# INTERLIS

- INTER Land Information Systems
- A data description language with special consideration of **geodata**
- Object oriented and extendable
- System neutral (platform independent)
- Readable by humans and machines
- Model driven approach

## Model definintion and data files

The model is defined in INTERLIS language and stored in an `.ili` file.

The data is in xml (considering the model) and stored as an `.xtf` file (former `.itf`).

## What made me to like INTERLIS

> With INTERLIS that you have your database schema in your poket.

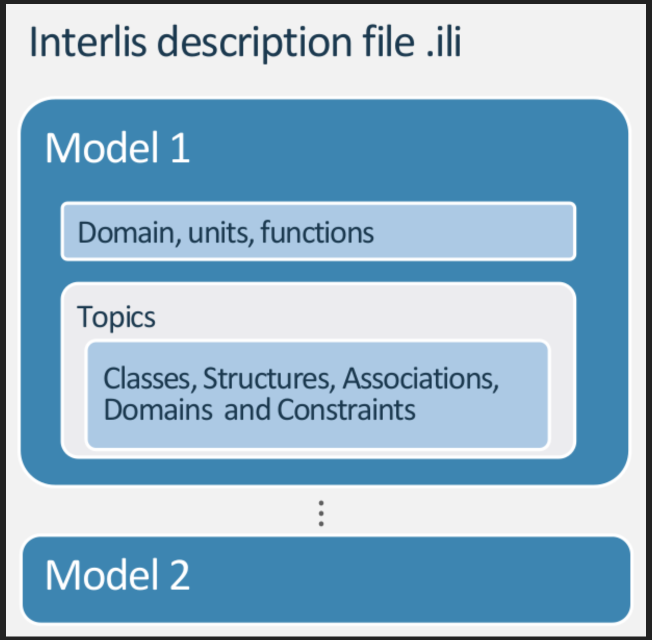
> It's easy readable and precice. Compared to e.g. SQL Scripts you can simply extend it.

> Thanks to the nice tools (ili2 and Model Baker) it's easy to implement in your database and in QGIS.

## INTERLIS Modelling in 10 Minutes

## Model Structure

```
INTERLIS 2.3;
MODEL Wildruhezonen_LV95_V2_1 (de)
AT "https://models.geo.admin.ch/BAFU/"
VERSION "2020-04-21"  =
  DOMAIN
    Punkt = GeometryCHLV95_V1.Coord2;
  TOPIC Wildruhezonen =
    CLASS Routennetz =
      Name : MANDATORY TEXT*80;
    END Routennetz;
  END Wildruhezonen;
END Wildruhezonen_LV95_V2_1.
```


Interlis description file .ili
Model 1
Domain, units, functions
Topics
Classes, Structures, Associations, Domains and Constraints
⋮
Model 2

## Classes

### Syntax

```
ClassDef =  'CLASS' Class-Name '='
              { AttributeDef }
            'END' Class-Name ';'.
```

### Example

```
CLASS Wildruhezone =
  ObjNummer : MANDATORY 0 .. 9999;
  Name : MANDATORY TEXT*80;
END Wildruhezone;
```

## Attributes

### Syntax

```
AttributDef = Attribute-Name : [MANDATORY]
                               Type | DomainRef;

DomainRef = [ Model-Name '.' [ Topic-Name '.' ] ] Domain-Name
```

### Example

```
Name : MANDATORY TEXT*80;
Schutzstatus : MANDATORY Wildruhezonen_Codelisten_V2_1.Codelisten.Schutzstatus_CatRef;
```

## Structures

### Syntax

```
StructureDef =  'STRUCTURE' Struct-Name '='
                  { AttributeDef }
                'END' Struct-Name ';'.
```

### Example

```
STRUCTURE PolygonStructure =
  Polygon: SURFACE WITH (STRAIGHTS) VERTEX GeometryCHLV03_V1.Coord2 WITHOUT OVERLAPS > 0.001;
END PolygonStructure;

STRUCTURE MultiPolygon =
  Polygons: BAG {1..*} OF PolygonStructure;
END MultiPolygon;
```

## Accociations

### Syntax

```
AssociationDef = 'ASSOCIATION' '='
                   { RoleDef }
                 'END' ';'.
RoleDef = Role-Name '--' ClassRef ';'.
```

### Example

```
ASSOCIATION RoutennetzWildruhezone =
  WRZ_Routennetz -- {0..*} Routennetz;
  WRZ -<#> {1} Wildruhezone;
END RoutennetzWildruhezone;
```

## Extends

```
CLASS Wildruhezone =
  ObjNummer : MANDATORY 0 .. 9999;
  Name : MANDATORY TEXT*80;
END Wildruhezone;

CLASS Wildruhezone (EXTENDED) =
    /** Zuordnung der Zielarten Schutzbestimmung zur Wildruhezone */
    Zielart: GL_Wildruhezonen_Codelisten_V1.Codelisten.Zielarten_CatRef;
END Wildruhezone;
```

## Types of classes

- Concrete
- Abstract
- Final
- Derivate/Extended

```
CLASS Wildruhezone (ABSTRACT)=
END Wildruhezone;
```

## What are catalogue

Catalogues are kind of data.

Catalogues are external codelists that can be used like `Enumerations` but less static.

## Structure of a catalogue

Catalogues base on the model `CatalogueObjects_V1` and extend the abstract classes and structures

```
CLASS Bestimmungen_Catalogue
EXTENDS CatalogueObjects_V1.Catalogues.Item =
    Code : MANDATORY TEXT*5;
    Description : MANDATORY LocalisationCH_V1.MultilingualText;
END Bestimmungen_Catalogue;

STRUCTURE Bestimmungen_CatRef
EXTENDS CatalogueObjects_V1.Catalogues.MandatoryCatalogueReference =
    Reference (EXTENDED) : MANDATORY REFERENCE TO (EXTERNAL) Bestimmungen_Catalogue;
END Bestimmungen_CatRef;
```

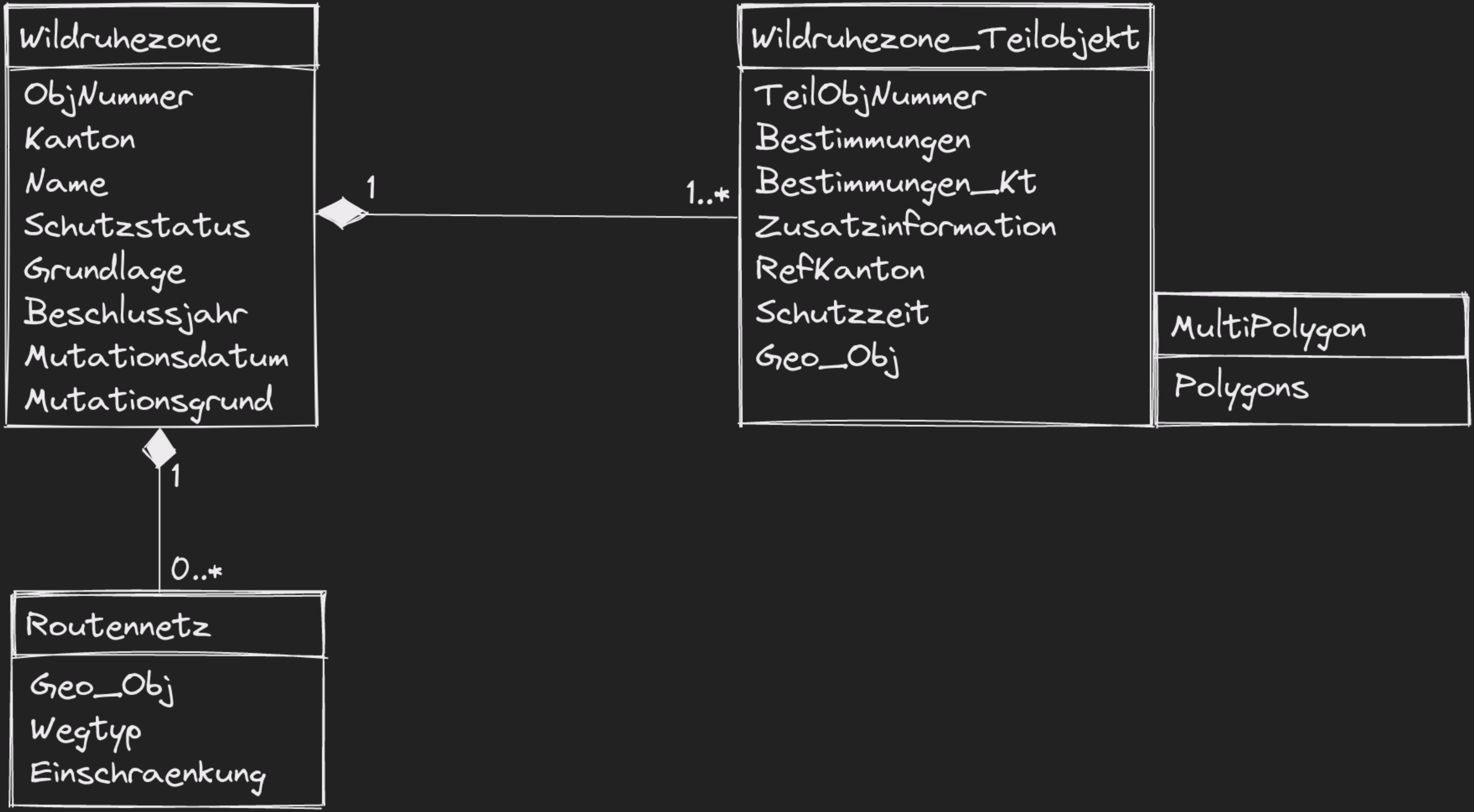
## Reference to the catalogue

```
CLASS Wildruhezone_Teilobjekt =
  Bestimmungen : MANDATORY Wildruhezonen_Codelisten_V2_1.Codelisten.Bestimmungen_CatRef;
END Wildruhezone_Teilobjekt;
```

## Have a look at a simple model

Buildings

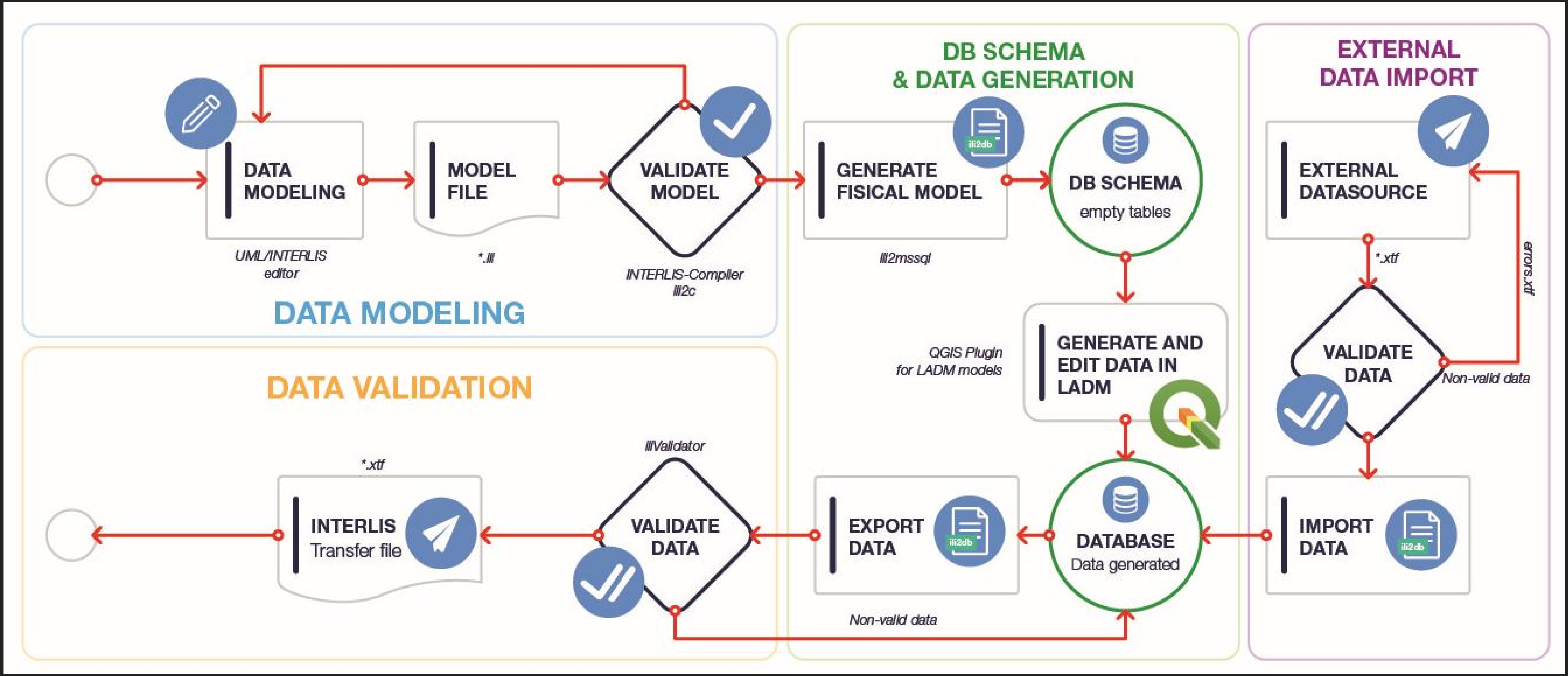
# Have a look at a real ILI file Wildruhezonen_V2_1



## Check out the real extended model for Glarus

Wildruhezonen_V2_1

## INTERLIS implementation workflow and tools



(Graphic by landnetwork.ch)

## ili2 Tools

made by Eisenhut Informatik

## Compiler ili2c

The INTERLIS Compiler checks an INTERLIS model if the constructs of the language INTERLIS were applied correctly. It reports syntactic errors in the model with the line number so that they can be corrected by the modeler.

## ili2fme and ili2db

ili2pg, ili2gpkg and ili2fgdb are programs that write an INTERLIS transfer file according to an INTERLIS model into a database (PostgrSQL/PostGIS, GeoPackage or ESRI FileGDB) or create such a transfer file from a database.

## ilivalidator

The ilivalidator tool checks whether data in the INTERLIS 1 and 2 transfer format (*.itf/*.xtf) complies with the associated model (*.ili). License terms and further information about the ilivalidator can be found here:

## Swiss geodata repositories

## ilimodels.xml

- Based on the model `IliRepository09`
- Contains objects of the class `ModelMetadata` where a model name and a file path is defined
- The files are on the same repository

## ilisites.xml

- Based on the model `IliSite09`
- Contains objects of the class `SiteMetadata` where path to other repositories are defined
  http://models.interlis.ch/ilisite.xml -> http://models.geo.kgk-cgc.ch/ilisite.xml -> http://models.geo.sh.ch/ilisite.xml

*Let's have a look*

## 🧁 QGIS MODEL BAKER

## While INTERLIS is the hard stuff

## MODEL BAKER is the beginner drug

## A QGIS Project Generator

Quickly **create a QGIS project** from a physical data model.

Analyzes the existing structure and configures a QGIS project with all available information.

## A QGIS Project Generator optimized for INTERLIS

Models defined in INTERLIS provide additional meta information like domains, units of attributes or object oriented definitions of tables.

This can be used to further optimize the project configuration.

## An ili2db controll station

```
java -jar /home/dave/dev/opengisch/QgisModelBaker/QgisModelBaker/libili2db/bin/ili2pg-4.6.1/ili2pg-4.6.1.jar --schemaimport --dbhost localhost --dbport 5432
--dbusr postgres --dbpwd ****** --dbdatabase bakery --dbschema adsfdsaf2 --setupPgExt --coalesceCatalogueRef --createEnumTabs --createNumChecks --createUnique
--createFk --createFkIdx --coalesceMultiSurface --coalesceMultiLine --coalesceMultiPoint --coalesceArray --beautifyEnumDispName --createGeomIdx --createMetaInfo
--expandMultilingual --createTypeConstraint --createEnumTabsWithId --createTidCol --importTid --smart2Inheritance --strokeArcs --defaultSrsCode 2056
--models Wildruhezonen_LV95_V2_1
```

## And it's a library

Can be used as a framework.

Like Asistente LADM-COL, created for the Colombian implementation of the Land Administration Domain Model (LADM) does it.

## Check it out now

## What is the UsabILIty Hub?

Receive meta data like ***ili2db settings***, ***layer styles*** and ***orders*** etc. automatically over the web.

**See https://usabilityhub.opengis.ch/**

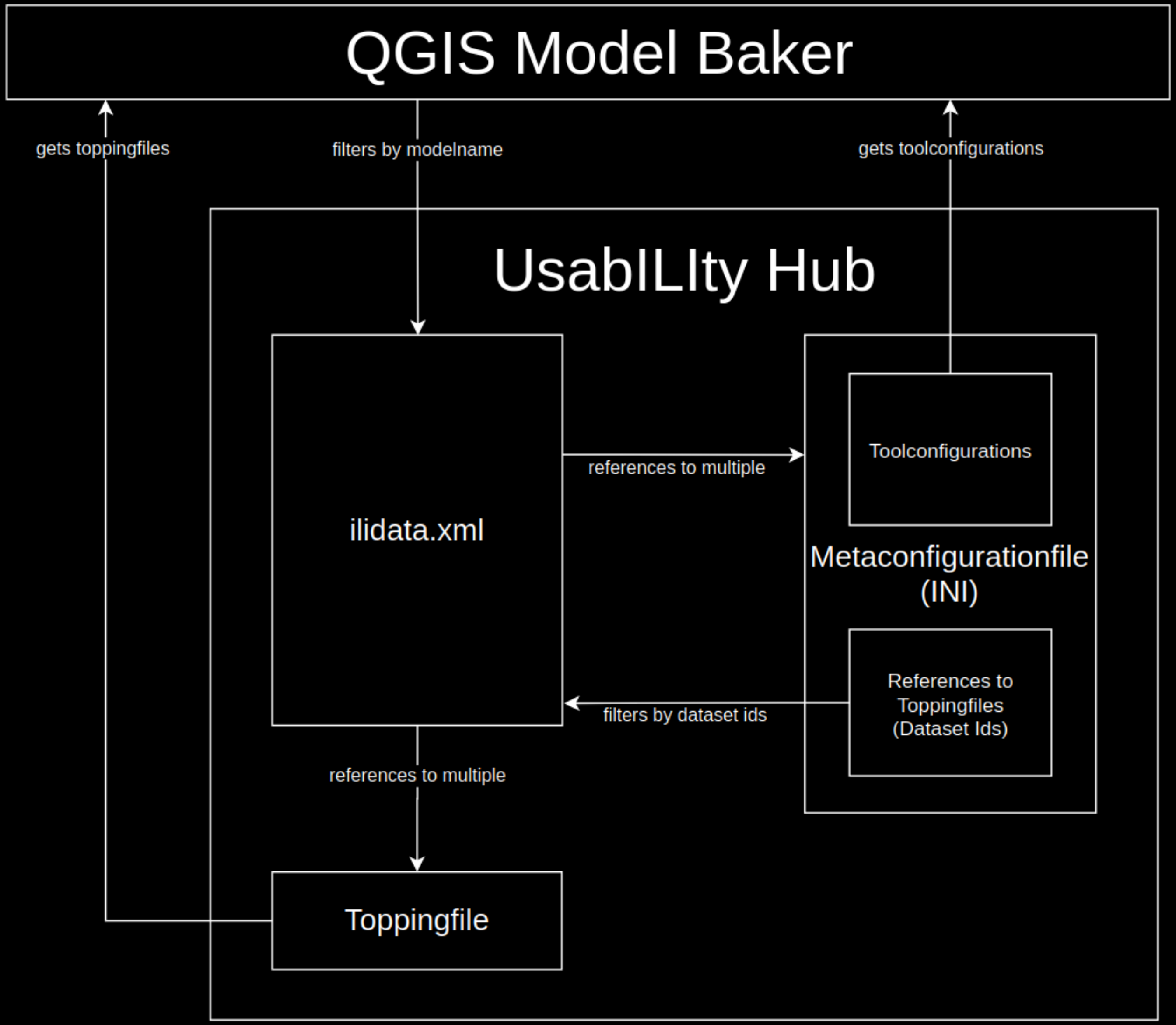
### Metaconfiguration and Toppings

Get the additional information with the `ilidata.xml` file on the UsabILIty Hub (currently https://models.opengis.ch) and the linked repositories.

## Metaconfiguration and Toppings

Settings for tools are configured in a metaconfiguration file, as well as links to topping files that contain information about GIS project.

***Thus, this additional information usually consists of a metaconfiguration and any number of toppings.***


QGIS Model Baker
gets toppingfiles
filters by modelname
gets toolconfigurations
UsabILIty Hub
ilidata.xml
references to multiple
Toolconfigurations
Metaconfigurationfile (INI)
References to Toppingfiles (Dataset Ids)
filters by dataset ids
references to multiple
Toppingfile

## Why not using INTERLIS