UML-Transformation
Version 1.0

Datum: 28.7.2020

Dr. J. Benner
Karlsruher Institut für Technologie (KIT)
Institut für Automation und Angewandte Informatik

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

1 Installation

1.1 Installations-Dateien

UML-Transformation.exe	Ausführbares Programm
Interop.EA.dll	*Enterprise Architect* Systembibliothek
UML-Transformation.ini	Konfigurations-Datei (XML-Format)
Ordner **Dokumentation**	• Benutzerhandbuch.pdf – Dies Dokument • ISO_19136-Annex_E.pdf – Annex E der GML 3.2.1 Spezifikation (*UML-to-GML application schema encoding rules)*
Ordner **XSD**	Verschiedene XML-Schema Dateien
Ordner **XSL**	Verschiedene Dateien mit XSLT-Transformationen

Tabelle 1: Installationsdateien der Software UML-Transformation

1.2 Installation der Software

Kopieren Sie die in Tabelle 1 angegebenen Installations-Dateien in einen beliebigen Ordner Ihres Rechners (**System-Ordner**). Die Ordnerstruktur dieses Installationsverzeichnisses darf nicht geändert werden.

Die Einstellungen in der Konfigurationsdatei müssen im Regelfall nicht geändert werden.

1.3 System-Voraussetzungen

Die Software läuft nur auf Windows-Systemen mit .NET Framework (ab Version 2.0), und wurde unter Windows 7 und Windows 10 getestet. Alle wichtigen System- und Sicherheitsupdates sollten installiert sein. Die Software erfordert weiterhin eine gültige Lizenz für die Software *Enterprise Architect* ab Version 10.

2 Überblick der Programm-Funktionalität

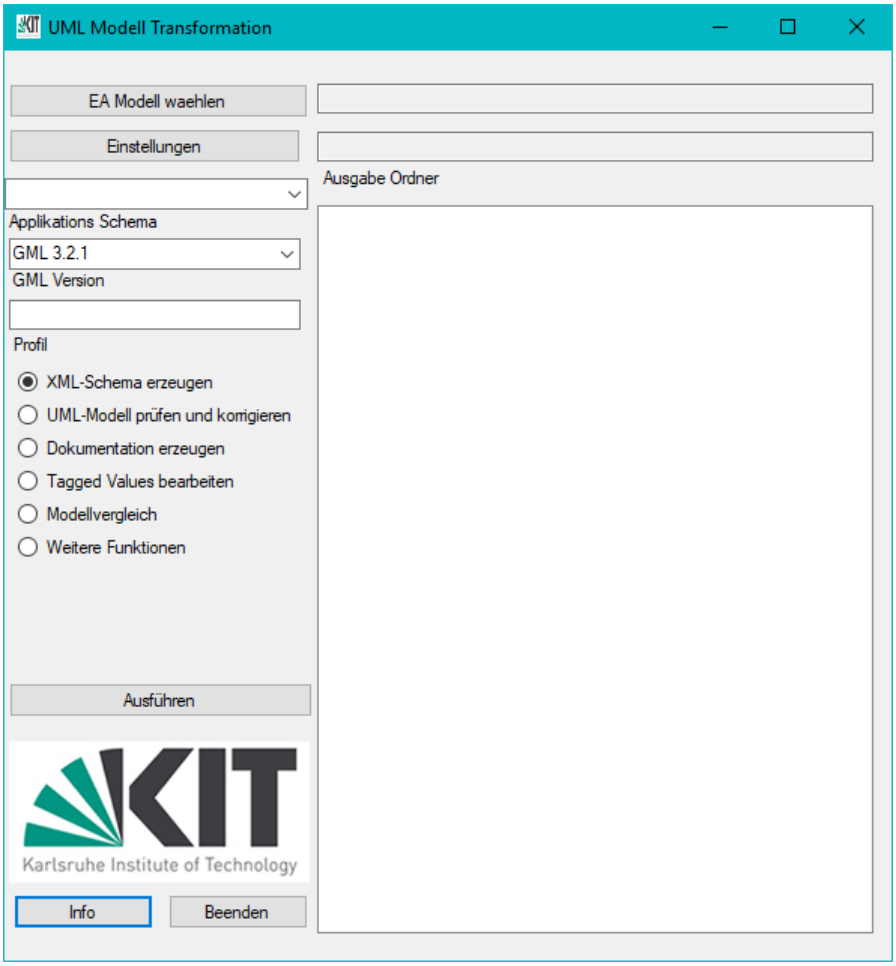


Abbildung 1: Benutzeroberfläche

EA Modell wählen	Auswahl und Öffnen des zu bearbeitenden *Enterprise Architect* (EA) UML-Modelles („*Eingabe-Modell*“). Anschließend wird das Eingabe-Modell kopiert, und alle Bearbeitungsoperationen finden auf dem kopierten Modell („*Temporäres Modell*“) statt.
Einstellungen	Öffnet eine Dialogbox, mit der zentrale Programm- Einstellungen geändert werden können (s. Kap. 2.1). Insbesondere kann hier der *Ziel-Ordner* festgelegt werden, in dem die Ausgabedateien abgelegt werden. Standardmäßig ist dies der Ordner des Eingabe-Modelles. Der Pfadname des Ziel-Ordners wird angezeigt.
Applikations Schema	In der ComboBox sind alle UML-Packages des ausgewählten UML-Modells aufgeführt, die den Stereotype <<ApplicationSchema>> haben. Hier muss das zu bearbeitende GML-Applikationsschema ausgewählt werden, dies ist standardmäßig das erste Package der Liste.
GML-Version	Diese ComboBox ist nur bei der Erzeugung von XML-Schemadateien aus dem UML-Modell aktiv. Hier wird festgelegt, ob die erzeugten XML-Schemata auf GML 3.1.1 oder GML 3.2.1 (Standard) aufbauen.
Profil	Wenn in diesem Eingabefeld der Name eines Applikationsschema-Profils eingetragen ist, werden die zugehörigen Profilregeln bei der

	Erzeugung der XML-Schemadateien und der Dokumentation berücksichtigt. Näheres dazu findet sich in Kap. 3.
UML-Modell Profil generieren	Diese CheckBox ist nur sichtbar, wenn über das Eingabefeld „Profil“ ein Profil des Applikationsschemas festgelegt wurde, und die Aktion „Dokumentation erzeugen“ ausgewählt wurde. Die Funktion ist unter „Dokumentation erzeugen“ beschrieben.
Ausführen	Startet die über die RadioButtons ausgewählte Programmfunktion. Vorher werden die für die Ausgabe benötigten Ordner angelegt, falls sie noch nicht vorhanden sind, und einige Schema-Dateien aus dem System-Ordner in den Ziel-Ordner kopiert.
XML-Schema erzeugen	Erzeugt aus dem UML-Modell ein GML-Applikationsschema (s. Kap. 4).
UML-Modell prüfen und korrigieren	Ermöglicht es, das UML-Modell vor Erzeugung der XML-Schema Dateien oder der Objektartenkataloge zu prüfen und optional auch bestimmte Korrekturen vorzunehmen (s. Kap. 5).
Dokumentation erzeugen	Bei dieser Funktion muss zunächst die Dateistruktur des erzeugten Objektartenkatalogs über die ComboBox „Struktur Objektartenkatalog“ festgelegt werden. • Bei Auswahl „Verteilte HTML-Dateien“ (Standard, s. Kap. 6.1) wird (bis auf Enumerationen und Codelisten) für jedes UML-Element eine eigene HTML-Datei erzeugt. Die einzelnen Dateien sind untereinander verlinkt und über gemeinsame Rahmendateien sowie eine Liste der UML-Elemente zugänglich. • Bei Auswahl „HTML-Gesamtdokument“ (s. Kap. 6.2) wird des gesamte Objektartenkatalog in eine HTML-Datei geschrieben. Eine Weiterverarbeitung, um daraus z.B. ein PDF-Dokument zu erzeugen, ist mit Standard-Textverarbeitungsprogrammen wie MS-Word möglich. • Bei Auswahl „Excel-Tabelle“ (s. Kap. 6.3) wird ein reduzierter Objektartenkatalog im Excel XML-Format erzeugt Wenn die CheckBox „UML-Modell Profil generieren“ aktiviert ist, wird das eingelesene Profil mit Hilfe der durch Tagged Values definierten Profilregeln (s. Kap. 3) modifiziert, um ein UML-Modell des Profils zu generieren. Dazu wird eine neue EA-Datei generiert, deren Namen sich aus dem Original-Namen und dem Namen des Profils zusammensetzt.
Modellvergleich	Ermöglicht es, zwei Applikationsschemata aus zwei unterschiedlichen UML-Modellen zu vergleichen und die Unterschiede zu protokollieren (s. Kap. 8).
Tagged Values bearbeiten	Ermöglicht es, die Steuerparameter („Tagged Values“) des UML-Modells zu bearbeiten oder zu ergänzen (s. Kap. 7).
Weitere Funktionen	Die in diesem Fall über den Button „Ausführen“ aktivierte Funktion muss über die ComboBox „Funktion“ ausgewählt werden: • „UML-Diagramme exportieren“: Exportiert alle UML-Diagramme des ausgewählten <<ApplicationSchema>> Packages in der Unter-Ordner **Diagrams** des Ziel-Ordners.

Info	Zeigt einen Dialog mit allgemeinen Informationen über die UML-Transformation Software.
Beenden	Schließt die Applikation.

Tabelle 2: Zentrale Benutzerschnittstelle der UML-Transformation Software

2.1Änderung der zentralen Programm-Einstellungen

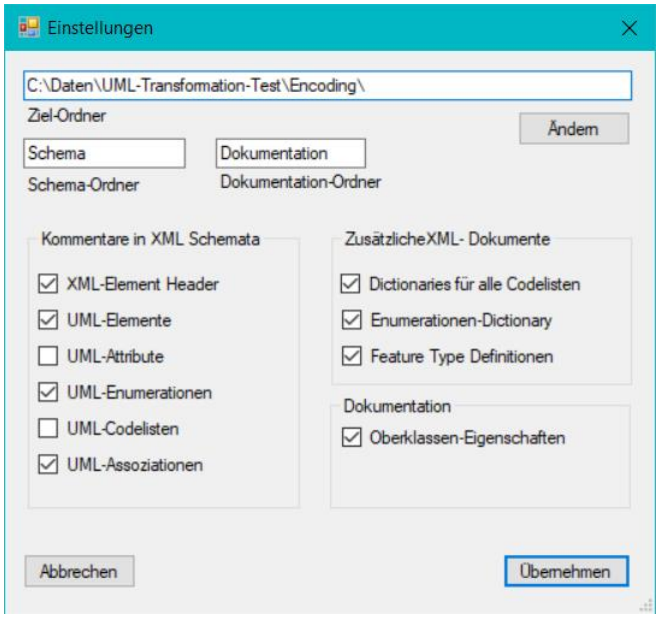


Abbildung 2: Parameter-Einstellung

Ziel-Ordner	Pfadname des Ordners („*Ziel-Ordner*“), in den sämtliche Ausgaben der Software geschrieben werden. Der voreingestellte Ordner (Ordner der EA-Eingabedatei) kann durch Überschreiben im Textfeld oder den Button „Ändern“ geändert werden.
Schema-Ordner	Unter-Ordner des Ziel-Ordners, in den die erzeugten XML-Schemata und verschiedene weitere XML-Dateien (s. Kap. 4) geschrieben werden.
Dokumentation-Ordner	Unter-Ordner des Ziel-Ordners, in den alle für die Dokumentation des Datenmodells notwendige Dateien (s. Kap. 6) geschrieben werden.
XML-Element-Header	Ist die CheckBox aktiviert, wird bei der Erzeugung der XML-Schema Dateien der Name des UML-Elements als Kommentar vor das zugehörige XML-Schema Konstrukt (element oder simpleType) geschrieben.
UML-Elemente	Ist die CheckBox aktiviert, wird bei der Erzeugung der XML-Schema Dateien die Definition (UML-Notes) des UML-Elements als annotation in das zugehörige XML-Schema Konstrukt (element oder simpleType) geschrieben.
UML-Attribute	Ist die CheckBox aktiviert, wird bei der Erzeugung der XML-Schema Dateien die Definition (UML-Notes) der UML-Attribute und Assoziation-Enden von <<FeatureType>>, <<DataType>> und

	<<Type>> UML-Elementen als annotation in das zugehörige XML-Schema Konstrukt (element) geschrieben.
UML-Enumerationen	Ist die CheckBox aktiviert, wird bei der Erzeugung der XML-Schema Dateien der einem bestimmten Enumerations-Code (UML-Attribut Initial Value) zugeordnete Langname (UML-Attribut Name) als Kommentar vor des zugehörige XML-Schema Konstrukt (enumeration) geschrieben.
UML-Codelisten	Ist die CheckBox aktiviert, wird bei der Erzeugung der XML-Schema Dateien für UML-Attribute, deren Wertebereich über eine Codeliste (Stereotype <<CodeList>>) definiert ist, der Name dieser Codeliste als annotation in das zugehörige XML-Schema Konstrukt (element) geschrieben.
UML-Assoziationen	Ist die CheckBox aktiviert, wird bei der Erzeugung der XML-Schema Dateien bei Assoziations-Enden der Name des referierten UML Elements als annotation (targetElement) in das zugehörige XML-Schema Konstrukt (element) geschrieben.
Dictionaries für alle Codelisten	Ist diese CheckBox aktiviert, wird im Unter-Ordner **Codelists** des Schema-Ordners für jedes UML-Element vom Stereotype <<Codelist>> ein GML-Dictionary gleichen Namens erzeugt, das die im UML-Element definierten Codes enthält.
Enumerationen Dictionary	Ist diese CheckBox aktiviert, wird im Unter-Ordner **Enumerationen** des Schema-Ordners ein GML-Dictionary erzeugt, das alle UML-Elemente von Stereotype <<Enumeration>> als DefinitionCollection abbildet. In jeder dieser Collections sind die einzelnen Enumerations-Items durch ihren Code (name, identifier), und die zugehörige Definition (description) repräsentiert.
Feature Type Definitionen	Ist diese CheckBox aktiviert, wird für jedes UML-Element mit Stereotype <<FeatureType>> im Unter-Ordner **Definitions** des Schema-Ordners eine XML-Datei gleichen Namens generiert. Diese Datei dokumentiert das UML-Element entsprechend des XML-Schemas **ShapeChangeDefinitions.xsd** (Unter-Ordner **XSD** des System-Ordners).
Oberklassen-Eigenschaften	Ist diese CheckBox aktiviert, wird bei der Erzeugung der Objektartenkataloge (s. Kap. 6.1 und 6.2) für jedes UML-Element mit Stereotype <<FeatureType>>, <<DataType>> oder <<Type>> eine vollständige Liste von Attributen und Relationen erzeugt, die auch die geerbten Attribute / Relationen von Oberklassen enthält.
Abbrechen	Beendet die Dialogbox, ohne die Änderungen zu übernehmen.
Beenden	Beendet die Dialogbox und übernimmt die Änderungen.

Tabelle 3: Einstellungs-Parameter der Software UML-Transformation

3 Definition von Profilen eines GML-Applikationsschemas

Unter einem Profil eines GML-Applikationsschemas („*Ausgangs-Schema*“) wird allgemein eine (formell oder informell) definierte Verschärfung der Schemaregeln verstanden. Damit muss jedes Instanzdokument, dass gegen ein Profil valide ist, auch bzgl. des Ausgangs-Schemas valide sein.

Die Software UML-Toolbox gestattet es,

- bestimmte Profilregeln über Tagged Values im UML-Modell des Ausgangs-Schemas zu definieren (Kap. 3.1),
- automatisch zu überprüfen, ob die definierten Profilregeln in sich konsistent sind (Kap. 3.2, und
- auf Basis der definierten Regeln XML-Schema-Dateien (*Profil-Schema*) und Objektartenkataloge für das Profil zu generieren.

Damit kann die Einhaltung von Profilregeln in einem Instanzdokument durch eine Validierung gegen das Profil-Schema überprüft werden. Profilregeln, die nicht über die Syntax von XML-Schema ausgedrückt werden können, werden von der Software nicht unterstützt.

3.1 Definition von Profilregeln

Profilregeln werden mit Hilfe von Tagged Values definiert, die UML-Paketen, UML-Elementen, UML-Attributen und UML-Assoziations-Enden zugeordnet werden können. Dabei gilt:

- Als Tagged Value **Name** wird ein beliebiger, dem zu definierenden Profil zugeordneter Text verwendet. Die Aggregation aller Tagged Values mit gleichem Namen definiert damit die Regeln eines bestimmten Profils.
- Der Tagged Value **Wert** definiert die für das jeweilige UML-Konstrukt definierte Einschränkung des Ausgangs-Schemas. Die für die einzelnen UML-Konstrukte zulässigen Werte und ihre Bedeutung sind in Tabelle 4 aufgeführt.

Tagged Value Wert	**Zugeordnet zu**	**Bedeutung und Auswirkung auf die Transformation UML → XML-Schema / Objektartenkatalog**
excludedInProfile	UML-Paket	Keines der UML-Elemente des UML-Pakets darf im zugeordneten Profil verwendet werden → Die UML-Elemente werden nicht in das XML-Schema und den Objektartenkatalog transformiert.
excludedInProfile	UML-Element	Das UML-Element darf im zugeordneten Profil nicht verwendet werden → Das UML-Element wird nicht in das XML-Schema und den Objektartenkatalog transformiert.
excludedInProfile	UML-Attribut	Das UML-Attribut darf im zugeordneten Profil nicht verwendet werden → Das UML-Attribut wird nicht in das XML-Schema und den Objektartenkatalog transformiert.
excludedInProfile	UML-Assoziations-Ende	In der UML-Assoziation darf diese Richtung im zugeordneten Profil nicht belegt werden → Das Assoziations-Ende wird nicht in das XML-Schema und den Objektartenkatalog transformiert.
mandatoryInProfile	UML-Attribut	Das UML-Attribut ist im zugeordneten Profil verpflichtend zu belegen → minOccurs="1" im erzeugten XML-Schema.
mandatoryInProfile	UML-Assoziations-Ende	In der UML-Assoziation ist diese Richtung im zugeordneten Profil verpflichtend zu belegen

		→ minOccurs="1" im erzeugten XML-Schema.

Tabelle 4: Tagged Values zur Definition von Profilen

3.2 Konsistenz der definierten Profilregeln

Jedes Instanzdokument, das bezüglich eines Profils des Applikationsschemas valide ist, muss auch gegen das Applikationsschema selbst validieren. Dies bedingt eine Reihe von Einschränkungen, die bei der Definition von Profilregeln zu beachten sind:

- Wenn ein UML-Element vom Stereotype <<FeatureType>>, <<DataType>> oder <<Type>> in einem Profil ausgeschlossen ist, dann müssen auch alle von diesem UML-Element (direkt oder indirekt) abgeleitete UML-Elemente ausgeschlossen sein.
- Wenn in einem UML-Element des Profils mit Stereotype <<FeatureType>>, <<DataType>> oder <<Type>> ein Attribut ausgeschlossen ist, dann muss dies Attribut im Basisschema als „optional" (Kardinalität [0..1] oder [0..*]) gekennzeichnet sein.
- Wenn in einem UML-Element des Profils mit Stereotype <<FeatureType>>, <<DataType>> oder <<Type>> ein Assoziations-Ende ausgeschlossen ist, dann muss dies Assoziations-Ende im Basisschema als „optional" (Kardinalität [0..1] oder [0..*]) gekennzeichnet sein.
- UML-Attribute in einem Profil dürfen keinen Datentyp haben, der in diesem Profil als „ausgeschlossen" deklariert ist.
- UML-Assoziations-Enden in einem Profil dürfen auf kein UML-Element verweisen, der in diesem Profil als „ausgeschlossen" deklariert ist.
- In einem UML-Element des Profils mit Stereotype <<Enumeration>> dürfen nicht alle Enumerations-Einträge ausgeschlossen werden.

4 Transformation UML → XML-Schema

Bevor die Transformation des UML-Modells in ein oder mehrere XML-Schema Dateien gestartet werden kann, müssen (Abbildung 1):

- Das zu transformierende Applikationsschema über die ComboBox „Applikation Schema" ausgewählt werden,
- Die diesem Applikationsschema zugrundeliegende GML-Version über die ComboBox „GML-Version" festgelegt werden (aktuell werden nur GML 3.1.1 und GML 3.2.1 unterstützt), sowie eventuell
- Der Name des Profils, für das XML-Schema Dateien erzeugt werden sollen, im Textfeld „Profil" eingetragen werden.

Die Software setzt voraus, dass das benutzte UML-Modell in sich konsistent ist, und das in der ISO 19136 definierte UML-Profil eingehalten wird. Die daraus resultierenden Anforderungen an das UML-Modell werden in Kap. 4.1 näher beschrieben. Bei der Erzeugung der Schema-Dateien werden die in der ISO Norm definierten Abbildungsregeln verwendet. Näheres zur Transformation UML → XML-Schema findet sich in den Kapiteln 4.2 - 4.5.

4.1 Anforderungen an das EA UML-Modell

4.1.1 Unterstützte Stereotypes

UML-Paketen kann optional ein Stereotype zugeordnet werden. Ist dies der Fall, darf nur einer der in Tabelle 5 aufgeführten Stereotypes verwendet werden.

<<ApplicationSchema>>	Nur UML-Pakete mit diesem Stereotype können als Wurzelelement für UML-Transformationen verwendet werden.
<<Leaf>>	Ein UML-Paket mit diesem Stereotype darf nur noch UML-Elemente und -Diagramme, aber keine weiteren UML-Pakete enthalten.

Tabelle 5: Stereotypes von UML-Packages

Jedem UML-Element muss ein Stereotype zugeordnet werden, wobei nur die in Tabelle 6 aufgeführten verwendet werden dürfen.

<<FeatureCollection>>	Wird auf eine gml:FeatureCollection abgebildet.
<<FeatureType>>	Wird verwendet für UML-Elemente, die Objekte der realen Welt abbilden.
<<Type>>	Wird verwendet für UML-Elemente, die komplexe Datentypen abbilden. In diesen Fall kann das Datenobjekt im Instanzdokument in andere komplexe Datenobjekte eingebettet werden, oder es kann über den xlink-Mechanismus referenziert werden.
<<DataType>>	Wird benutzt für komplexe Datentypen, die in andere komplexe Datentypen eingebettet werden können, aber nicht über xlink referenziert werden können.
<Enumeration>>	Wird benutzt für Aufzählungen von *Codes* oder *Code* / *Name* Paaren, die im Schema gespeichert werden.
<<CodeList>>	Wird benutzt für Aufzählungen von *Codes*, die extern gespeichert werden.
<ADEElement>>	Wird benutzt für Erweiterungen der Attribute oder Assoziations-Enden eines UML-Elements gemäß des „Application Domain Extension“ Mechanismus.

Tabelle 6: Unterstützte Stereotypes von UML-Elementen

4.1.2 Unterstützte Tagged Values

Tabelle 7 - Tabelle 9 dokumentieren alle von der Software unterstützte Tagged Values mit:

- Dem Tagged Value **Name**;
- Der Angabe, ob der Parameter für die korrekte Ableitung der XML-Schemata verpflichtend belegt werden muss;
- Der Angabe, für welche UML-Stereotypes der Tagged Value relevant ist;
- Der Angabe des Datentyps vom Tagged Value **Wert**.

Tagged Value Name	Notwendig	Relevant für Stereotypes	Typ des Werts
noPropertyType	•	<<FeatureType>>, <<DataType>>, <<Type>>	true oder false
byValuePropertyType	•	<<FeatureType>>, <<DataType>>, <<Type>>	true oder false

adeHook		<<FeatureType>>, <<DataType>>, <<Type>>	true oder false
deviantPropertyTypeName		<<FeatureType>>, <<DataType>>, <<Type>>	Text
deviantTypeName		<<FeatureType>>, <<DataType>>, <<Type>>	Text

Tabelle 7: Unterstützte Tagged Values von UML-Elementen

Tagged Value Name	Notwendig	Relevant für Stereotypes	Typ des Werts
targetNamespace	●	<<ApplicationSchema>>	URL
version	●	<<ApplicationSchema>>	Text
xmlns	●	<<ApplicationSchema>>	Text
xsdDocument	(●)	Alle Stereotypes; notwendig für <<ApplicationSchema>>	XML-Schema Dateiname

Tabelle 8: Unterstützte Tagged Values von UML-Paketen

Tagged Value Name	Notwendig	Relevant für Stereotypes	Typ des Werts
inlineOrByReference	●	<<FeatureType>>, <<Type>>	*inline, byReference, inlineOrByReference*
sequenceNumber	●	<<FeatureType>>, <<DataType>>, <<Type>>	Integer Zahl, innerhalb des UML-Elements eindeutig

Tabelle 9: Unterstützte Tagged Values für Attribute und Assoziations-Enden von UML-Elementen

4.2 Transformation von UML Paketen

Aus UML-Paketen mit Stereotype <<ApplicationSchema>> wird bei der Transformation grundsätzlich ein XML-Schema in einer separaten Datei erzeugt, deren Name durch den Wert des Tagged Value xsdDocument festgelegt wird. Der Header der erzeugten Schema-Datei wird wie folgt gebildet:

- Default-Namespace (Attribut xmlns) ist grundsätzlich der XML-Schema Namespace (http://www.w3.org/2001/XMLSchema).
- Als Target-Namespace (Attribut targetNamespace) wird der Wert des Tagged Value targetNamespace verwendet.
- Derselbe Namespace wird auch dem durch den Tagged Value xmlns definierten Namespace-Kürzel zugewiesen.
- Der verwendete GML-Namespace (Attribut xmlns:gml) richtet sich nach der über die Benutzeroberfläche spezifizierten GML-Version (s. Tabelle 2). Bei einem Applikationsschema für GML 3.1.1 ist der Namespace http://www.opengis.net/gml, bei GML 3.2.1 http://www.opengis.net/gml/3.2.

- Die Version des Schemas (Attribut version) wird über den Tagged Value version festgelegt.
- Das Attribut elementFormDefault hat grundsätzlich den Wert qualified.

<<AplicationSchema>> Packages können in beliebiger Schachtelungstiefe weitere Packages enthalten, denen über den Tagged Value xsdDocument optional eigene Schema-Dateien zugewiesen sein können. Beim XML-Schema Encoding werden diese Dateien automatisch erzeugt und per include oder import in die Basisdatei eingefügt. Falls ein untergeordnetes UML-Package nicht den Stereotype <<ApplicationSchema>> hat, werden dabei Target-Namespace, Kürzel und Version des Basisschemas verwendet. Ist dem untergeordneten UML-Package keine Schemadatei zugewiesen, wird der Inhalt direkt in die Schema-Datei des Basisschemas transformiert.

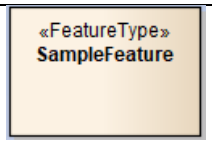
4.3 Transformation UML-Elemente

4.3.1 Basis UML-Elemente vom Stereotype <<FeatureType>>

Tabelle 10 zeigt das XML-Schema Encoding von <<FeatureType>> Elementen, die nicht von einem anderen UML-Element abgeleitet sind, in Abhängigkeit vom Wert verschiedener Tagged Values. In allen Varianten wird dabei die GML-Version 3.2.1 benutzt. Die in der ersten Zeile gezeigte Variante (byValuePropertyType = false, noPropertyType = false) ist der in den meisten Anwendungsschemata verwendete Standardfall. Die Variante byValuePropertyType = false, noPropertyType = true kann verwendet werden, wenn das zugehörige XML element grundsätzlich nur über ein xlink:href referiert werden soll (s. Kap. 4.4.), was z. B. bei der Standards XPlanGML und ALKIS der Fall ist. Die Angabe byValuePropertyType = true ist im umgekehrten Fall notwendig, wenn erzwungen werden soll, dass das XML element nicht über ein xlink referiert werden kann, sondern in das referierende Objekt eingebettet werden muss.

Die Verwendung des Tagged Values „adeHook = true“ führt dazu, dass am Ende des erzeugten XML complexType ein spezielles XML element erzeugt wird, das als Anknüpfungspunkt für Erweiterungen nach der Application Domain Extension (ADE) Methode dient („*ADE-Hook*“).

Falls das GML-Applikationsschema sich nicht an die übliche Namenskonvention zur Bezeichnung des Property-Types hält (Suffix PropertyType), kann der zu verwendende Name über den Tagged Value deviantPropertyTypeName festgelegt werden. In ähnlicher Art und Weise kann über deviantTypeName ein spezieller Name des generierten complexType festgelegt werden, wenn die Namenkonvention (Suffix Type) im Applikationsschema nicht eingehalten wurde.

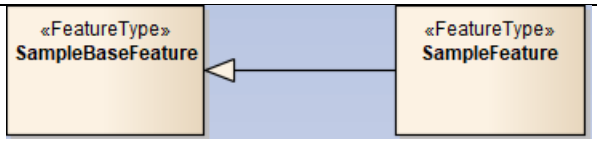
	
byValuePropertyType = false noPropertyType = false	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractFeatureType" /> </complexContent> </complexType> <complexType name="SampleFeaturePropertyType">

	<sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType>
byValuePropertyType = false noPropertyType = true	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractFeatureType" /> </complexContent> </complexType>
byValuePropertyType = true noPropertyType = true	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractFeatureType" /> </complexContent> </complexType> <complexType name="SampleFeaturePropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType> <complexType name="SampleFeaturePropertyByValueType"> <sequence> <element ref="spl:SampleFeature" /> </sequence> </complexType>
byValuePropertyType = true noPropertyType = true	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractFeatureType" /> </complexContent> </complexType> <complexType name="SampleFeaturePropertyByValueType"> <sequence> <element ref="spl:SampleFeature" /> </sequence> </complexType>
byValuePropertyType = false noPropertyType = false adeHook = true	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractFeatureType"> <sequence> <element ref="spl:_GenericApplicationPropertyOfSampleFeature" minOccurs="0" maxOccurs="unbounded" />

	</sequence> </extension> </complexContent> </complexType> <element name="_GenericApplicationPropertyOfSampleFeature" type="anyType" abstract="true" /> <complexType name="SampleFeaturePropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType>
byValuePropertyType = false noPropertyType = false deviantPropertyType = SampleFeatureOtherPropertyType	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractFeatureType" /> </complexContent> </complexType> <complexType name="SampleFeatureOtherPropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType>

Tabelle 10: XML-Schema Encoding von UML Basis-Elementen mit Stereotype <<FeatureType>>

Tabelle 11 zeigt für den Standardfall (byValuePropertyType = false, noPropertyType = false) das XML-Schema Encoding eines abgeleiteten <<FeatureType>>.

	
byValuePropertyType = false noPropertyType = false	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="spl:SampleBaseFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="spl:SampleBaseFeatureType" /> </complexContent> </complexType> <complexType name="SampleFeaturePropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType>
byValuePropertyType = false	<element name="SampleBaseFeature" type="spl:SampleBaseFeatureOtherName" substitutionGroup="gml:AbstractFeature" />

noPropertyType = false deviantTypeName = SampleBaseFeatureOtherName (in SampleBaseFeature)	<complexType name="SampleBaseFeatureOtherName"> <complexContent> <extension base="gml:AbstractFeatureType" /> </complexContent> </complexType> <complexType name="SampleBaseFeaturePropertyType"> <sequence> <element ref="spl:SampleBaseFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType> <element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="spl:SampleBaseFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="spl:SampleBaseFeatureOtherName" /> </complexContent> </complexType> <complexType name="SampleFeaturePropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType>

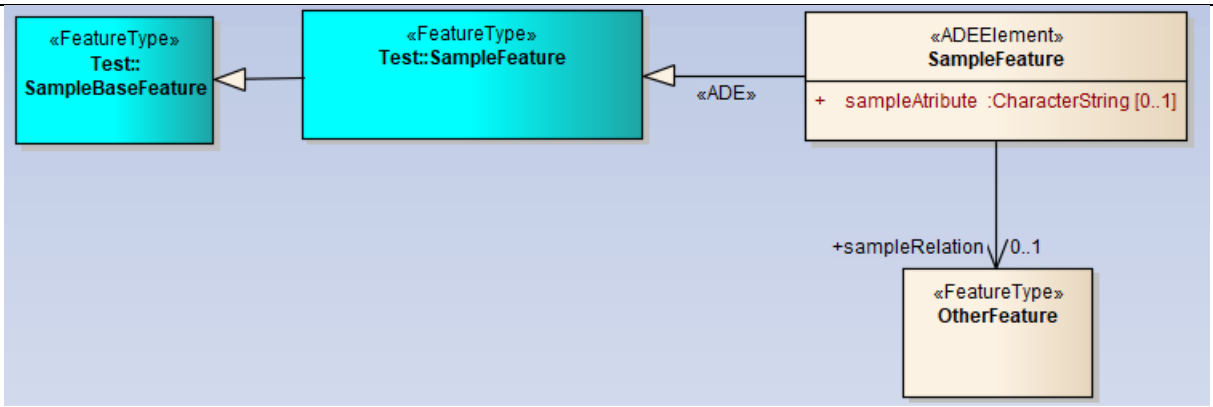
Tabelle 11: XML-Schema Encoding von abgeleiteten UML-Elementen mit Stereotype <<FeatureType>>

4.3.2 UML-Elemente vom Stereotype <<ADEElement>>

UML Elemente mit Stereotype <<ADEElement>> dienen dazu, unter Ausnutzung des Application Domain Extension (ADE) Mechanismus zusätzliche Properties einer vorhandenen Klasse eines Applikationsschemas (Basisschema) zu definieren. Dazu muss das UML Modell die folgenden Voraussetzungen erfüllen:

- Das <<ADEElement>> muss in einem eigenen UML-Paket vom Stereotype <<ApplicationSchema>> definiert sein, das vom Basisschema abweichende Tagged Values für Namespace, Namespace-Kürzel und Schema-Dateiname hat.
- Das <<ADEElement>> muss von einer gleichnamigen Klasse (Stereotype <<FeatureType>>, <<DataType>> oder <<Type>>) des Basisschemas abgeleitet werden.
- Die XML-Encoding der Oberklasse muss den ADE-Hook (s. Kap. 4.3.1) enthalten.
- Gibt es im ADE-Schema mehrere Elemente mit Stereotype <<ADEElement>>, so müssen die darin definierten Attribute und Assoziations-Enden jeweils unterschiedliche Namen haben.

Tabelle 12 zeigt das XML-Encoding eines <<ADEElement>>.

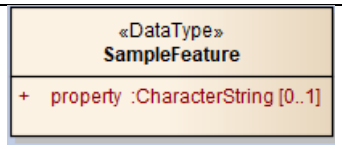



```
<element name="sampleAtribute" type="string"
substitutionGroup="spl:_GenericApplicationPropertyOfSampleFeature" />
  <element name="sampleRelation" type="apl-ade:OtherFeaturePropertyType"
substitutionGroup="spl:_GenericApplicationPropertyOfSampleFeature">
```

Tabelle 12: XML-Schema Encoding von UML-Elementen mit Stereotype <<ADEElement>>

4.3.3 UML-Elemente vom Stereotype <<DataType>>

Tabelle 13 und Tabelle 14 zeigen das XML-Schema Encoding von <<DataType>> Elementen. Im Regelfall wird mit byValuePropertyType = false und noPropertyType = false gearbeitet.

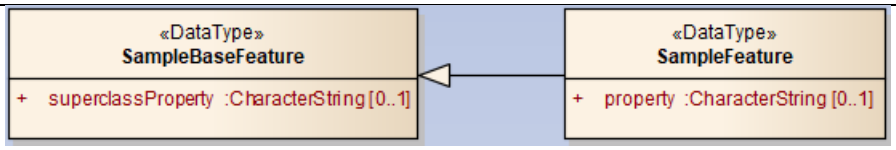


byValuePropertyType = false

noPropertyType = false

```
<element name="SampleFeature"
type="spl:SampleFeatureType"/>
<complexType name="SampleFeatureType">
   <sequence>
     <element name="property" minOccurs="0" type="string" />
   </sequence>
</complexType>
<complexType name="SampleFeaturePropertyType">
   <sequence>
     <element ref="spl:SampleFeature" minOccurs="0" />
   </sequence>
</complexType>
```

Tabelle 13: XML-Schema Encoding von UML Basis-Elementen mit Stereotype <<DataType>>



byValuePropertyType = false

noPropertyType = false

```
<element name="SampleFeature" type="spl:SampleFeatureType"
substitutionGroup="spl:SampleBaseFeature" />
<complexType name="SampleFeatureType">
   <complexContent>
     <extension base="spl:SampleBaseFeatureType">
       <sequence>
         <element name="property" minOccurs="0" type="string" />
       </sequence>
     </extension>
```

	</complexContent> </complexType> <complexType name="SampleFeaturePropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> </complexType>

Tabelle 14: XML-Schema Encoding von abgeleiteten UML-Elementen mit Stereotype <<DataType>>

4.3.4 UML-Elemente vom Stereotype <<Type>>

Das XML-Schema Encoding von UML-Elementen mit Stereotype <<Type>> (s. Tabelle 15) ist weitgehend identisch mit dem Encoding von <<FeatureType>> Elementen (s. Kap. 4.3.1). Der einzige Unterschied ist, dass <<Type>> Basis-Elemente im GML-Schema von gml:AbstractGML abgeleitet werden, während <<FeatureType>> Elemente von gml:AbstractFeature abgeleitet werden.

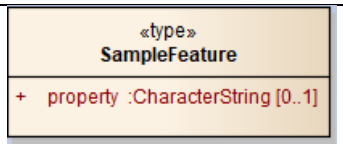
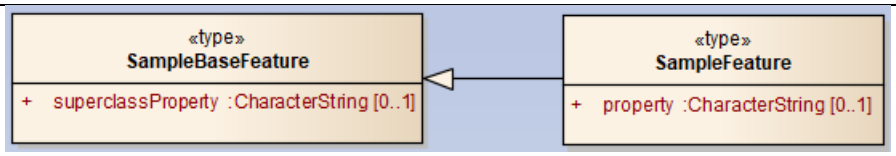
 	
byValuePropertyType = false noPropertyType = false	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractGML" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractGMLType"> <sequence> <element name="property" minOccurs="0" type="string" /> </sequence> </extension> </complexContent> </complexType> <complexType name="SampleFeaturePropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType>

Tabelle 15: XML-Schema Encoding von UML Basis-Elementen mit Stereotype <<Type>>

 	
byValuePropertyType = false noPropertyType = false	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="spl:SampleBaseFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="spl:SampleBaseFeatureType"> <sequence> <element name="property" minOccurs="0" type="string" /> </sequence> </extension> </complexContent> </complexType>

	<complexType name="SampleFeaturePropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType>

Tabelle 16: XML-Schema Encoding von abgeleiteten UML-Elementen mit Stereotype <<DataType>>

4.3.5 UML-Elemente vom Stereotype <<Enumeration>>

UML-Elemente vom Stereotype <<Enumeration>> dienen dazu, den diskreten Wertebereich bestimmter Attribute innerhalb des XML-Schemas zu spezifizieren. In vielen Anwendungsschemata werden die erlaubten Attributwerte durch Paare von **Code** und **Name** spezifiziert:

- Der **Code** ist ein kurzer, abstrakter Text, der als Attributwert benutzt werden darf.
- Der **Name** ist ein längerer und aussagekräftiger Text, der die semantische Bedeutung des Codes beschreibt. Er ist nur als Kommentar im XML-Schema repräsentiert.

Tabelle 17 zeigt, wie derartige Enumerationen mit dem Werkzeug Enterprise Architect generiert und als XML simpleType repräsentiert werden.

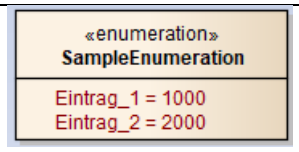
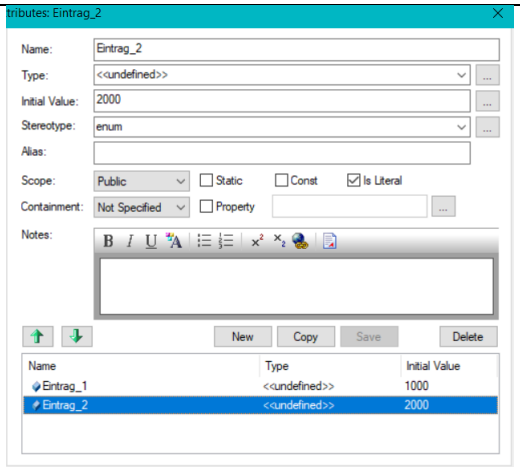
<table>
<tr><td> 
</td></tr>
<tr><td> 
</td></tr>
<tr><td><pre>
<simpleType name="SampleEnumeration">
  <restriction base="string">
    <enumeration value="1000">
      <!--Eintrag_1-->
    </enumeration>
    <enumeration value="2000">
      <!--Eintrag_2-->
    </enumeration>
  </restriction>
</simpleType>
</pre></td></tr>
</table>

Tabelle 17: Generierung und XML-Schema Encoding von UML-Elementen mit Stereotype <<Enumeration>>

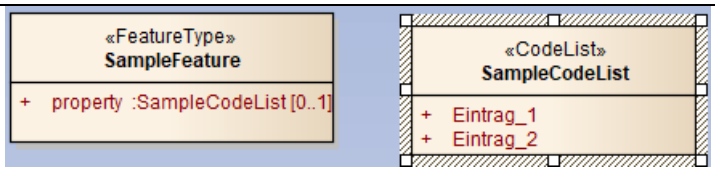
4.3.6 UML-Elemente vom Stereotype <<CodeList>>

Auch über UML-Elemente mit Stereotype <<CodeList>> lässt sich der diskrete Wertebereich von Attributen definieren. Im Gegensatz zu Enumerationen werden die zulässigen Werte aber nicht direkt im XML-Schema gespeichert, sondern extern in einer

sog. Codeliste. Welche Codeliste für ein bestimmtes Attribut benutzt werden soll, wird nicht im erzeugten XML-Schema festgelegt, sondern erst im Instanzdokument. Auch das Datenformat einer Codeliste ist nicht festgelegt.

Tabelle 18 zeigt beispielhaft das XML-Encoding von <<CodeList>> Elementen und von Attributen, deren Datentyp durch ein <<CodeList>> Element beschrieben wird.

- Codelist-Attribute werden beim XML-Encoding grundsätzlich auf den GML-Datentyp gml:CodeType abgebildet.
- Aus jedem <<CodeList>> Element kann bei der UML → XML-Schema Transformation ein GML-Dictionary erzeugt werden.

	
Schema-Encoding SampleFeature	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractFeatureType"> <sequence> <element name="property" minOccurs="0" type="gml:CodeType"> <annotation> <appinfo> <referenzierteCodelist>spl:SampleCodeList</referenzierteCodelist> </appinfo> </annotation> </element> </sequence> </extension> </complexContent> </complexType>
GML-Dictionary SampleCodeList.xml	<Dictionary xmlns="http://www.opengis.net/gml/3.2" xmlns:gml="http://www.opengis.net/gml/3.2" gml:id="SampleCodeList"> <identifier codeSpace="urn:spl">urn:spl:def:codelist:spl:SampleCodeList</identifier> <name>spl:SampleCodeList</name> <dictionaryEntry> <Definition gml:id="S_Eintrag_1"> <description>Eintrag_1</description> <identifier codeSpace="urn:spl">urn:spl:def:codelist:spl:SampleCodeList:Eintrag_1</identifier> <name>Eintrag_1</name> </Definition> </dictionaryEntry> <dictionaryEntry> <Definition gml:id="S_Eintrag_2"> <description>Eintrag_2</description> <identifier codeSpace="urn:spl">urn:spl:def:codelist:spl:SampleCodeList:Eintrag_2</identifier> <name>Eintrag_2</name> </Definition>

	</dictionaryEntry> </Dictionary>

Tabelle 18: XML-Encoding von <<CodeList>> Elementen und Attributen

4.4 Transformation UML-Attribute und -Assoziationen

Auch für Attribute und Assoziations-Enden legt der ISO 19136 Standard eine Anzahl von Tagged Values fest, die die Transformation UML → XML-Schema steuern. Die Software UML-Transformation unterstützt zwei dieser Parameter: sequenceNumber und inlineOrByReference.

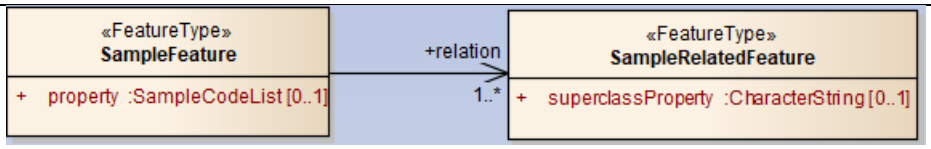
Die sequenceNumber ist grundsätzlich eine Integer-Zahl. Bei UML-Elementen mit Stereotype <<FeatureType>>, <<DataType>> oder <<Type>> muss jedes Attribut und jedes navigierbare, vom Element wegzeigende Assoziations-Ende eine eindeutige sequenceNumber haben. Bei der Erzeugung des XML-Schema complexType wird über diese Nummern die Reihenfolge der untergeordneten element Objekte festgelegt.

Für Elemente mit Stereotype <<FeatureType>>, <<DataType>> oder <<Type>> ist auch der Tagged Value inlineOrByReference relevant. Er muss allerdings nur gesetzt werden, wenn der Datentyp des Attributs oder das Ziel des Assoziations-Endes wiederum ein UML-Element mit Stereotype <<FeatureType>> oder <<Type>> ist.

Der Tagged Value inlineOrByReference darf nur drei Werte annehmen: inline, byReference, oder inlineOrByReference. Wie diese Tagged Values das XML-Encoding einer Relation von einem <<FeatureType>> Element auf ein anderes beeinflusst, ist exemplarisch in Tabelle 19 beschrieben. Bei Attributen / Assoziationsenden mit einfachem Datentyp oder Verweis auf einen <<DataType>> wird automatisch das Encoding nach dem „inline“ Schema (s. u.) verwendet.

Die Auswirkung des Tagged Value inlineOrByReference auf die Struktur von Instanzdokumenten kann man wie so zusammenfassen:

- Im Fall von „byReference“ Encoding werden Relationen zwischen Features immer durch einen xlink repräsentiert. Das GML-Instanzdokument hat eine flache Hierarchie, und es ist durch eine Schemavalidierung nicht nachprüfbar, ob eine Relation auch auf das nach UML-Modell korrekte Ziel-Feature zeigt.
- Im Fall von „inline“ Encoding wird das referierte Feature immer in das referierende Feature eingebettet, so dass die Korrektheit der Relation über eine Schema-Validierung überprüft werden kann. In diesem Fall muss der referierte <<FeatureType>> im UML-Schema den Tagged Value byValuePropertyType = true haben.
- Im Fall von von „inlineOrByReference“ Encoding können im Instanzdokument beide Varianten für Referenzen zwischen Features verwendet werden. In diesem Fall muss der referierte <<FeatureType>> im UML-Schema den Tagged Value noPropertyType = false haben.



inlineOrByReference = inline **Wichtig**:	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractFeatureType">

Das referierte Element muss mit byValuePropertyType = true transformiert werden	<sequence> <element name="property" minOccurs="0" type="gml:CodeType" /> <element name="relation" maxOccurs="unbounded" type="spl:SampleRelatedFeaturePropertyByValueType" /> </sequence> </extension> </complexContent> </complexType> <complexType name="SampleFeaturePropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType> <complexType name="SampleRelatedFeaturePropertyByValueType"> <sequence> <element ref="spl:SampleRelatedFeature" /> </sequence> </complexType>
inlineOrByReference = byReference	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractFeatureType"> <sequence> <element name="property" minOccurs="0" type="gml:CodeType" /> <element name="relation" maxOccurs="unbounded" type="gml:ReferenceType"> <annotation> <appinfo> <gml:targetElement>spl:SampleRelatedFeature</gml:targetElement> </appinfo> </annotation> </element> </sequence> </extension> </complexContent> </complexType> <complexType name="SampleFeaturePropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType>
inlineOrByReference = inline **Wichtig**: Das referierte Element muss mit noPropertyType =	<element name="SampleFeature" type="spl:SampleFeatureType" substitutionGroup="gml:AbstractFeature" /> <complexType name="SampleFeatureType"> <complexContent> <extension base="gml:AbstractFeatureType"> <sequence> <element name="property" minOccurs="0" type="gml:CodeType" /> <element name="relation" maxOccurs="unbounded"

false transformiert werden	type="spl:SampleRelatedFeaturePropertyType"> <annotation> <appinfo> <gml:targetElement>spl:SampleRelatedFeature</gml:targetElement> </appinfo> </annotation> </element> </sequence> </extension> </complexContent> </complexType> <complexType name="SampleFeaturePropertyType"> <sequence> <element ref="spl:SampleFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType> <complexType name="SampleRelatedFeaturePropertyType"> <sequence> <element ref="spl:SampleRelatedFeature" minOccurs="0" /> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup" /> </complexType>

Tabelle 19: XML-Encoding von Relationen auf einen <<FeatureType>>

4.5 Sonderbehandlung XPlanGML

Die XPlanGML Datenmodelle weisen eine Reihe von Besonderheiten auf, die eine direkte Anwendung der ISO 19136 Abbildungsregeln für die Ableitung der XML-Schema Dateien verhindern. Es sind deshalb zwei fest in das Programm eingebaute Transformationsschritte vorgeschaltet, die das XPlanGML UML Modell in ein ISO 19136 konformes UML Modell transformieren.

4.5.1 Schritt 1: Auflösung von Mehrfach-Vererbungen

Einige UML-Elemente der XPlanGML Schemata haben mehrere Oberklassen. Verantwortlich dafür sind die <<Type>> Elemente *BP_GestaltungBaugebiet*, *BP_FestsetzungenBaugebiet* und *BP_ZusaetzlicheFestsetzungen*. Deshalb werden in allen <<FeatureType>> Elementen, die von einer dieser Klassen abgeleitet werden (z.B. *BP_BaugebietsTeilFlaeche* oder *BP_UeberbaubareGrundstuecksFlaeche*) die Attribute der Oberklasse in die Attributliste des Elements kopiert. Die sequenceNumber Tagged Values sind dabei so angelegt, dass bei der XML-Schema Erzeugung die folgende Attribut-Reihenfolge erzwungen wird:

1. Attribute von *BP_GestaltungBaugebiet*
2. Attribute von *BP_FestsetzungenBaugebiet*
3. Attribute von *BP_ZusaetzlicheFestsetzungen*
4. Attribute des abgeleiteten <<FeatureType>>, dessen sequenceNumber nicht unter 200 liegen dürfen.

Nachdem die Mehrfach-Vererbungen auf diese Art und Weise aufgelöst wurde, werden die Original <<Type>> Elemente aus dem temporären UML-Modell gelöscht.

4.5.2 Schritt 2: Korrektur der Geometrieklassen

Die XPlanGML-Datenmodelle verwenden als Geometrieklassen Auswahl-Listen (<<Union>>) von einfachen Geometrietypen (GM_Point, GM_Curve, GM_Surface) und Aggregationstypen (GM_MultiPoint, GM_MultiCurve, GM_MultiSurface). Die ISO 19136

Abbildungsregeln lassen eine direkte XML-Schema Transformation dieser Datentypen nicht zu. In einem vorbereitenden Transformationsschritt werden deshalb die entsprechenden XPlanGML Datentypen (*XP_Punktgeometrie*, *XP_Liniengeometrie*, *XP_Flaechengeometrie*, *XP_VariableGeometrie*) in den ISO Datentyp GM_Objekt konvertiert, der beim XML-Schema Encoding auf den allgemeinen Geometrietyp gml:GeometryPropertyType abgebildet wird. Danach werden die XPlanGML Geometrieelemente aus dem temporären UML-Modell gelöscht.

5 Prüfung und Korrektur des UML-Modells

Nach Aktivierung dieser Funktion öffnet sich der in Abbildung 3 gezeigte Dialog, über den der Umfang der durchzuführenden Prüfungen und Korrekturen spezifiziert werden kann (s. Tabelle 20).

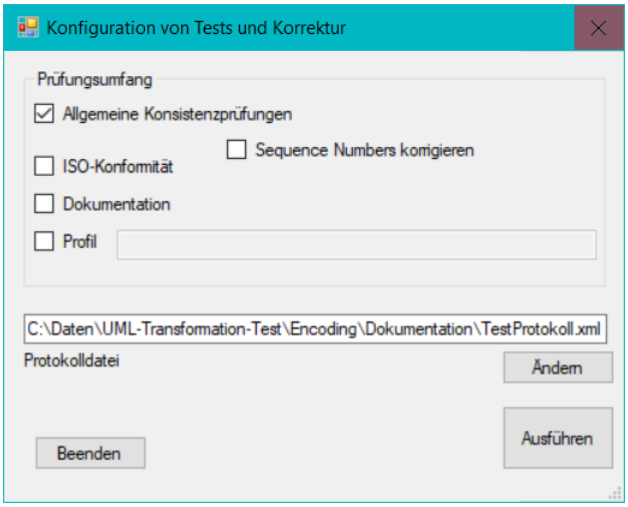


Abbildung 3: Dialogbox „Konfiguration von Prüfung und Korrektur“

Allgemeine Konsistenzprüfungen Betrifft nur <<FeatureType>>, <<DataType>> und <<Type>> Elemente	• Existenz und Eindeutigkeit von sequenceNumber Tagged Values. • Die Datentypen von Attributen und Assoziations-Enden müssen definiert sein. • Attribute und navigierbare Assoziations-Enden müssen eine definierte Kardinalität haben. • Assoziationen müssen mindestens ein navigierbares Ende haben, dem ein Rollen-Name zugewiesen ist.
Sequence Numbers korrigieren	Wenn diese CheckBox aktiviert ist, werden bei der allgemeinen Konsistenzprüfung fehlende oder doppelte sequenceNumber Tagged Values automatisch korrigiert.
ISO-Konformität	• Überprüfung der korrekten Verwendung der Stereotype Bezeichnungen (Tabelle 5 und Tabelle 6). • Überprüfung der korrekten Verwendung der als „notwendig“ gekennzeichneten Tagged Values (Tabelle 7 - Tabelle 9).
Dokumentation	Überprüfung. ob allen UML-Elementen, Attributen, navigierbaren Assoziations-Enden und Enumerations-Einträgen ein Definitionstext in den UML-Notes bzw. Role-Notes zugeordnet ist.

Profil	Überprüft für das im Textfeld spezifizierte Profil, ob die in Kap. 3.2 beschriebenen Bedingungen erfüllt sind.
Protokolldatei	Name der XML-Datei mit den Prüfergebnissen. Standardmäßig hat sie den Namen „TestProtokoll.xml“ und liegt im Dokumentations-Ordner. Über den Button „Ändern“ kann ein anderer Dateiname und Speicherort gewählt werden.
Ausführen	Startet die Überprüfung / Korrektur und erzeugt die Protokolldatei.
Beenden	Schließt den Dialog

Tabelle 20: Einstellung der Modell-Überprüfungen und Korrekturen

6 Erzeugung von Dokumentation

Wenn in der Benutzeroberfläche ein Profil angegeben wurde, bezieht sich der Inhalt der erzeugten Dokumentation nur auf die UML-Elemente, Attribute und Relationen, die in diesem Profil unterstützt werden. Inhalt und Struktur der erzeugten Dokumentation werden über die ComboBox „Struktur Objektartenkatalog“ festgelegt (s. Tabelle 2). Wenn die CheckBox „UML Modell Profil generieren“ aktiviert ist, wird für dies Profil im Ordner des Ausgangsmodells zusätzlich noch ein eigenes EA UML-Modell generiert.

6.1 Objektartenkatalog mit verteilten HTML-Dateien

Nach Starten der Funktion über den Button „Ausführen“ (s. Abbildung 1) wird der in Abbildung 4 gezeigte Dialog aktiv, über den der Inhalt der erzeugten HTML-Dokumente beeinflusst werden kann.

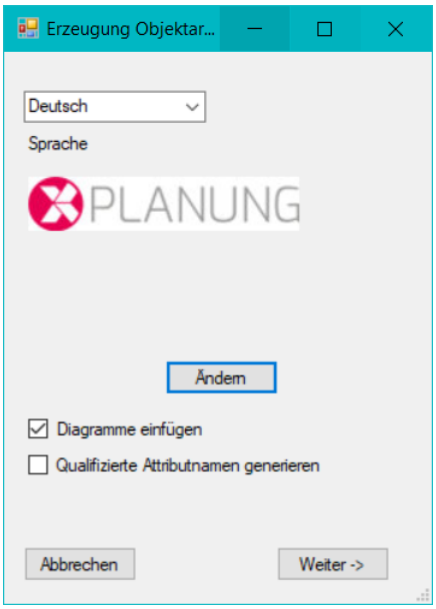


Abbildung 4: Parameter für die Erzeugung eines Objektartenkataloges

Sprache	Auswahl der im Objektartenkatalog verwendeten Sprache (Deutsch oder Englisch), soweit diese nicht durch den Inhalt des EnrepriseArchitect Modelles bestimmt ist.

Ändern	Ermöglicht die Festlegung einer Rasterbild-Datei, z.B. im JPG- oder PNG-Format, die als Logo in den Objektartenkatalog übernommen wird. Der Inhalt der Datei wird angezeigt.
Diagramme einfügen	Wenn die CheckBox aktiviert ist, werden alle Diagramme im UML-Modell mit in der Objektartenkatalog aufgenommen.
Qualifizierte Attributnamen generieren	Wenn die CheckBox aktiviert ist, wird alle Attributnamen in Objektartenkatalog mit dem Kürzel des zugehörigen Namespace qualifiziert.
Abbrechen	Schließt den Dialog, ohne die Generierung des Objektartenkatalogs zu starten.
Weiter →	Schließt den Dialog und startet die Generierung des Objektartenkataloges.

Tabelle 21: Parameter zur Steuerung der Objektartenkatalog-Erzeugung

Die Erzeugung des HTML-gestützten Objektartenkatalogs erfolgt in mehreren Schritten.

- Zunächst wird der HTML-Frameset, der den Zugriff auf den verteilten Objektartenkatalog erlaubt, direkt im HTML-Format aus dem UML-Modell abgeleitet.
- Danach wird für alle einzelnen Bestandteile des UML-Modells (s.u.) jeweils eine spezielle XML-Datei (im Format **FeatureCatalogue.xsd**) mit den notwendigen Informationen abgeleitet:
  - Alle einzelnen UML-Packages.
  - Alle einzelnen UML-Elemente, bei XPlanGML werden dabei vorher die Mehrfach-Vererbungen, wie in Kap. 4.5.1 beschrieben, aufgelöst.
  - Eine Liste aller <<FeatureType>>, <<DataType>> und <<Type>> Element-Namen.
- Mit Hilfe spezieller XSLT-Stylesheets (s. Tabelle 22) werden abschließend die XML-Dateien in HTML-Dateien transformiert.

	Deutsch	**Englisch**
Liste der UML-Element Namen	**stylesheetFeatureList.xsl**	**stylesheetFeatureListEnglisch.xsl**
UML-Packages	**stylesheetPackage.xsl**	**stylesheetPackageEnglisch.xsl**
UML-Elemente	**stylesheetObjektart.xsl**	**stylesheetObjektartEnglisch.xsl**

Tabelle 22: XSLT-Stylesheets zur Erzeugung des verteilten HTML-Objektartenkatalogs

6.2 Objektartenkatalog in einer HTML-Datei

Auch in diesem Fall wird als Erstes der in Abbildung 4 bzw. Tabelle 21 beschriebene Dialog aktiv. Es sollte beachtet werden, dass das Einfügen aller UML-Diagramme (CheckBox „Diagramme einfügen“) zu einer sehr großen und evtl. unübersichtlichen Ausgabedatei führen kann.

Die Erzeugung des Objektartenkataloges ist analog zu dem in Kap. 6.1 beschriebenen Prozess mit dem einzigen Unterschied, dass jetzt alle Ausgaben in einer XML-Datei (**FeatureCatalogue.xml**) zusammengefasst sind. Mit Hilfe der XSLT-Stylesheets **StylesheetTotal.xsl** bzw. **StylesheetTotalEnglish.xsl** wird daraus eine HTML-Datei (**FeatureCatalogue.html**) generiert. Falls der Objektartenkatalog in einem anderen Dokumentenformat benötigt wird, muss dies mit Standard-Textverarbeitungssystemen (z.B. MS-Word) manuell erzeugt werden.

6.3 FeatureType Definitionen als Excel-Tabelle

In diesem Fall wird ein Teil der Information des HTML-gestützten Objektartenkatalogs in Form einer Excel-Tabelle (im Excel-XML Format) aus dem UML-Modell exportiert. In der erzeugten EXCEL-Datei gibt es für jedes UML-Paket eine eigene Arbeitsmappe. In jeder dieser Arbeitsmappen gibt es drei Spalten:

- In der ersten Spalte sind die Namen aller UML-Elemente des Pakets aufgeführt, unabhängig von ihrem Stereotype.
- In der zweiten Spalte befinden sich, in der Reihenfolge des XML-Schemas, die Namen aller Attribute bzw. Rollen-Namen von Assoziations-Enden des zugehörigen UML-Elements, bzw. die Namen der Enumerations-Einträge bzw. Codelist-Einträge.
- In der dritten Spalte sind die zugehörigen Definitionen (UML-Notes bzw. Role-Notes) aufgeführt.

7 Bearbeitung von Tagged Values

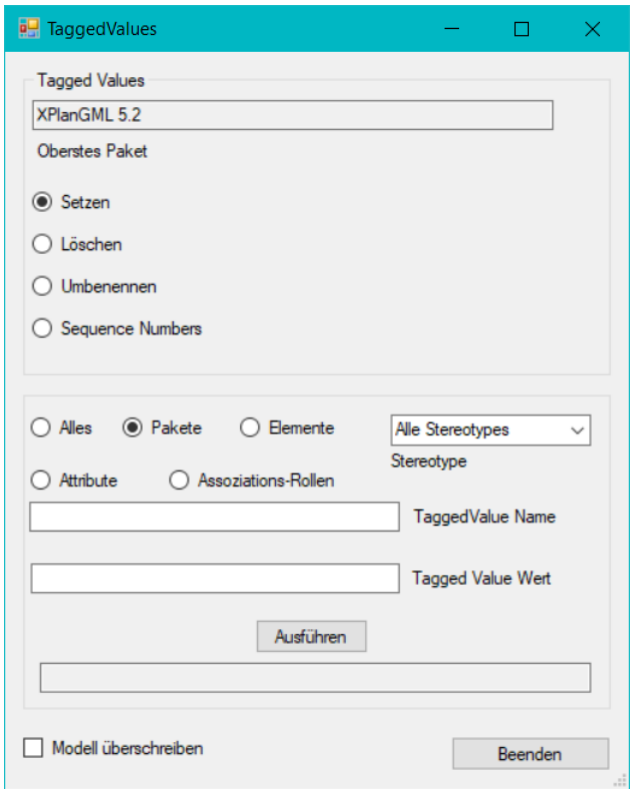


Abbildung 5: Bearbeitung von Tagged Values

Über den in Abbildung 5 gezeigten Dialog ist es möglich, die Tagged Values des ausgewählten GML-Applikationsschemas global zu bearbeiten. Dazu muss als erstes über die entsprechenden RadioButtons die Bearbeitungsaktion ausgewählt werden:

- Setzen von neuen Tagged Values; In den Textfeldern müssen der zu setzende Tagged Value Name und der zugehörige Tagged Value Wert angegeben werden.
- Löschen existierender Tagged Values; Im Textfeld muss der zu löschende Tagged Value Name angegeben werden.
- Umbenennen eines Tagged Values, ohne seinen Wert zu ändern: In den Textfeldern müssen der Name des umzubenennenden Tagged Value (Tagged Value Name) und der neue Name (Tagged Value Name neu) angegeben werden.

- Mit der Auswahl Sequence Numbers werden eindeutige sequenceNumber Tagged Values für UML-Attribute und Assoziations-Enden generiert. Ausgenommen davon sind UML-Elemente vom Stereotype <<Enumeration>> und <<CodeList>>. Eventuell schon vorhandene sequenceNumber Tagged Values werden vorher gelöscht. Die generierten Werte beginnen mit dem im Textfeld eingegebenen Wert, der eine Integer-Zahl sein muss.

Bis auf die sequenceNumber Generierung muss für alle Bearbeitungsaktionen angegeben werden, auf welche Teile des UML-Modells sie angewandt werden sollen. Eine Einschränkung kann sowohl über den Typ der UML-Objekte (s. Tabelle 23) als auch über zugeordnete Stereotypes (s. Tabelle 24) erfolgen.

Alles	Alle UML-Objekte
Pakete	Nur UML-Packages
Elemente	Nur UML-Elements
Attribute	Nur UML-Attribute
Assoziations-Enden	Nur UML-Assoziations-Enden

Tabelle 23: Einschränkung der Tagged Value Bearbeitung auf bestimmte UML-Objekte

Alle Stereotypes	Keine Einschränkung
ApplicationSchema	Nur UML-Packages mit Stereotype << ApplicationSchema>>
Leaf	Nur UML-Packages mit Stereotype << Leaf>>
FeatureType	Nur UML-Elemente mit Stereotype <<FeatureType>>
DataType	Nur UML-Elemente mit Stereotype <<DataType>>
Enumeration	Nur UML-Elemente mit Stereotype <<EnumerationType>>
CodeList	Nur UML-Elemente mit Stereotype <<CodeList>>
Union	Nur UML-Elemente mit Stereotype <<Union>>
Type	Nur UML-Elemente mit Stereotype <<Type>>
Kein Stereotype	Nur UML-Elemente ohne Stereotype

Tabelle 24: Einschränkung der UML-Bearbeitung auf bestimmte Stereotypes

Der Button „Ausführen startet die Bearbeitung der Tagged Values. Wie alle anderen Funktionen der UML-Transformation Software erfolgt die Bearbeitung auf dem Temporären Modell. Wenn die CheckBox „Modell überschreiben“ aktiviert ist, wird nach Abschluss der Tagged Value Bearbeitung das temporäre Modell in das Ausgangsmodell kopiert. Der Button „Beenden“ schließt den Dialog, ohne weitere Aktionen auszulösen.

8 Vergleich von UML-Modellen

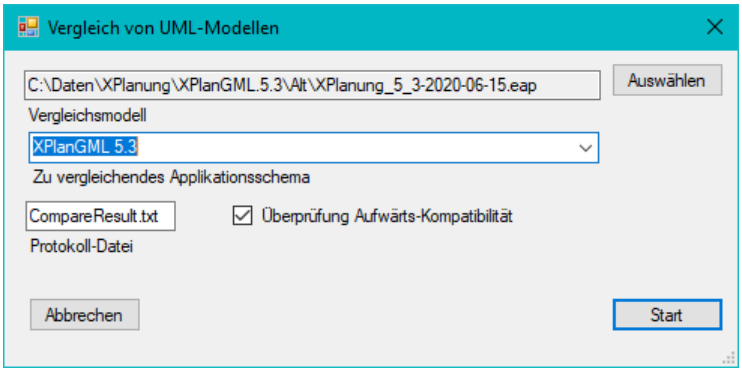


Abbildung 6: Vergleich von UML-Applikationsschemata

Der in Abbildung 6 gezeigte Dialog ermöglicht, zwei Appikationsschemata in unterschiedlichen EnterpriseArchitect UML-Modellen zu vergleichen und die Unterschiede zu protokollieren. Die Bedeutung der Bedienelemente zeigt Tabelle 25.

Auswählen	Der Button ermöglicht die Auswahl eines EnterpriseArchitect UML-Modelles („Vergleichsmodell“), das mit dem über die zentrale Benutzeroberfläche (Abbildung 1) festgelegten Modell („Basismodell“) verglichen wird. Der Pfadname des Vergleichsmodells wird im entsprechenden Feld angezeigt.
Zu vergleichendes Applikationsschema	Über die ComboBox muss ein Applikationsschema des Vergleichsmodells ausgewählt werden, das mit den Applikationsschema des Basismodells verglichen wird.
Protokoll-Datei	Name der Datei mit den Ergebnissen des Modellvergleichs. Die Datei befindet sich im Ordner des Basismodells
Überprüfung Aufwärts-Kompatibilität	Überprüft, ob des Schema des Vergleichsmodells aufwärts-kompatibel zum Schema des Basismodells ist. Alle Änderungen, die die Aufwärts-Kompatibilität verletzen, werden im Protokoll gesondert ausgezeichnet.
Start	Startet den Modellvergleich
Abbrechen	Beendet den Dialog, ohne einen Modellvergleich durchzuführen.

Tabelle 25: Parameter des Modellvergleich-Dialogs

9 Weitere Funktionen

9.1UML-Diagramme exportieren

Erzeugt im Ordner des ausgewählten UML-Modells einen Unterordner „diagrams“, in den alle im UML-Diagramme des gewählten Applikationsschemas (im *.emf Format) geschrieben werden.

9.2Info

Öffnet einen Dialog mit Informationen über die GML-Toolbox. Der Dialog bietet insbesondere auch die Möglichkeit, dies Benutzerhandbuch zu öffnen.