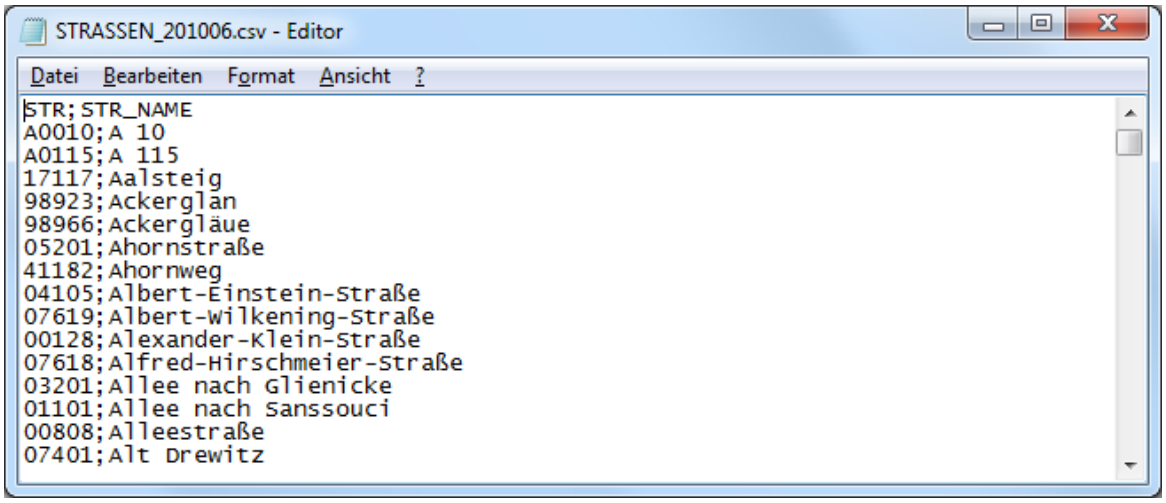


Abbildung 15 Beispiel einer CSV Datei die Straßenschlüsseln Straßennamen zuordnet

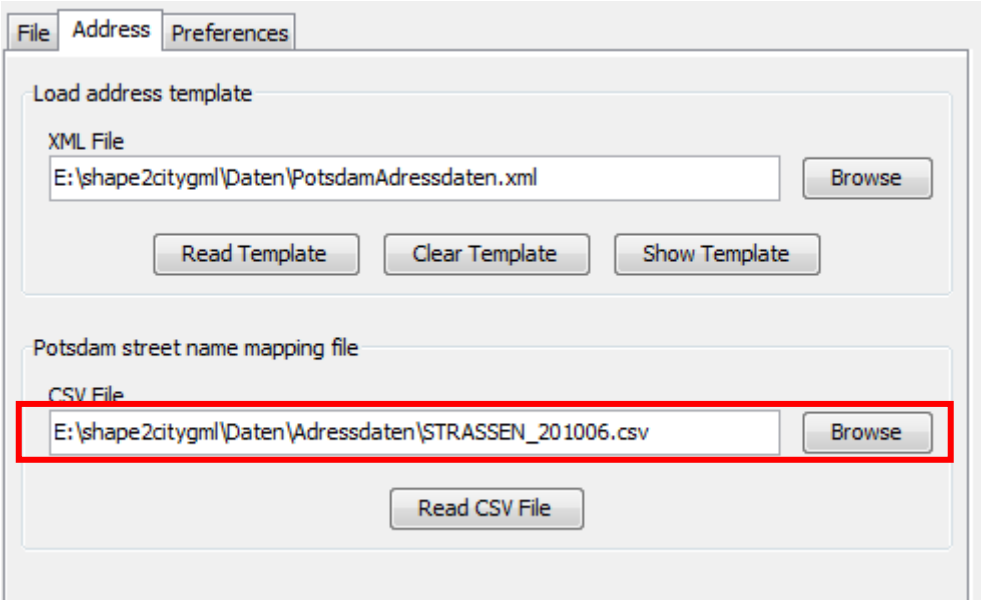


Abbildung 16: Laden einer CSV Datei

Das Laden einer CSV Zuordnungs Datei für ein Projekt erfolgt im Kartenreiter „Address“. Dort kann die CSV-Datei per „Browse“-Button bzw. per Drag & Drop ausgewählt werden. Über den Button „Read CSV File“ wird die CSV-Datei getestet und gezählt wieviele Einträge vorhanden sind. Folgende Ausgabe wird auf der Konsole erzeugt.

```
[07:59:56 INFO] Potsdam street mapping: 1681 entries found
```

4.6 Speicherung der Adressvorlagen

Wird einem Projekt auf die im vorigen Abschnitt dargestellte Weise eine xAL-Adressvorlage hinzugefügt, so werden beim Speichern des Projekts **automatisch** Kopien der Vorlagendatei und der CSV-Datei im gleichen Verzeichnis abgelegt, in welchem auch die Projektdatei gespeichert wird. Die Namen der originalen Dateien werden dabei auch für die Kopien übernommen.

Wird das Projekt nach einem Neustart der Anwendung erneut geladen, so werden automatisch auch die xAL-Adressvorlage und die CSV-Datei aus den kopierten Dateien eingelesen. Während des Einlesens der Adressvorlage wird allerdings zunächst eine xAL-

Validitätsprüfung durchgeführt – nur valide Vorlagen werden akzeptiert. Dies stellt sicher, dass zwischenzeitliche Änderungen an der Vorlagendatei nicht zu ungültigen CityGML-Dokumenten führen.

Wird eine Projektdatei auf einen anderen Computer übertragen, so muss sichergestellt werden, dass auch die zugehörige xAL-Adressvorlage und CSV-Datei im selben Verzeichnis mit übertragen werden.

5 Benutzung des Kommandozeilen-Interface

Der Shape2CityGML-Konverter kann sowohl mit einer graphischen Benutzeroberfläche (Graphical User Interface, GUI) als auch mit einem Kommandozeilen-Interface (Command-Line Interface, CLI) bedient werden. Folgendes Kapitel befasst sich mit der Steuerung des Shape2CityGML-Konverters über das Kommandozeilen-Interface.

Befehle werden in der CLI-Variante über sogenannte Kommandozeilen-Parameter an den Shape2CityGML-Konverter weitergeleitet. Das erlaubt eine Einbettung dieser Befehle in komplexe Batch-Dateien, die nicht-interaktiv und bei Bedarf zeitgesteuert vom Betriebssystem abgearbeitet werden (vgl. bereits die Ausführungen in Kapitel 1.4.2). Die Programmfunktionalität zum Konvertieren der Shape-Eingangsdatensätze in CityGML-Dokumente ist für beide Varianten jedoch identisch.

Alle nachfolgenden Beispiele wurden auf einem Windows 7-System mit Hilfe der Windows-spezifischen MS-DOS-Eingabeaufforderung erstellt. Jedes aktuelle Betriebssystem stellt jedoch vergleichbare Kommandozeileninterpreter zur Verfügung. Auch wenn die Bedienung der jeweiligen Kommandozeileninterpreter von System zu System variiert, unterscheiden sich die Befehle zur Steuerung des Shape2CityGML-Konverters in diesen Umgebungen nicht. Die nachfolgenden Beispiele können daher leicht auf andere Betriebssysteme portiert werden.

5.1 Starten des Kommandozeileninterpreters unter Windows 7

In Windows 7 kann die MS-DOS-Eingabeaufforderung über „Start → Programme → Zubehör → Eingabeaufforderung“ gestartet werden. Alternativ genügt es, „Start → Ausführen…“ anzuklicken und im nachfolgenden Dialog den Befehl `cmd` einzugeben sowie den „Ok“-Button zu betätigen. Es öffnet sich ein der Abbildung 17 vergleichbares Fenster.

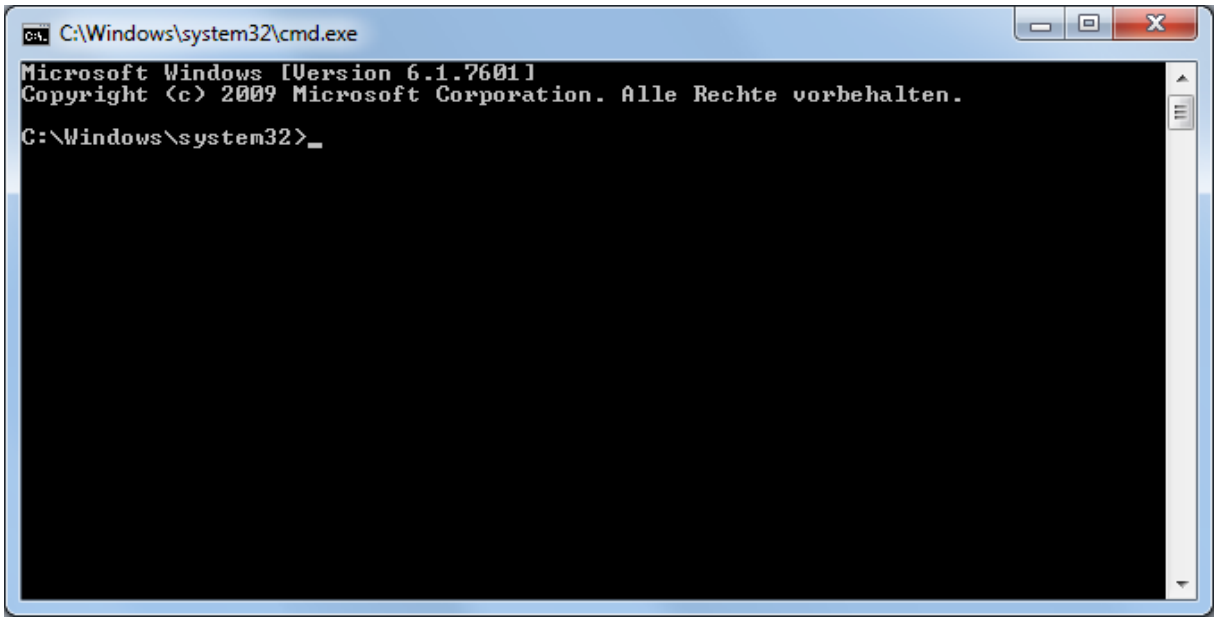


Abbildung 17: Windows 7 Eingabeaufforderung

Um über die Kommandozeile mit dem Shape2CityGML-Konverter zu interagieren, muss in einem ersten Schritt in das Programmverzeichnis der Anwendung gewechselt werden. In Windows-Umgebungen wird hierfür das Kommando `cd` verwendet.

Befindet sich die Anwendung beispielsweise im Ordner „C:\shape2citygml“, ist der nachfolgende Befehl erforderlich.

```
cd e:\shape2citygml
```

Ein zusätzliches `dir`-Kommando zeigt sodann den Inhalt des Ordners an, in welchem sich der Kommandozeileninterpreter aktuell befindet. So kann überprüft werden, ob es sich dabei tatsächlich um das Programmverzeichnis des Shape2CityGML-Konverters handelt. Die Inhalte des Ordners sollten denen in Kapitel 1.3 dargestellten entsprechen. Die Abbildung 18 zeigt dies.

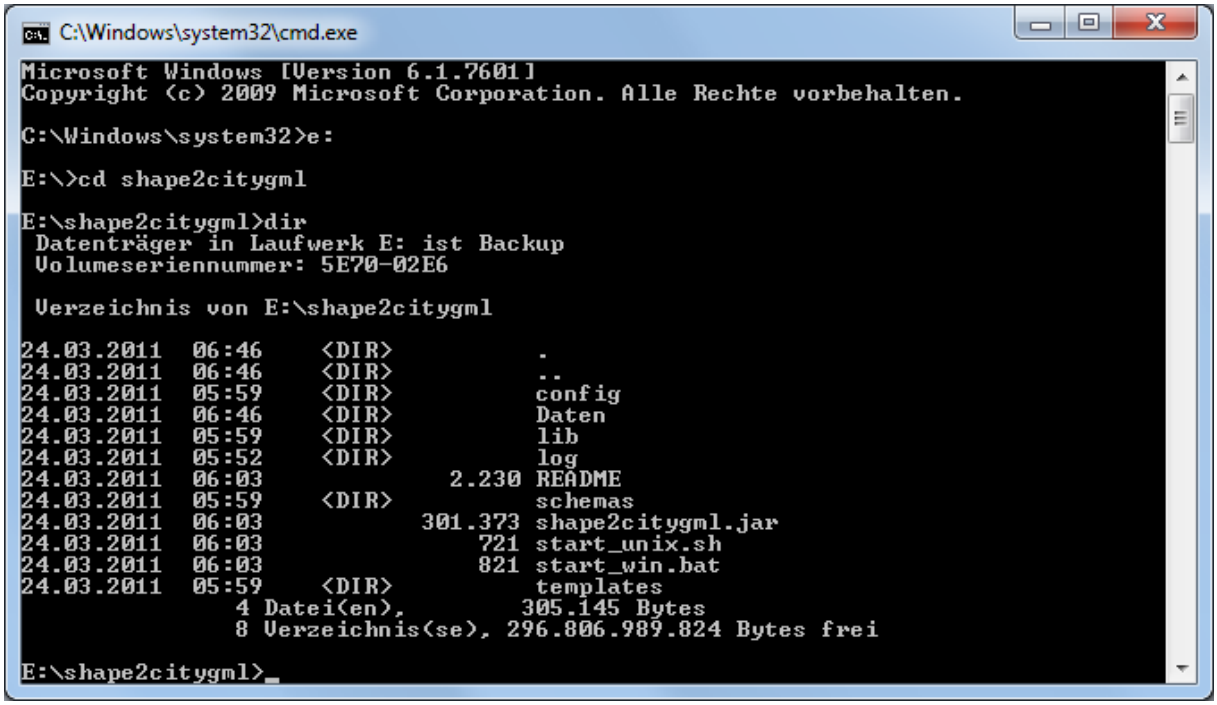


Abbildung 18: Ordner shape2citygml

5.2 Grundsätzliche Verwendung des Kommandozeilen-Interfaces

Sobald sich der Kommandozeileninterpreter im Programmverzeichnis des Shape2CityGML-Konverters befindet, kann das Kommandozeilen-Interface durch folgenden Befehl ausgeführt werden (vgl. bereits die Ausführungen zum Starten des Programms in Kapitel 1.4).

```
java –jar shape2citygml.jar –shell
```

Der zusätzliche Parameter `–shell` gibt an, dass nicht die graphische Benutzeroberfläche gestartet werden soll. Die Kommunikation über das CLI erfordert daher immer zumindest die Angabe des Parameters `–shell`.

Wird der obige Befehl abgesetzt, führt dies allerdings zu folgender Fehlermeldung:

```
Please provide at least one file to be converted using "-lod1"
for shell version
```

Zusätzlich zur Option `–shell` sind somit weitere Parameter erforderlich, um den Konvertierungsprozess zu starten. Der Shape2CityGML-Konverter beanstandet durch entsprechende Fehlermeldungen auf der Konsole solange fehlende Parameter an, bis alle erforderlichen Mindestangaben erfüllt sind. Eine Übersicht aller verfügbaren Parameter sowie derjenigen Parameter, die mindestens bei Verwendung des CLI erwartet werden, erfolgt im nächsten Abschnitt.

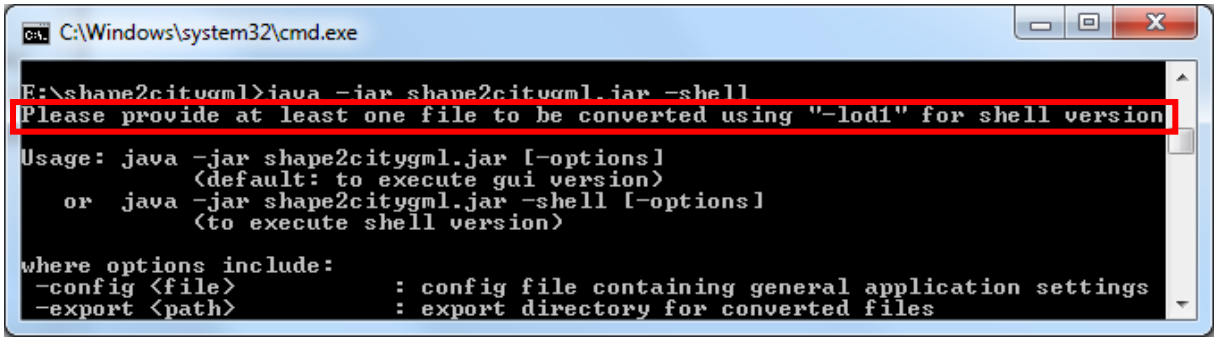


Abbildung 19: Kommandozeile Fehlermeldung (Darstellung gekürzt)

5.3 Die verfügbaren Kommandozeilen-Parameter

Eine Übersicht aller verfügbaren Kommandozeilen-Parameter erhält man mit dem Befehl:

```
java –jar shape2citygml.jar –help
```

Die Ausgabe des Befehls zeigt die Abbildung 3. Oberhalb der Liste der verfügbaren Parameter wird zusätzlich ein kurzer Hinweis zur Verwendung (Usage) des Shape2CityGML-Konverters über die Kommandozeile ausgegeben.

5.3.1 Parameter zur Festlegung der Eingangsdatensätze

Die zu konvertierenden Shape-Eingangsdatensätze werden dem Shape2CityGML-Konverter im Kommandozeilen-Interface über den Parameter `–lod1` übergeben.

- `-lod1`
  Verwendung: zwingend
  Beschreibung: Über den Parameter `–lod1` werden diejenigen Shape-Dateien angegeben, die bei der Konvertierung nach CityGML für die Generierung der LOD1-Repräsentation verwendet werden. Unterstützt wird die Angabe von Dateien und Ordnern. Im letzteren Fall werden alle Shape-Dateien in dem angegebenen Ordner sowie rekursiv in allen Unterordnern für die Konvertierung verwendet. Sollen mehrere Dateien und/oder Ordner in einem Durchgang konvertiert werden, so müssen diese durch Semikolon (;) voneinander getrennt angegeben werden. Die Verwendung von Wildcards innerhalb von Dateinamen ist erlaubt.

5.3.2 Parameter zur Definition des Attributmappings

Das Kommandozeilen-Interface erwartet eine vorgefertigte Projektdatei (.shp2gml-Datei), in welcher das Attributmapping vollständig definiert ist. Die Übergabe dieser Projektdatei an den Shape2CityGML-Konverter erfolgt durch den Parameter `–project`.

Die Projektdateien des Shape2CityGML-Konverters sind XML-Dokumente, welche einer vordefinierten Struktur folgen müssen. Diese Struktur ist in einem eigenen XML-Schema-Dokument festgelegt.

- `-project`
  Verwendung: zwingend
  Beschreibung: Über die Option `–project` wird die Projektdatei (.shp2gml-Datei) spezifiziert, welche das Attributmapping für den Konvertierungsprozess festlegt. Die Zuweisungsvorschriften in der Projektdatei werden auf die Eingangsdatensätze angewandt. Daher müssen die Zuweisungen in der Projektdatei auf die Shape-Attribute in den Eingangsdaten passen. Für die CLI-Version ist dieser Parameter

zwingend anzugeben. In der GUI-Version ist er optional – wird er angeben, so wird die Projektdatei bereits beim Programmstart geladen.

5.3.3 Parameter zur Festlegung des Exportverzeichnisses

Im Exportverzeichnis legt der Shape2CityGML-Konverter die aus dem Konvertierungsprozess resultierenden CityGML-Dokumente ab. Das Verzeichnis wird über den Parameter `-export` bekannt gemacht.

- `-export`
  Verwendung: zwingend
  Beschreibung: Über den Parameter `-export` wird das Exportverzeichnis für den Konvertierungsprozess festlegt. Das angegebene Verzeichnis muss bereits existieren und beschreibbar sein, ansonsten bricht der Konvertierungsprozess mit einer entsprechenden Fehlermeldung ab.

5.3.4 Parameter zur Definition benutzerspezifischer Anwendungseinstellungen

Benutzerspezifische Anwendungseinstellungen steuern das lokale Verhalten des Shape2CityGML-Konverters. Sie dürfen nicht mit den Projekteinstellungen verwechselt werden.

Projekteinstellungen definieren das Attributmapping zwischen den Shape-Eingangsdaten und den CityGML-Datensätzen, die aus dem Konvertierungsprozess resultieren. Sie sind unabhängig von einer konkreten Installation des Shape2CityGML-Konverters, da sie zwischen beliebigen Instanzen auch über Betriebssystemgrenzen hinweg ausgetauscht werden können.

Benutzerspezifische Anwendungseinstellungen hingegen beziehen sich auf eine lokale Installation des Shape2CityGML-Konverters und betreffen daher keine Einstellungen des Konvertierungsprozesses. Eine weitere wichtige benutzerspezifische Einstellung ist das Log-Verhalten des Shape2CityGML-Konverter. So kann der Log-Level für Meldungen der Anwendung festgelegt werden, oder aber spezifiziert werden, ob Log-Meldungen zusätzlich zur Konsolenausgabe auch in eine Datei geschrieben werden sollen. Das Log-Verhalten kann somit zwischen unterschiedlichen Instanzen des Shape2CityGML-Konverters variieren.

Grundsätzlich speichert der Shape2CityGML-Konverter benutzerspezifische Anwendungseinstellungen in der Datei `config.xml` im Unterordner `config` des Programmverzeichnisses ab. Beim Programmstart wird an ebendieser Stelle nach den Benutzereinstellungen gesucht und diese entsprechend gesetzt. Kann die Datei `config.xml` nicht gefunden werden, werden alle Benutzereinstellungen automatisch auf Standardwerte gesetzt.

Über die Kommandozeile ist es nun mittels des Parameters `-config` möglich, dem Shape2CityGML-Konverter eine andere Datei mit Benutzereinstellungen vorzugeben. Die Einstellungen in dieser Datei gehen damit den Einstellungen in der Standarddatei `config.xml` vor. So kann beispielsweise sicher gestellt werden, dass der Konvertierungsprozess immer mit denselben Log-Einstellungen durchgeführt wird, auch wenn unterschiedliche Instanzen des Shape2CityGML-Konverters eingesetzt werden. Änderungen an Benutzereinstellungen während der Programmlaufzeit werden dann auch in diese vorgegebene Datei zurückgeschrieben.

- `-config`
  Verwendung: optional
  Beschreibung: Der Parameter `-config` ermöglicht die Angabe einer alternativen Konfigurationsdatei mit benutzerspezifischen Anwendungseinstellungen, die

anstelle der Standarddatei `config.xml` geladen werden soll. Änderungen an Benutzereinstellungen während der Programmlaufzeit werden in diese Datei zurückgeschrieben.

5.3.5 Parameter zur Festlegung des Log-Verhaltens

Die Einstellungen zum Log-Verhalten des Shape2CityGML-Konverters sind von zentraler Bedeutung für den Konvertierungsprozess. In den Log-Meldungen werden alle Schritte des Konvertierungsprozesses sowie eventuell aufgetretene Probleme oder Fehler mit Zeitstempel und Schweregrad mitprotokolliert. Die entstehenden Log-Protokolle sollten daher immer nach Beendigung des Konvertierungsprozesses analysiert werden. Da es sich um reine Textmeldungen handelt, können die Log-Protokolle auch sehr einfach automatisiert verarbeitet werden.

Das Log-Verhalten kann in der graphischen Benutzeroberfläche eingestellt werden und wird dann in den benutzerspezifischen Anwendungseinstellungen (`config.xml`) abgespeichert. Beim Programmstart wird das Log-Verhalten entsprechend den Einstellungen in dieser Datei wieder initialisiert.

Für das Kommandozeilen-Interface ist es nun möglich, über eine Reihe zusätzlicher Parameter das Log-Verhalten direkt zu beeinflussen. Die Einstellungen gehen denjenigen in der Standard-Konfigurationsdatei `config.xml` bzw. einer alternativ angegebenen Konfigurationsdatei (über die Option `–config`) vor. Somit kann bereits auf Ebene des Kommandozeilen-Befehls zum Starten des Shape2CityGML-Konverters sichergestellt werden, dass ein bestimmtes Log-Verhalten eingehalten wird.

- `-logFile`
  Verwendung: optional
  Beschreibung: Der Schalter `–logFile` zeigt dem Shape2CityGML-Konverter an, dass alle Log-Meldungen zusätzlich zur Ausgabe auf der Konsole auch in eine Log-Datei geschrieben werden sollen. Bei der Verarbeitung mehrerer Eingangsdatensätze wird das Anlegen einer Log-Datei immer empfohlen.

- `-logFilePath`
  Verwendung: optional, nur in Zusammenhang mit `–logFile`
  Beschreibung: Standardmäßig werden alle Log-Dateien in den Unterordner `log` des Programmverzeichnisses geschrieben. Dieses Verhalten kann über den Parameter `-logFilePath` geändert werden, indem ein alternatives Verzeichnis angegeben wird. Existiert das Verzeichnis nicht, wird es beim Programmstart automatisch angelegt.

- `-logLevelFile`
  Verwendung: optional, nur in Zusammenhang mit `–logFile`
  Beschreibung: Über den Parameter `-logLevelFile` kann der Log-Level für die auszugebenden Log-Meldungen in der Log-Datei geändert werden. Erlaubte Werte sind `error`, `warn`, `info`, und `debug`. Der Standardwert ist `info`.

- `-logLevelConsole`
  Verwendung: optional
  Beschreibung: Über den Parameter `-logLevelConsole` kann der Log-Level für die auszugebenden Log-Meldungen in der Konsole geändert werden. Erlaubte Werte sind `error`, `warn`, `info`, und `debug`. Der Standardwert ist `info`.

5.3.6 Sonstige Parameter

- `-shell`
  Verwendung: zwingend
  Beschreibung: Der Parameter `–shell` zeigt dem Shape2CityGML-Konverter an,

dass beim Programmstart nicht die graphische Benutzeroberfläche geladen werden soll, sondern dass die Interaktion mit dem Benutzer über das Kommandozeilen-Interface erfolgt. Dieser Parameter ist daher zwingend, soll der Shape2CityGML-Konverter über sein CLI gesteuert werden.

- `-version`
  Verwendung: optional
  Beschreibung: Wird der Schalter `-version` angegeben, so wird der Versionsstring des Shape2CityGML-Konverter auf der Konsole ausgegeben. Es erfolgt danach kein Programmstart des Shape2CityGML-Konverters. Eventuell zusätzlich angegebene Parameter werden nicht ausgewertet. Der Schalter `-version` sollte daher immer alleine verwendet werden.

- `-help` (alternativ: `-h`)
  Verwendung: optional
  Beschreibung: Wird der Schalter `-help` bzw. `-h` angegeben, erfolgt die Ausgabe einer Hilfe auf der Konsole. Diese beschreibt kurz die Verwendung des Shape2CityGML-Konverters über die Kommandozeile und listet alle verfügbaren Kommandozeilen-Parameter auf. Es erfolgt danach kein Programmstart des Shape2CityGML-Konverters. Eventuell zusätzlich angegebene Parameter werden nicht ausgewertet. Der Schalter `-help` bzw. `-h` sollte daher immer alleine verwendet werden.

5.4 Beispiel für die Verwendung des Kommandozeilen-Interfaces

In diesem Abschnitt wird beispielhaft ein Kommandozeilen-Befehl entwickelt, welcher über das Kommandozeilen-Interface des Shape2CityGML-Konverters den Konvertierungsprozess von Shape-Eingangsdaten anstößt. Im Folgenden werden zunächst die Randbedingungen des Konvertierungsprozesses dargestellt.

Für das Beispiel wird angenommen, dass alle erforderlichen Eingangsdateien in Unterordnern des Hauptverzeichnisses `E:\shape2citygmlinput\Daten` organisiert sind. Die Abbildung 20 zeigt den Inhalt dieses Hauptverzeichnisses.

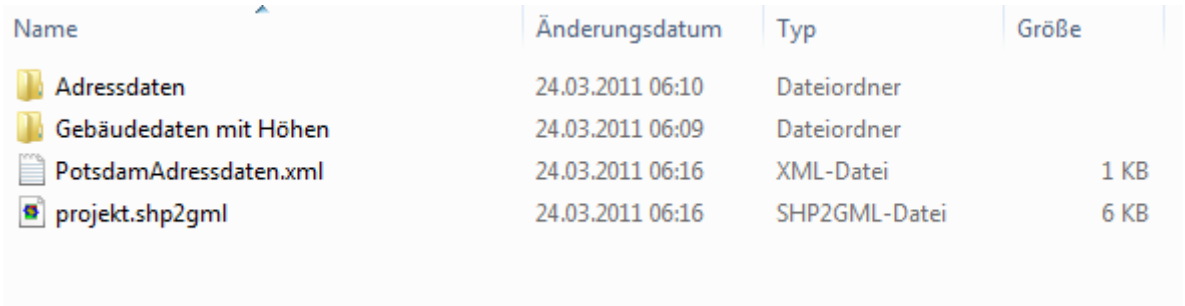


Abbildung 20: Beispieldaten

Die abgebildeten Unterordner wiederum sollen folgende Dateien enthalten:

- **Unterordner `Gebäudedaten mit Höhen`**
  Der Unterordner `Gebäudedaten mit Höhen` enthält eine einzelne Shape-Datei `Gebäude_05.shp`, welche zur Geometrierepräsentation der Gebäude in LOD1 verwendet wird. Daraus leitet sich folgende Option für den Shape2CityGML-Konverter ab:

```
-lod1 E:\shape2citygml\daten\Gebäudedaten mit Höhen\
```

- **Unterordner `Adressdaten`**
  Der Unterordner `Adressdaten` enthält die xAL-Adressvorlage sowie die zur Straßenschlüssel Auflösung benötigte CSV Datei

- **Datei `projekt.shp2gml`**
  Die Projekteinstellungsdatei `projekt.shp2gml` enthält alle erforderlichen Zuweisungen von Shape-Attributen auf CityGML-Attribute. Daraus leitet sich folgende Option für den Shape2CityGML-Konverter ab:

```
-project E:\shape2citygml\daten\projekt.shp2gml
```

Als zusätzliche Anforderungen sollen alle Log-Meldungen des Konvertierungsprozesses zusätzlich zur Ausgabe in der Konsole in eine Log-Datei geschrieben werden. Um diese später einfach auf Warnungen und Fehler untersuchen zu können, soll der Log-Level auf `warn` gesetzt werden. Daraus leiten sich folgende Optionen für den Shape2CityGML-Konverter ab:

```
-logFile -logLevelFile warn
```

Werden die einzelnen Optionen zusammengesetzt, so ergibt sich schließlich als Kommandozeilen-Befehl:

```
java -jar shape2citygml.jar -shell -lod1
E:\shape2citygml\daten\Gebäudedaten mit Höhen\ -export
E:\shape2citygml\daten -project
E:\shape2citygml\daten\projekt.shp2gml –logFile –logLevelFile
warn
```

Die Abbildung 21 zeigt die Meldungen des Shape2CityGML-Konverters nach Absetzung des Befehls.

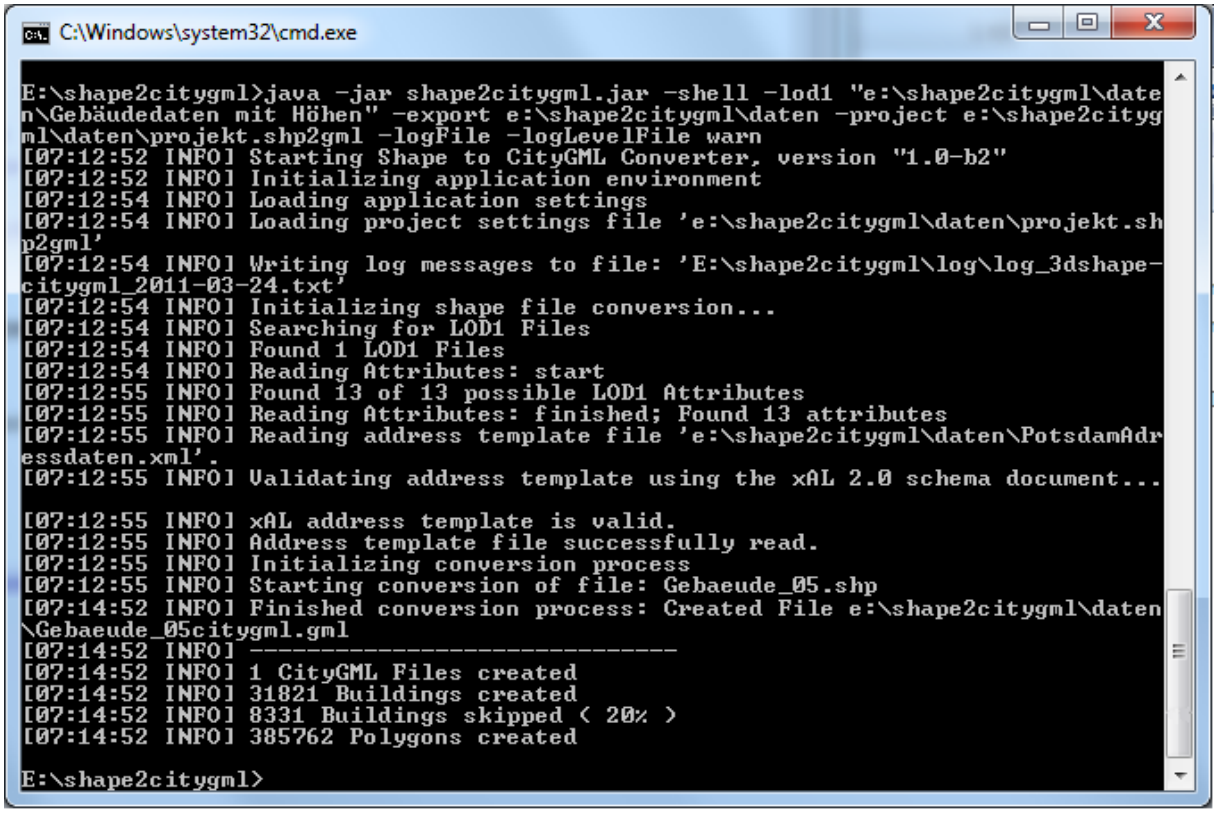


Abbildung 21: Kommandozeile Beispiel Meldungen