KANTON solothurn

Modellbasierte Datenerfassung - Handbuch

Amt für Geoinformation Kanton Solothurn <agi@bd.so.ch>
Version 2.0.22, 2024-07-29 04:30

Inhaltsverzeichnis

1. Einleitung

Das Handbuch «Modellbasierte Datenerfassung» ist ein Hilfsmittel für das Arbeiten mit INTERLIS-Datenmodellen in der kantonalen Geodateninfrastruktur. Es ist entstanden aus den ursprünglichen Modellierungsregeln, umfasst in der heutigen Form aber mehr. Es behandelt in der jetzigen Form auch vor- und nachgelagerte Themen wie die Modellablage oder Integrations- und Nachführungsprozesse.

Abweichung zu den vorliegenden minimalen Vorgaben sind natürlich erlaubt. Zudem kann und soll das Handbuch nie allumfassend sein. Abweichung müssen jedoch immer bewusst gemacht werden und begründbar sein.

Änderungswünsche, Ergänzungen, Hinweise auf Fehler/Widersprüche etc. sind an Stefan Ziegler zu richten.

2. Modelltypen

INTERLIS-Modelle werden für verschiedene Zwecke eingesetzt. Entsprechend muss dies beim Modellieren berücksichtigt werden. Nicht in jedem Fall werden oder müssen sowohl ein Erfassungs- und eine Publikationsmodell geschrieben werden.

2.1. Erfassungsmodell

Erfassungsmodelle dienen der Erfassung und Nachführung von Daten in der Datenbank. Als Werkzeug zur Bewirtschaftung der Daten wird *QGIS* eingesetzt. Der Fokus bei der Modellierung liegt auf den Verzicht von Redundanzen, d.h. die Modelle sind normalisiert. So werden z.B. zwischen den einzelnen Klassen Beziehungen modelliert und Aufzähltypen modelliert, um allfällige Redundanzen zu vermeiden. Im Zusammenspiel mit *QGIS* entsteht somit eine «generische» Erfassungsfachschale.

2.2. Publikationsmodell

Aufgrund der Normalisierung eignen sich Daten im Erfassungsmodell nicht für die Publikation in einem Web GIS oder als «einfacher» Layer in *QGIS*. Die für den Benutzer wichtigen Informationen sind in einem Erfassungsmodell oftmals auf verschiedene Klassen resp. Datenbanktabellen verteilt. Für die reine Darstellung und das einfache Abfragen von Informationen ist ein simpleres Modell – das Publikationsmodell – zu erarbeiten. Hauptmerkmale eines Publikationsmodelles:

- Verzicht auf Beziehungen (Assoziationen) zwischen Klassen.
- Eine Geometriespalte pro Klasse.
- Der Inhalt kann zu jedem Zeitpunkt und automatisiert aus dem Erfassungsmodell und weiterer vorhandener Daten abgeleitet werden.

Es wird kein zusätzliches Datenabgabemodell angestrebt.

2.2.1. Aufzähltypen

Damit Aufzähltypen sinnvoll im Web GIS Client eingesetzt werden können, müssen zwei Bedingungen erfüllt sein: Jeder Aufzähltyp-Wert muss ein Metaattribut `@ili2db.dispName` (siehe dazu auch die offizielle Dokumentation von ili2pg) erhalten und es braucht in der Datentabelle eine zusätzliche Spalte, die den Wert des Metaattributes (den «schönen» Text) zum Aufzähltyp-Wert speichert.

Für die erste Bedingung muss mindestens UML-Editor grösser V3.9.0 verwendet werden. Diese Version erlaubt es für jedes Objekt beliebige Metaattribute zu erfassen.

Für die zweite Bedingung muss beim Erstellen des Schemas die Option `--createEnumTxtCol` verwendet werden. Diese erstellt für Aufzähltypattribute eine zusätzliche Spalte `<Attributname>_txt`. Werden z.B. Daten in ein solches Schema importiert, werden die `@ili2db.dispName`-Werte der Aufzähltypwerte gespeichert. Im häufigen Fall einer Publikation von Daten aus der Edit-DB in die Pub-DB muss man sich jedoch selber um das Speichern der `@ili2db.dispName`-Werte kümmern. Am einfachsten ist ein Update-Befehl mit einem SqlExecutor-Task nach dem Db2Db-Task. Der Update-Befehl macht ein Join mit der Aufzähltyp-Tabelle (`--createEnumTabs`).

2.3. Seitenwagenmodell

Ein Seitenwagenmodell ist strukturell ein Publikationsmodell, liegt aber in der Erfassungsdatenbank. Es wird für den CCC-Cache verwendet, d.h. die externe Fachanwendung schreibt via Data-Service in die Erfassungsdatenbank.

2.4. Validierungsmodell

Das Validierungsmodell dient zur Definition von zusätzlichen Prüfungen mit der Software *ilivalidator*. In diesem Modell müssen zuerst Views erstellt werden. In diesen Views können mit Constraints die zusätzlichen Prüfungen definiert werden. Ein Beispiel findet sich hier.

Weitergehende Informationen zu den Modelltypen gibt es in der Präsentation des AGI vom 29. August 2017 zum Thema «Arbeiten mit einem Datenmodell».

Zu einem Validierungsmodell gehört auch immer eine Konfigurations- und Metakonfigurationsdatei.

2.5. Extension-Functions-Modell

Zum jetzigen Zeitpunkt (2022-06-22) gibt es *ein* Modell (`SO_FunctionsExt`), in welchem zusätzliche Prüffunktionen deklariert werden. Die Funktionen selber werden in Java implementiert. Der Modellnamen dieses Modelles ist statisch, damit nicht bei jedem neuen Release sämtliche Validierungsmodelle, welche dieses Modell importieren, nachgeführt werden müssen. Dies bedingt jedoch eine strenge Stabilität der Funktionssignaturen.

3. Modellierungsprozess

Auf eine strenge Formalisierung des eigentlichen Modellierungsprozesses («Startsitzung», «FIG» etc.) wird verzichtet. Die Datenmodelle werden durch das AGI gemeinsam mit den Dienststellen erarbeitet. Für den Modellierungsprozess existiert eine Checkliste (aktuelle Version unter `H:\BJSVW\Agi\KGDM\Vorlagen\`). Diese ist zu verwenden und gegebenenfalls zu ergänzen.

3.1. Datenerfassungswunsch in der Dienststelle

Verantwortung: Dienststelle

Auslöser kann eine neue Aufgabe sein oder es können bestehende Daten sein, welche in die modellbasierte Struktur gebracht werden sollen.

3.2. Erarbeitung der Modelle

Verantwortung: AGI und Dienststelle

Zusammen mit der Dienststelle erarbeitet das AGI die benötigten Datenmodelle (in der Regel ein Erfassungs- und ein Publikationsmodell).

Hilfreich für die Dienststelle kann eine Exceldatei (`H:\BJSVW\Agi\KGDM\Vorlagen\Datenmodell_Grundlage_Vorlage.xlsx`) sein, in die sie in tabellarischer Form ihr «Modell» eintragen (Attributename, -typen etc.) kann. Eventuell lohnt es sich bereits mit Shapefiles ein paar Testdaten zu erfassen.

3.3. Review der Modelle

Verantwortung: Dienststelle

Die Dienststellen müssen zwingend das Modell verstehen und reviewen.

3.4. Entwicklung und Testen

Verantwortung: AGI und Dienststelle

Schemen und Tabellen in der Datenbank werden mit Schema-Jobs erstellt. Analog den «normalen» GRETL-Jobs werden sie in einem Github-Repository verwaltet. Das Repository ist im Gegensatz zum GRETL-Jobs-Repository nicht öffentlich.

Die Verwendung der Schema-Jobs ist im README.md des Repository klar und ausführlich beschrieben.

Ein Entwickler resp. eine Entwicklerin von Schema-Jobs darf Schemen auf der Test- und Integrationsumgebung selbständig erstellen und löschen.

3.5. Integration in Produktionsumgebung

Verantwortung: AGI

In der Produktionsumgebung kann in der Regel das Modell auch durch den Entwickler resp. die Entwicklerin integriert werden (ohne es an den Betrieb abzudelegieren). Die QGIS-Projektdatei muss mit den entsprechenden Datenbankparametern angepasst werden.

Eine Qualitätskontrolle der Daten wird durch den Publisher (momentan nur für öffentliche Daten) sichergestellt.

3.6. Modelländerungen

Verantwortung: AGI und Dienststelle

Anforderungen an ein Modell können im Laufe der Zeit ändern. Sogenannte Modelländerungen sind zwar nicht gewünscht, aber sind nicht vermeidbar. Mit den Schema-Jobs wurde auch konsequent eine Versionierung in den Schemen-Namen eingeführt.

4. Modelldokumentation und -Ablage

4.1. Dokumentation

In der Regel genügt eine Dokumentation im Modell selbst (Bemerkungen zu Topics, Klassen und Attributen). Bei grösseren Modellen und/oder z.B. im Rahmen von Erfassungsrichtlinien ist eine Dokumentation des Modelles in einem zusätzlichen Dokument notwendig. Die Vorlage für eine solche Dokumentation findet sich hier: `H:\BJSVW\Agi\KGDM\Vorlagen\Modelldokumentation_Vorlage_v01.docx`.

Die Bemerkungen im Modell zu Topics, Klassen und Attributen werden beim Anlegen der Tabellen in der Datenbank automatisch übernommen und als Kommentar zu Tabellen und Attributen geführt. Kommentare zu einem einzelnen Aufzählwert werden in die Spalte `description` in der jeweiligen Aufzähltyp-Tabelle gemappt (nicht zu Verwechseln mit dem Metaattribut `ili2db.dispName`).

Zusätzlich zu den INTERLIS-Objekten muss das Schema in der Datenbank kommentiert werden und falls immer möglich mit Auskunftspersonen (E-Mail-Adressen) hinterlegt werden, z.B.:

`Dieses Schema wird für die Erfassung der Hoheitsgrenzen verwendet. Fragen: stefan.ziegler@bd.so.ch.`

4.2. Ablage

Die definitiven und abgenommenen Datenmodelle (`*.ili`) werden in der INTERLIS-Modellablage des Kantons Solothurn publiziert. Ist das Datenmodell publiziert, darf es ohne Änderung des Modellnamens inhaltlich nicht verändert werden (redaktionelle Änderungen sind denkbar). Die Publikation ist automatisiert und hier beschrieben.

Durch die Publikation in einer Modellablage und der Verknüpfung der verschiedenen Modellablagen untereinander, finden die eingesetzten Werkzeuge (meistens) ohne weiteres Zutun sämtliche INTERLIS-Modelle, die im eigenen Modell importiert werden.

Die UML-Datei wird ebenfalls in der Modellablage abgelegt. Die UML-Datei ist der Master. Es dürfen keine Änderungen nur in der INTERLIS-Modelldatei vorgenommen werden.

Ilivalidator-Konfigurationsdateien (Validierungsmodell, Config-Ini- und Meta-Config-Ini-Datei) werden auch in der Modellablage im jeweiligen Amtsordner abgelegt. Dadurch reicht eine simple Angabe mit "ilidata:xxx.ini", um die Validierung resp. den Validator überall zu steuern.

Die QGIS-Projekte werden im QGIS-Projekte-Repository[https://github.com/sogis/qgis-projects] gespeichert. Das Repository ist nicht öffentlich. Zukünftig soll vermehrt der UsablLIty-Hub verwendet werden.

4.3. Weitere Unterlagen

Weitere Unterlagen im Rahmen der Modellierung sind im Projekte-Ordner abzulegen.

5. Werkzeuge

Für die verschiedenen Modellierungsprozesse stehen passende Werkzeuge zur Verfügung.

5.1. UML/INTERLIS-Editor

Der UML/INTERLIS-Editor dient der Modellierung von INTERLIS-Modellen mittels UML-Diagrammen. Er benötigt Java und kann lokal installiert und verwendet werden. Die Lernkurve ist relativ steil und die Bedienung gewöhnungsbedürftig. Mit dem Verständnis von INTERLIS als Modellierungssprache wächst auch die Verständnis für die Bedienung des *UML/INTERLIS-Editors*.

5.2. ili2c

Mittels INTERLIS-Compiler kann das INTERLIS-Modell syntaktisch geprüft werden. Die Software ist bereits im *UML/INTERLIS-Editor* eingebaut und muss nicht zwingend separat installiert werden.

Es sollte wenn immer möglich die gleiche Version, wie die in GRETL integrierte Version, verwendet werden.

5.3. ili2pg

Ili2pg erstellt aufgrund eines INTERLIS-Modelles ein Schema mit leeren Tabellen in einer PostgreSQL/PostGIS-Datenbank. Es kann auch INTERLIS-Transferdateien importieren oder aus einer Datenbank Daten in eine Transferdatei exportieren.

Es sollte wenn immer möglich die gleiche Version, wie die in GRETL integrierte Version, verwendet werden.

Mit `ili2gpkg` und `ili2fgdb` stehen Derivate für GeoPackage und FileGeodatabase zur Verfügung.

5.4. ilivalidator

Dank des ilivalidators kann eine INTERLIS-Transferdatei auf ihre Modellkonformität geprüft werden. Die Software ist ebenfalls in *ili2pg* eingebaut und ist standardmässig aktiviert und muss mit `--disableValidation` ausgeschaltet werden.

Es sollte wenn immer möglich die gleiche Version, wie die in GRETL integrierte Version, verwendet werden.

5.5. QGIS Model Baker

Das QGIS-Plugin QGIS Model Baker erstellt anhand eines INTERLIS-Modelles resp. der in der Datenbank angelegten Tabellen automatisch ein QGIS-Projekt mit bereits vorkonfigurierten und verknüpften Formularen.

6. Modellierungsregeln

6.1. Modellierungssprache / Compiler

#101

Modellierungssprache ist INTERLIS 2.3 gemäss Referenzhandbuch vom 13. April 2006.

#102

Für die Kontrolle der Datenmodelle wird die aktuelle INTERLIS-Compiler Version eingesetzt.

#103

Der Namensraum (`AT` in der ili-Datei) für Modelle (`Issuer URI` in *UML/INTERLIS-Editor*) ist https://<AMT>.so.ch, wobei das entsprechende Amt angegeben wird.

6.2. Modellname / Version / Formatierung

#201

Die Benennung von Modell- und UML- Dateien erfolgt nach folgendem Schema:

kk_ds_n*_[Publikation]_v*[_Validierung_v*].ili/.uml

kk: Kantonskuürzel (SO)

ds: Kürzel Amt / Dienststelle (AV)

n*: Sprechender Name des Modells (`Nachfuehrungskreise`)

v*: Version des Modells im Format JJJJMMTT (`20160407`)

Für den Dateinamen des Publikationsmodelles wird «Publikation» hinzugefügt. Für den Dateinamen des Validierungsmodells wird dem Dateinamen des zu validierenden Modelles «Validierung» (inkl. Datum) hinzugefügt.

#202

Die Benennung von Modellen erfolgt nach folgendem Schema:

kk_ds_n*_[Publikation]_v*[_Validierung_v*]

kk: Kantonskürzel (SO)

ds: Kürzel Amt / Dienststelle (AV)

n*: Sprechender Name des Modells (`Nachfuehrungskreise`)

v*: Version des Modells im Format JJJJMMTT (`20160407`)

Der Modellnamen des Validierungsmodelles und des Publikationsmodells wird analog der Regel für Dateinamen gewählt.

#203

Die Benennung von Konfigurationsdateien (*.ini) erfolgt nach folgendem Schema:

<Datenmodellname>[_<Kontext>]_v*[-meta].ini

Datenmodellname: Name des Stamm-/Kernmodelles.

Kontext: Kontext für das die Konfigurationsdatei gilt. Z.B. IPW, Drainagen etc. Kann auch leer

sein.

#204

Die Metaattribute `furtherInformation` (in der Regel der Link zur UML-Datei), `technicalContact` (`mailto:agi@bd.so.ch`), `title` und `shortDescription` sind im Header der Modelldatei (als Metaattribute zum Modell) zu erfassen.

#205

Die Änderungshistorie wird im Header der Modelldatei (via UML-Editor) dokumentiert.

#206

Für die Formatierung der Modelldateien dürfen keine Tabulatoren verwendet werden.

#207

In Kommentaren sollen Umlaute verwendet werden. Das Encoding der ili-Datei ist UTF-8.

#208

Die Version (= Datum) des Modelles ist (via UML-Editor) anzugeben. `VERSION "2017-01-19"`

#209

Jedes Attribut muss mit einem Kommentar versehen werden, welcher das Attribut sinnvoll beschreibt.

#210

Jede Klasse muss mit einem Kommentar versehen werden, welcher die Klasse sinnvoll beschreibt.

6.3. Namenskonvention

#301

Alle Modellelemente (Modellnamen, Topics, Klassen, Attribute etc.) werden ausschliesslich auf Deutsch bezeichnet.

#302

Namen von Topics, Klassen, Assoziationen und Attributen sollte nicht länger als 29 Zeichen sein.

#303

Topic-Namen: Gross- und Kleinschrift mit Underscore als Trennzeichen. Plural.

#304

Klassen-Namen: Gross- und Kleinschrift mit Underscore als Trennzeichen. Singular.

#305

Attribut-Namen: Gross- und Kleinschrift mit Underscore als Trennzeichen. Singular. Bei Verwendung von `BAG OF` resp. `LIST OF` oder in (Publikations-)modellen bei kommaseparierten Inhalten wird der Plural verwendet.

#306

Die Verwendung von reservierten Namen ist nicht erlaubt (z.B. Name).

6.4. Modellstruktur

#401

Es müssen die Geometrietypen aus dem CHBase-Modell `GeometryCHLV95_V1` verwendet werden.

#402

Für Kantonskürzel muss `CHCantonCode` aus dem CHBase-Modell `CHAdminCodes_V1` verwendet werden

#403

BFS-Nummern sind als Wertebereich 0 .. 9999 zu definieren.

#404

Jahre sind als `GregorianYear` zu definieren.

#405

Monate sind als Wertebereich 1 .. 12 mit der Einheit (Unit) `M` zu definieren.

#406

Daten (Datum) sind als `XMLDate` zu definieren.

6.5. Einschränkungen zum Gebrauch von INTERLIS 2.3

#501

Views dürfen nur in Validierungsmodellen verwendet werden.

#502

Für `TEXT` muss immer eine konkrete Länge angegeben werden.

6.6. Konsistenzbedingungen

#601

Die Kardinalitäten von Rollen muss erfasst werden.

#602

UNIQUE-Bedingungen müssen, wo sinnvoll, erfasst werden.

#603

Den Objekten ist immer eine eindeutige Objekt-Identifikation zuzuweisen. Als OID soll in der Regel `INTERLIS.UUIDOID` verwendet werden.

6.7. Darstellungsinformationen

#701

Textpositionen werden nur definiert, wenn diese schwer aus den Daten berechnet werden können oder spezielle Anforderungen an die Darstellung bestehen.

#702

Für Labelorientierungen etc. wird die Einheit `Units.Angle_Degree` verwendet.

6.8. Allgemeines

#801

Allgemeiner Grundsatz: Es wird nur die IST-Situation beschrieben. Also weder Archivierung noch Historisierung respektive die dafür benötigten Attribute.

6.9. Beispielheader

```
INTERLIS 2.3;
/**
*
!!------------------------------------------------------------------------
--
* !! Version    | wer | Änderung
*
!!------------------------------------------------------------------------
--
* !! 2015-05-13 | SK  | Modell (v26) für Pilot durch Stefan Keller (SK)
erstellt
* !! 2016-11-11 | SK  | Überarbeitung auf Version 32
(dm_npl_ktso_v32_LV95_ili2.ili)
* !! 2016-11-29 | OJ  | Tech. Review und Finalisierung durch Oliver Jeker
(AGI)
* !! 2017-01-05 | OJ  | Korrektur Beziehungsrollennamen = Klassennamen
* !! 2017-09-01 | al  | - Lockerung der Beziehung Dokument <-> Geometrie
* !!            |     | - NP_Typ_Kanton_Grundnutzung mit N134 ergänzt
* !!            |     | - NP_Typ_Kanton_Ueberlagernd_Flaeche mit N812,N813 und
* !!            |     |   N820-823 ergänzt
* !!            |     | - Rechtschreibung bei Ueberbauungsziffer
* !!            |     | - Modell mit Beschreibung ergänzt
* !! 2017-09-15 | al  | OID AS INTERLIS.UUIDOID wieder eingefügt
* !! 2017-11-18 | sz  | - OID AS INTERLIS.UUIDOID für sämtliche Klassen
* !!            |     | - Zusätzliche Assoziation Geometrie <-> Dokument
gelöscht
* !!            |     | - Klasse Plandokument gelöscht
*
!!========================================================================
==
*/
!!@ technicalContact = "mailto:agi@bd.so.ch";
!!@ furtherInformation =
"http://geo.so.ch/models/ARP/SO_ARP_Nutzungsplanung_20171118.uml";
!!@ shortDescription="Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam."
!!@ title="Lorem ipsum dolor sit"
MODEL SO_ARP_Nutzungsplanung_20171118 (de)
  AT "https://arp.so.ch"
  VERSION "2017-11-18" =

END SO_ARP_Nutzungsplanung_20171118.
```

7. Historisierung und Archivierung

Das INTERLIS-Modell bildet die IST-Situation ab und verwaltet keine Zeitstände. Die Historisierung und Archivierung ist in der neuen GDI-Umgebung noch nicht gelöst.

Angedacht ist, dass die Archivierung der Geodaten mittels INTERLIS-Transferdateien der Erfassungsmodelle gemacht wird und die Historisierung in der Datenbank auf Basis der Publikationsmodelle geschieht. Auch aus diesem Grund soll auf Assoziationen im Publikationsmodell verzichtet werden, da die Umsetzung der Historisierung einfacher wird.

Auf den Einsatz der bestehenden Datenbank-Historisierungslösung («update_layer()») ist in der Publikationsdatenbank zu verzichten. Eine neue Historisierungslösung wird als Umsetzungsprojekt von SO!GIS 2.0 erarbeitet.