Praktische Aspekte der INTERLIS-Verarbeitung

Jens Ingensand

KKGEO/IGArc Workshop
BAFU, 07.05.2014

Praktische Aspekte der INTERLIS-Verarbeitung

Ausgangslage:

- Konzeptionelle INTERLIS (2)-Modelle müssen in konkrete Datenbankmodelle umgesetzt werden

  => Die 1:1 Umwandlung ist den meisten Fällen unmöglich

- Schnittstellen für den Import- und Export müssen erstellt werden

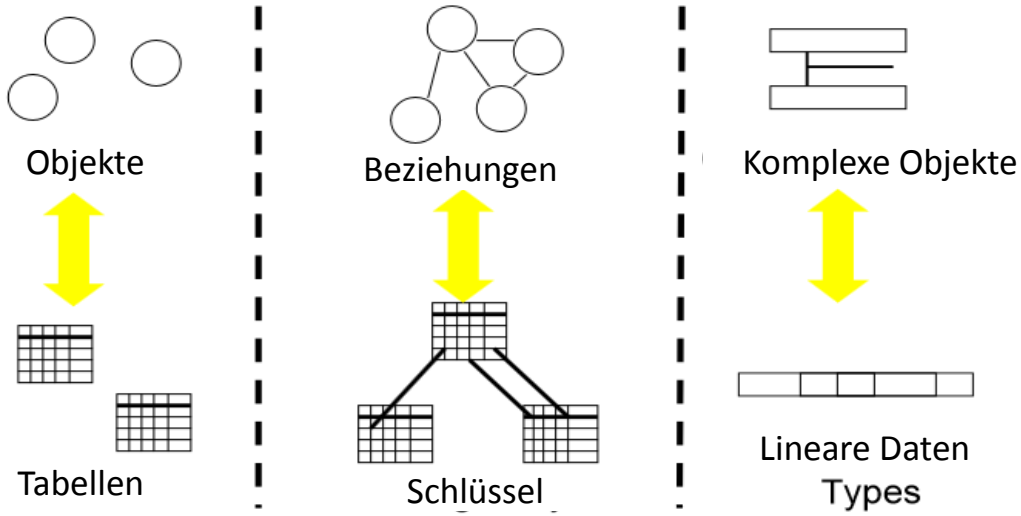
Praktische Aspekte der INTERLIS-Verarbeitung

Programm:

- Unterschiede INTERLIS 2 und konventionale Datenbanken
- Umwandlungsmöglichkeiten
- Konkrete Beispiele
- Schnittstellen erstellen mit FME
- Zusammenfassung

Unterschiede INTERLIS 2 und konventionale Datenbanken

- Interlis 2 ist objektorientiert



Objektorientiert
(Interlis 2)

Relational (z.B. GDB)

=> Modelle müssen entschachtelt werden, Schlüssel müssen erstellt werden

Unterschiede INTERLIS 2 und konventionale Datenbanken

- Interlis 2 ermöglicht den Import von anderen Modellen

```
IMPORTS UNQUALIFIED CHAdminCodes_V1;
IMPORTS UNQUALIFIED GeometryCHLV03_V1;
IMPORTS UNQUALIFIED CatalogueObjects_V1;
IMPORTS UNQUALIFIED LocalisationCH_V1;
IMPORTS UNQUALIFIED WasserBase_V1;
```

=> Die wahre Komplexität eines Modells kann um ein Vielfaches grösser sein, als auf den ersten Blick ersichtlich

Unterschiede INTERLIS 2 und konventionale Datenbanken

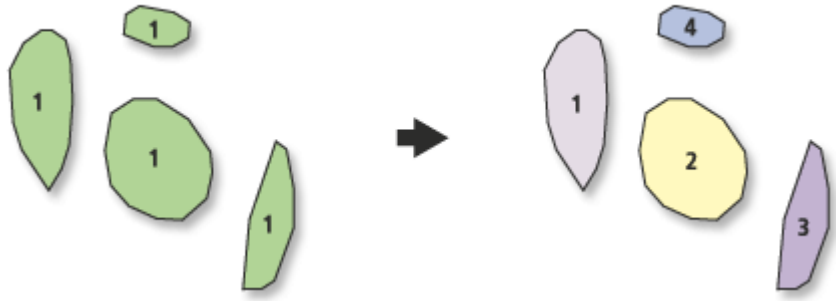
- Interlis Modelle sind in Topics und Klassen organisiert.

```
TOPIC RevitOeko =
!!============================================================================
  !! Oekomorphologie
  !!=============================================
  !! Abschnitt Oekomorphologie
  !!=============================================
  CLASS AbschOekomorph =
    Abschnitt                 : MANDATORY WasserBase_V1.Gewaesser.StrOrt
    Sohlenbreite              : MANDATORY 0.00 .. 1000.00 [INTERLIS.m];
    Eindolung                 :           BOOLEAN;
    Breitenvariabilitaet      : MANDATORY RevitalisierungGewaesser_V1.Ca
    Tiefenvariabilitaet       :           RevitalisierungGewaesser_V1.Ca
    Sohlenverbauung           : MANDATORY RevitalisierungGewaesser_V1.Ca
```

=> Viele Datenbanken kennen nur Schemas und Tabellen

Unterschiede INTERLIS 2 und konventionale Datenbanken

- Interlis 2 Modelle unterstützten Multipartobjekte nur über CH-Base Erweiterungen



=> Gegebenenfalls müssen Multipartobjekte in Singlepartobjekte umgewandelt werden

Unterschiede INTERLIS 2 und konventionale Datenbanken

- Interlis unterstützt mehrere Geometrien pro Klasse

```
CLASS StrOrtPlanar =
  Punkt           : Coord3;
  Abschnitt       : DirectedLine;
  Flaeche         : Surface;

  MANDATORY CONSTRAINT
        DEFINED(Punkt)
     OR DEFINED(Abschnitt)
     OR DEFINED(Flaeche);
END StrOrtPlanar;
```

=> Viele Datenbankensysteme unterstützten nur eine Geometrie pro Tabelle

Unterschiede INTERLIS 2 und konventionale Datenbanken

- Interlis 2 unterstützt mehrere Bedingungen innerhalb einer Klasse

```
CLASS Intensitaet_ProProzessquelle EXTENDS Gefahrenkartierung_V1.Intensitaeten.Intensitaet =
  ProzQuelle: MANDATORY TEXT;
  Teilprozess: MANDATORY Gefahrenkartierung_V1.Gefahrenkartierung_Basis_WithOneState.Prozess_detailli
  Teilszenario_Wahrscheinlichkeit: MANDATORY Gefahrenkartierung_V1.Gefahrenkartierung_Basis_WithOneSt
  Szenario_Beschreibung: MTEXT;
  MANDATORY CONSTRAINT
    (Teilszenario_Wahrscheinlichkeit == 1.00) OR
    DEFINED(Szenario_Beschreibung);
  MANDATORY CONSTRAINT
    (Teilprozess != #Rutschung.permanente_Rutschung) OR
    NOT (DEFINED (Jaehrlichkeit));
```

Unterschiede INTERLIS 2 und konventionale Datenbanken

- Interlis 2 Datentypen und reservierte Bezeichnungen unterscheiden sich von konventionalen Datenbanken

Beispiele:

- Wertebereiche: `TEXT*30`
- Keine Sonderzeichen éàöü etc. in Attributnamen erlaubt
- Keine Zwischenräume in Attributnamen erlaubt
- Attributname muss mit Buchstaben beginnen
- Es gibt eine Reihe von reservierten Worten (100+): z.B. BOOLEAN, END, IN, NAME, MODEL, NULL, OID, PARAMETER, THIS, SURFACE, etc.

Umwandlungsmöglichkeiten

Ziel: Erstellung einer Datenbankstruktur

- um modellkonforme Daten in Empfang zu nehmen
- um modellkonforme Daten zu liefern

=> Es ist im Prinzip unmöglich eine Datenbank zu erstellen,
welche 100% modellkonform ist

Umwandlungsmöglichkeiten

Verschiedene Ansätze:

- Vollautomatisch (=> Tools wie iliX)
- Halbautomatisch (=> Tools wie iliX + manuelle Anpassung)
- Manuell

=> Eine fachliche Validierung macht in jedem Fall Sinn

Umwandlungsmöglichkeiten

Wichtige Punkte und Fragen

- Welche Topics / Klassen / Attribute müssen überhaupt berücksichtigt werden?
- Flache Strukturen machen durchaus Sinn
- Bedingungen müssen nicht unbedingt in der Datenbank modelliert werden (=> Import oder Exportprozesse)
- Speziell für den Export / Import erstellte Datenbanken können Sinn machen (=> Zwischenspeicherung der Daten)

Konkrete Beispiele

Übersetzung INTERLIS 2 -> GDB: Einfaches Beispiel

Interlis 2

```
CLASS Hochmoor =
  ObjNummer : MANDATORY TEXT*40;
  ObjName : MANDATORY TEXT*30;
  Mutationsdatum : MANDATORY INTERLIS.XMLDate;
  Mutationsgrund_Text : MANDATORY LocalisationCH_V1.MultilingualMText;
END Hochmoor;
```

Konkrete Beispiele

Übersetzung INTERLIS 2 -> GDB: Einfaches Beispiel

Interlis 2

```
CLASS Hochmoor =
  ObjNummer : MANDATORY TEXT*40;
  ObjName : MANDATORY TEXT*30;
  Mutationsdatum : MANDATORY INTERLIS.XMLDate;
  Mutationsgrund_Text : MANDATORY LocalisationCH_V1.MultilingualMText;
END Hochmoor;
```

```
STRUCTURE MultilingualMText =
  LocalisedText : BAG {1..*} OF LocalisedMText;
  UNIQUE (LOCAL) LocalisedText:Language;
END MultilingualMText;
```

- Mutationsgrund_Text: Ein Objekt vom Typ MultilingualMText

Konkrete Beispiele

Übersetzung INTERLIS 2 -> GDB: Einfaches Beispiel

Interlis 2

```
CLASS Hochmoor =
  ObjNummer : MANDATORY TEXT*40;
  ObjName : MANDATORY TEXT*30;
  Mutationsdatum : MANDATORY INTERLIS.XMLDate;
  Mutationsgrund_Text : MANDATORY LocalisationCH_V1.MultilingualMText;
END Hochmoor;
```

```
STRUCTURE MultilingualMText =
  LocalisedText : BAG {1..*} OF LocalisedMText;
  UNIQUE (LOCAL) LocalisedText:Language;
END MultilingualMText;
```

```
STRUCTURE LocalisedMText =
  Language: LanguageCode_ISO639_1;
  Text: MANDATORY MTEXT;
END LocalisedMText;
```

- Mutationsgrund_Text: Ein Objekt vom Typ MultilingualMText
- LocalisedText: Ein Bag-Objekt vom Typ LocalisedMText

Konkrete Beispiele

Übersetzung INTERLIS 2 -> GDB: Einfaches Beispiel

Interlis 2

```
CLASS Hochmoor =
  ObjNummer : MANDATORY TEXT*40;
  ObjName : MANDATORY TEXT*30;
  Mutationsdatum : MANDATORY INTERLIS.XMLDate;
  Mutationsgrund_Text : MANDATORY LocalisationCH_V1.MultilingualMText;
END Hochmoor;
```

```
STRUCTURE MultilingualMText =
  LocalisedText : BAG {1..*} OF LocalisedMText;
  UNIQUE (LOCAL) LocalisedText:Language;
END MultilingualMText;
```

```
STRUCTURE LocalisedMText =
  Language: LanguageCode_ISO639_1;
  Text: MANDATORY MTEXT;
END LocalisedMText;
```

Lösung 1: Flache Tabelle

Field Name	Data Type
OBJECTID	Object ID
ObjNummer	Text
ObjName	Text
Mutationsdatum	Date
Mutationsgrund_text	Text
Mutationsgrund_language	Text

Konkrete Beispiele

Übersetzung INTERLIS 2 -> GDB: Einfaches Beispiel

Interlis 2

```
CLASS Hochmoor =
  ObjNummer : MANDATORY TEXT*40;
  ObjName : MANDATORY TEXT*30;
  Mutationsdatum : MANDATORY INTERLIS.XMLDate;
  Mutationsgrund_Text : MANDATORY LocalisationCH_V1.MultilingualMText;
END Hochmoor;
```

```
STRUCTURE MultilingualMText =
  LocalisedText : BAG {1..*} OF LocalisedMText;
  UNIQUE (LOCAL) LocalisedText:Language;
END MultilingualMText;
```

```
STRUCTURE LocalisedMText =
  Language: LanguageCode_ISO639_1;
  Text: MANDATORY MTEXT;
END LocalisedMText;
```

Lösung 1: Flache Tabelle

Field Name	Data Type
OBJECTID	Object ID
ObjNummer	Text
ObjName	Text
Mutationsdatum	Date
Mutationsgrund_text	Text
Mutationsgrund_language	Text

- Von ESRI generiertes Attribut
- Mutationsgrund_language: notwendig?
- Achtung: Attributnamen (Export => shapefile)

Konkrete Beispiele

Übersetzung INTERLIS 2 -> GDB: Einfaches Beispiel

Interlis 2

```
CLASS Hochmoor =
  ObjNummer : MANDATORY TEXT*40;
  ObjName : MANDATORY TEXT*30;
  Mutationsdatum : MANDATORY INTERLIS.XMLDate;
  Mutationsgrund_Text : MANDATORY LocalisationCH_V1.MultilingualMText;
END Hochmoor;
```

```
STRUCTURE MultilingualMText =
  LocalisedText : BAG {1..*} OF LocalisedMText;
  UNIQUE (LOCAL) LocalisedText:Language;
END MultilingualMText;
```

```
STRUCTURE LocalisedMText =
  Language: LanguageCode_ISO639_1;
  Text: MANDATORY MTEXT;
END LocalisedMText;
```

Lösung 2: Zwei Tabellen + Verknüpfung

OBJECTID	Object ID
HochmoorID	Long Integer
ObjNummer	Text
ObjName	Text
Mutationsdatum	Date

OBJECTID	Object ID
HochmoorID	Long Integer
Language	Text
Text	Text

- Haupttabelle Hochmoor
- Tabelle Mutationsgrund_Text
- HochmoorID: Verknüpfungsattribut: muss erstellt werden (es gibt keine Garantie, dass ObjNummer als Schlüssel benutzt werden kann)

Konkrete Beispiele

Datentypen: Grundtypen

	Interlis	ESRI	
Zahlen	Wertebereiche: Z.B. 0 .. 130	Short integer	-32,768 bis 32,767
		Long integer	-2,147,483,648 bis 2,147,483,647
	Wertebereiche: z.B. 0.0 .. 130.0	Single-precision floating-point number (float)	Ca. -3.4E38 bis 1.2E38
		Double-precision floating-point number (double)	Ca. -2.2E308 bis 1.8E308
Text	Mit Längedefinition: TEXT*30	Text(Länge)	

=> Text ohne Längendefinition (INTERLIS) kann problematisch sein

Konkrete Beispiele

Domains für Aufzählungen und Wertebereiche

Interlis2

```
DOMAIN D = (
    D1,
    D2,
    D3)
CLASS A =
    Attribut_1: D;
END A;
```

GDB

Domains können benutzt werden;
=> sowohl für Wertebereiche, als auch für Aufzählungen

Konkrete Beispiele

Bedingungen

Interlis2

Topologiebedingungen

```
SURFACE WITHOUT OVERLAPS;
```

Genauigkeit

INTERLIS gibt die Genauigkeit der Koordinaten durch Nachkommastellen (z. B. hier 1 mm) an

```
COORD
480000.000 .. 840000.000 [m],
7000.000 .. 300000.000 [m],
200.000 .. 5000.000 [m];
```

GDB

nur Überprüfung möglich über Topologiebedingungen

Resolution : 1mm
Tolerance (ESRI Empfehlung):
mindestens den doppelten Wert
der Resolution: 2mm
nur Überprüfung der Bounding-box
möglich

Schnittstellen erstellen mit FME

FME (Feature Manipulation Engine)
- Das Armeemesser für Geodaten
- Liest und schreibt die meisten Datenformate (275+)

Plugin ili2fme
- Interlis 1+2 Daten lesen und schreiben mit FME
- Seit 2011 in der FME Standardinstallation dabei

http://www.safe.com
http://www.eisenhutinformatik.ch/interlis/ili2fme/

Schnittstellen erstellen mit FME

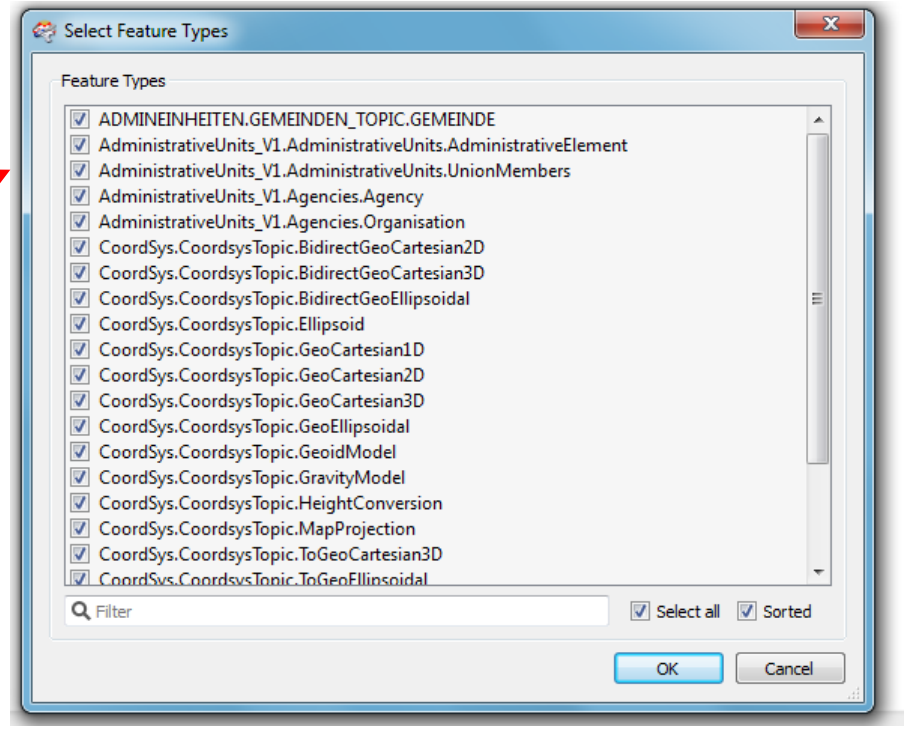
FME + ili2fme: Möglichkeiten

- Überprüfung von Bedingungen (Topologien, etc); z.B. um die Modellkonformität von Daten sicherzustellen (und zu korrigieren)
- Attributmapping
- Deaggregieren von Multipartobjekten / Aggregieren von Singlepartobjekten
- Erstellen der Verschachtelungsstruktur
- Einlesen von INTERLIS-Modellen, um modellkonforme Writer-Featuretypes zu erstellen

Schnittstellen erstellen mit FME – INTERLIS lesen

Das Ziel: eine XTF-Datei einlesen

- FME kann mit ili2fme XTF/XML Dateien lesen und für jede Klasse einen Reader Feature Type erstellen
- Sämtliche in der INTERLIS-Modell Datei enthaltene Modelle werden berücksichtigt
  ⇒ Der Benutzer wählt z.B. das Hauptmodell in der Liste aus



Schnittstellen erstellen mit FME – INTERLIS lesen

Das Ziel: eine XTF-Datei einlesen

Wichtige ili2fme Parameter:

Inheritance Mapping Strategy:
- Superclass: Attribute von den Klassen, welche Attribute von einer Oberklasse (Superclass) erben, werden der Oberklasse zur Verfügung gestellt
- Subclass: Attribute von der Oberklasse (Superclass), werden den Klassen zur Verfügung gestellt, welche Attribute von der Oberklasse erben.
  (default: Superclass)

Topics Filter:
- Topics, welche gelesen werden sollen
  (default: leer)

Check TID/OID Uniqueness:
- Der Reader prüft, ob TID und OID eindeutig sind
  (default: leer); bei voluminösen Daten kann dieser Parameter zu langen Bearbeitungszeiten führen

Geometry Encoding
- Definiert, wie Attribute mit Geometrien behandelt werden; dies ist wichtig, wenn ili2fme die Geometrie nicht sofort erkennt (z.B wenn die

Schnittstellen erstellen mit FME – INTERLIS schreiben

Das Ziel: eine XTF-Datei, die dem Modell entspricht

```
INTERLIS 2.2;

MODEL AHV_Kasse (fr) =

  TOPIC Versicherte =

    CLASS Person =
     AHV_Nummer : MANDATORY TEXT*20;
     Nachname : TEXT*20;
     Vorname : TEXT*20;
    END Person;

  END Versicherte;

END AHV_Kasse.
```

.ili Modell

Schnittstellen erstellen mit FME – INTERLIS schreiben

Das Ziel: eine XTF-Datei, die dem Modell entspricht

```
INTERLIS 2.2;

MODEL AHV_Kasse (fr) =

  TOPIC Versicherte =

    CLASS Person =
     AHV_Nummer : MANDATORY TEXT*20;
     Nachname : TEXT*20;
     Vorname : TEXT*20;
    END Person;

  END Versicherte;

END AHV_Kasse.
```

.ili Modell

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- File AHV_Kasse.xml 2005-01-15 -->
<TRANSFER xmlns="http://www.interlis.ch/INTERLIS2.2">
        <HEADERSECTION VERSION="2.2" SENDER="EPFL">
                <ALIAS>
                        <ENTRIES FOR="AHV_Kasse">
                                <TAGENTRY FROM="AHV_Kasse.Versicherte" TO="AHV_Kasse.Versicherte"/>
                                <TAGENTRY FROM="AHV_Kasse.Versicherte.Person" TO="AHV_Kasse.Versicherte.Person"/>
                        </ENTRIES>
                </ALIAS>
        </HEADERSECTION>
        <DATASECTION>
                <AHV_Kasse.Versicherte TOPICS="AHV_Kasse.Versicherte" BID="xREFHAND0000000">
                        <AHV_Kasse.Versicherte.Person TID="x1">
                                <AHV_Nummer>6568.455.84</AHV_Nummer>
                                <Nachname>PetitNachname</Nachname>
                                <Vorname>Jean-Paul</Vorname>
                        </AHV_Kasse.Versicherte.Person>
                        <AHV_Kasse.Versicherte.Person TID="x2">
                                <AHV_Nummer>9856.534.54</AHV_Nummer>
                                <Nachname>GrandNachname</Nachname>
                                <Vorname>Robert</Vorname>
                        </AHV_Kasse.Versicherte.Person>
                </AHV_Kasse.Versicherte>
        </DATASECTION>
</TRANSFER>
```

.xtf Datei mit den Daten

Schnittstellen erstellen mit FME – INTERLIS schreiben

Das Ziel: eine XTF-Datei, die dem Modell entspricht

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- File AHV_Kasse.xml 2005-01-15 -->
<TRANSFER xmlns="http://www.interlis.ch/INTERLIS2.2">
        <HEADERSECTION VERSION="2.2" SENDER="EPFL">
                <ALIAS>
                        <ENTRIES FOR="AHV_Kasse">
                                <TAGENTRY FROM="AHV_Kasse.Versicherte" TO="AHV_Kasse.Versicherte"/>
                                <TAGENTRY FROM="AHV_Kasse.Versicherte.Person" TO="AHV_Kasse.Versicherte.Person"/>
                        </ENTRIES>
                </ALIAS>
        </HEADERSECTION>
        <DATASECTION>
                <AHV_Kasse.Versicherte TOPICS="AHV_Kasse.Versicherte" BID="xREFHAND0000000">
                        <AHV_Kasse.Versicherte.Person TID="x1">
                                <AHV_Nummer>6568.455.84</AHV_Nummer>
                                <Nachname>PetitNachname</Nachname>
                                <Vorname>Jean-Paul</Vorname>
                        </AHV_Kasse.Versicherte.Person>
                        <AHV_Kasse.Versicherte.Person TID="x2">
                                <AHV_Nummer>9856.534.54</AHV_Nummer>
                                <Nachname>GrandNachname</Nachname>
                                <Vorname>Robert</Vorname>
                        </AHV_Kasse.Versicherte.Person>
                </AHV_Kasse.Versicherte>
        </DATASECTION>
</TRANSFER>
```

Wichtig: DATASECTION

BID = Basket-ID (eindeutige ID pro TOPIC)

TID = Transfer-ID(eindeutige ID pro Objekt)

=> BID und TID müssen erzeugt werden

Schnittstellen erstellen mit FME – INTERLIS schreiben

Problem

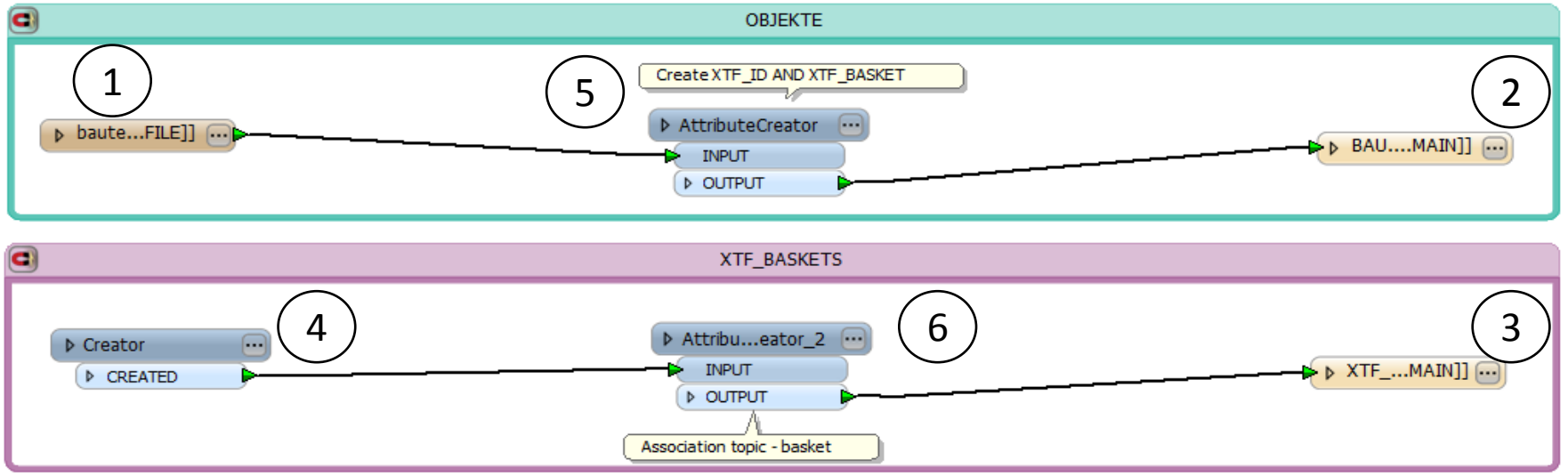
- FME muss eine verschachtelte XML – Struktur (Topics, Objekte, etc) erzeugen

Lösung: XTF_BASKETS Feature Type

- Beschreibung der Verschachtelung
- Beinhaltet die Attribute xtf_topic und xtf_id
  - Xtf_id ist das XML Attribut TID für Objekte oder das XML Attribut BID für Topics

Schnittstellen erstellen mit FME – INTERLIS schreiben

Minimaler Workspace



1: Input: z.B. GDB
2. Output Interlis (Features)
3. Output Interlis (XTF Baskets)
4. Creator: neues Objekt (pro Topic eins) erstellen

5. Attribute Creator 1:
   Attribute Name: xtf_id
   Attribute Value: (eindeutige ID)
   Attribute Name: xtf_basket
   Attribute Value: (eindeutige ID für das Topic) = TID
6. Attribute Creator 2:
   Attribute Name: xtf_id
   Attribute Value: (eindeutige ID für das Topic) = BID
   Attribute Name: xtf_topic
   Attribute Value: (MODEL.TOPIC)

Schnittstellen erstellen mit FME – INTERLIS schreiben

Workspace erstellen

1. Reader wie gewohnt erstellen (z.B. GDB)
2. Writer erstellen:
    1. Writer => Add Writer
    2. Format: Swiss INTERLIS (ili2fme)
    3. Dataset: Outputordner und outputfile (z.B. output.xtf) auswählen
    4. !Wichtig!: das Dialogfeld erscheint:

auf «No» klicken

Schnittstellen erstellen mit FME – INTERLIS schreiben

5. Writer => Import Feature Types

Format: Swiss INTERLIS (ili2fme)
Dataset: die Interlis – Modell-Datei auswählen (Achtung: Fileextension auf .ili ändern)
=> Eine Liste mit Feature Types erscheint
=> Mindestens die entsprechende Klasse und XTF_BASKETS auswählen

Der Workspace sollte nun mindestens zwei Feature Types enthalten: eine Klasse und XTF_Baskets – beide müssen abgefüllt werden

Schnittstellen erstellen mit FME – INTERLIS schreiben

Benutzung von List-Objekten in FME um Daten ineinander zu verschachteln

```
TOPIC TEST =

    STRUCTURE TestStruct =
    a: TEXT*30;
    b: TEXT*30;
    c: TEXT*30;
    END TestStruct;

    CLASS TestClass =
    StructAttr: TestStruct;
    END TestClass;
```

Ein verschachteltes Attribut: StructAttr

Schnittstellen erstellen mit FME – INTERLIS schreiben

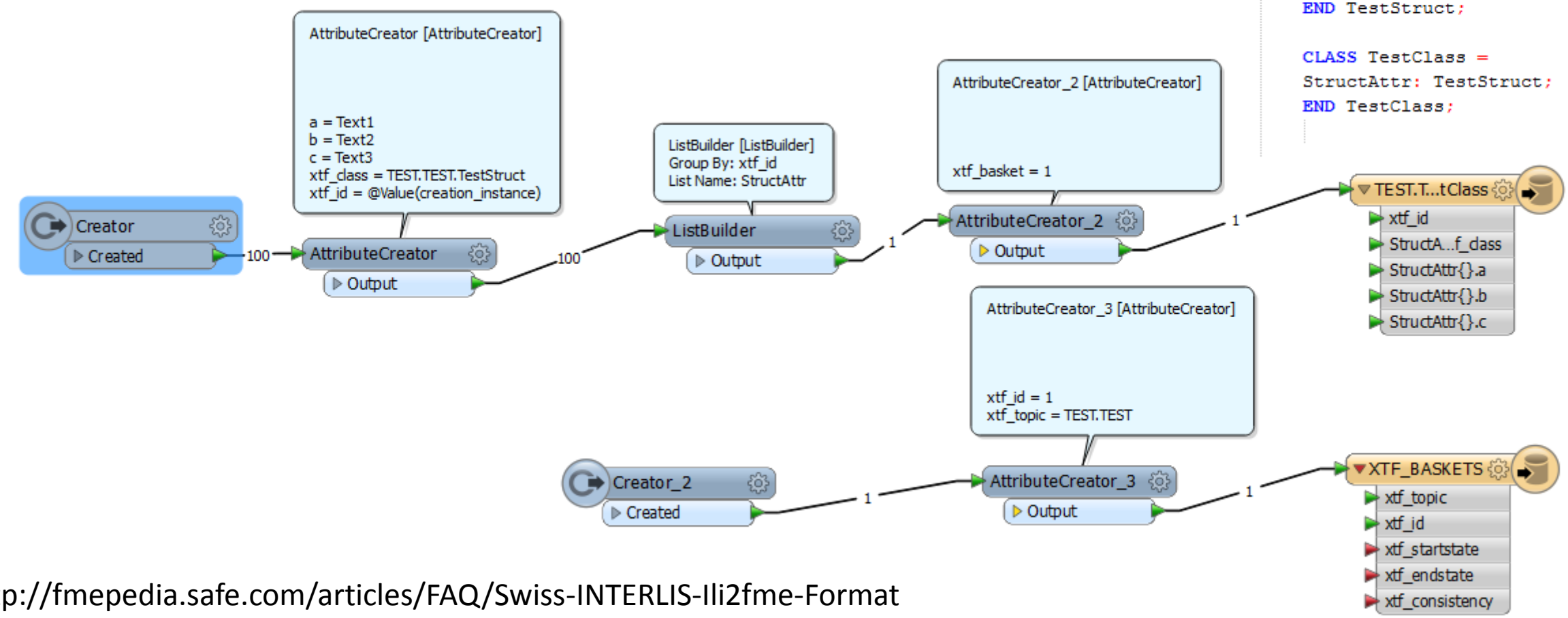
Benutzung von List-Objekten in FME um Daten ineinander zu verschachteln

Beispiel:

```
TOPIC TEST =

    STRUCTURE TestStruct =
    a: TEXT*30;
    b: TEXT*30;
    c: TEXT*30;
    END TestStruct;

    CLASS TestClass =
    StructAttr: TestStruct;
    END TestClass;
```



http://fmepedia.safe.com/articles/FAQ/Swiss-INTERLIS-Ili2fme-Format

Best practice

Interlis lesen

- Standard-Interlisdaten lassen sich einfach mit dem FME Plugin in ArcGIS lesen
- CH-Base- basierte Interlisdaten müssen mit FME Workbench entschachtelt werden
- CH-Base- basierte Geometrien müssen mit dem GeometryReplacer Transformer definiert werden
- Entschachteln von INTERLIS-Daten: ListExploder-Transformer

Interlis schreiben

- Multipart-Geometrien müssen de-aggregiert werden (Deaggregator-Transformer)
- Domains müssen entweder manuell re-codiert werden oder im Reader re-codiert werden (plus ein AttributeExposer, um die Attribute zugänglich zu machen)

Zusammenfassung

- INTERLIS (2) und konventionale Datenbankensysteme weisen einige Unterschiede auf
- Eine 1:1 Übersetzung von INTERLIS 2 Modellen ist im Prinzip unmöglich
- Flache(re) Strukturen genügen in vielen Fällen
- Eine fachliche Begutachtung einer erstellten Datenbank ist sinnvoll
- Datenbanken für die Zwischenspeicherung können die Arbeit erleichtern
- Die Modellkonformität kann über Import / Exportprozesse sichergestellt werden; z.B. mit FME

Fragen?

jens.ingensand@heig-vd.ch