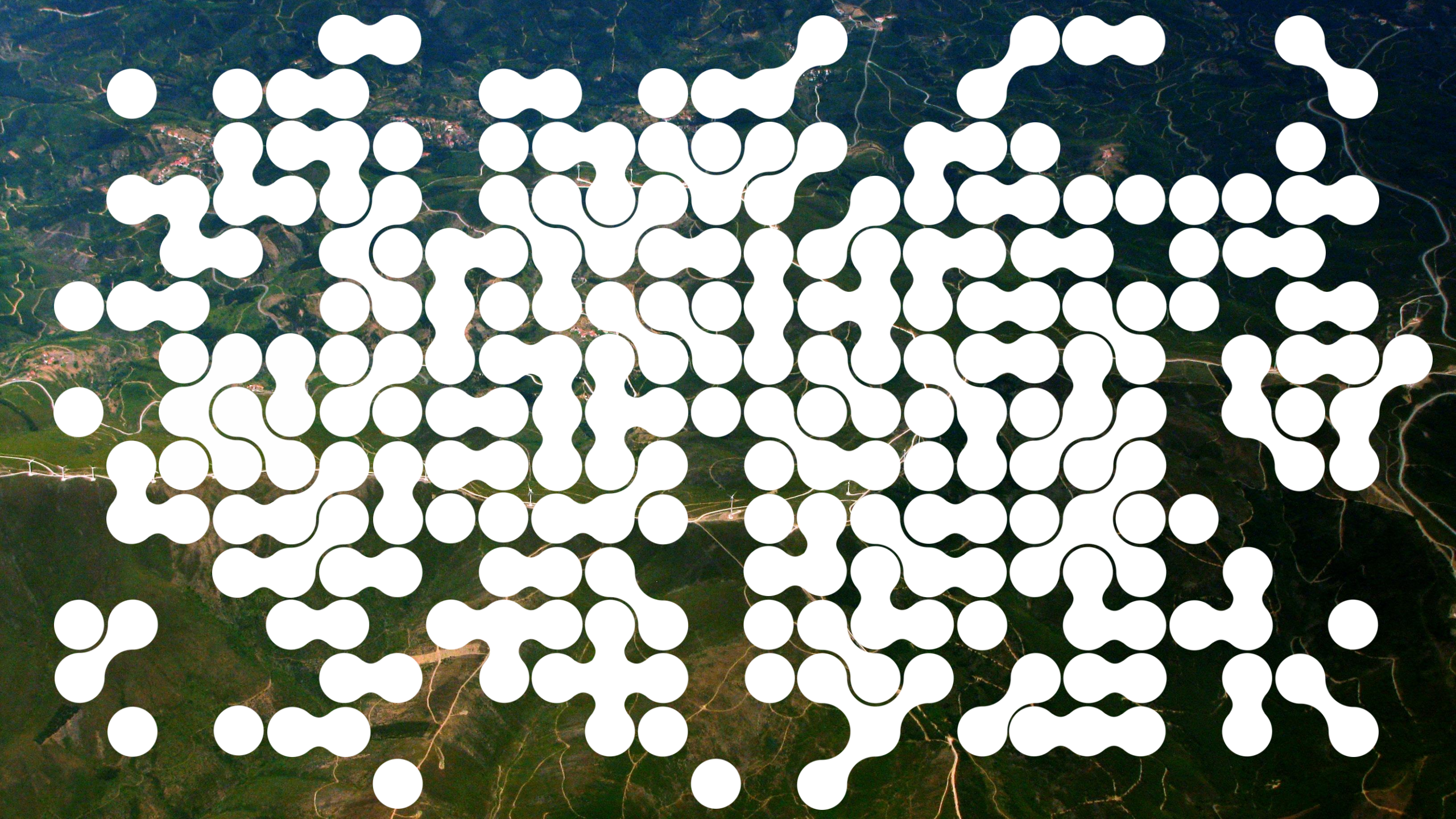



LandNetwork

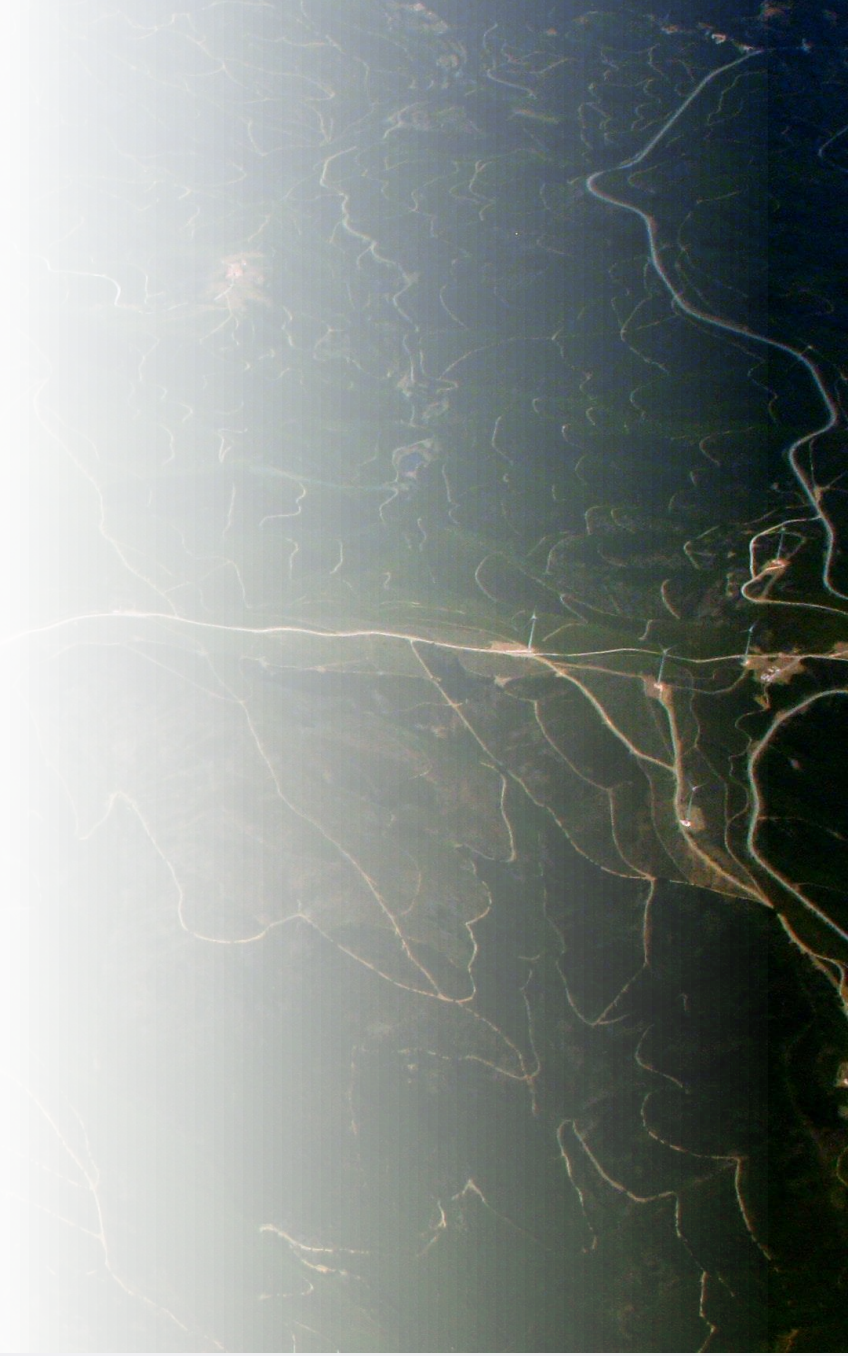
INTERLIS Training

Fabian Mejia
Land Administration Specialist
Fabian.mejia@landnetwork.ch

LandNetwork

Agenda

- Using INTERLIS/UML Editor
- Ili2db
  - General reference
  - SmartMapping
- Constraints
- Functions



Course Resources

Tools

- Text Editor: Notepad ++ (Recommended)
- QGis 3.16 +
- PostgreSQL 9.6 or upper + PostGIS 2.3 or upper
- Java VM (JRE 1.6 or upper)

Reference:

https://www.interlis.ch/download/interlis2/ili2-refman_2006-04-13_e.pdf

https://drive.infomaniak.com/app/share/189474/c1d19a50-43c8-4b34-b238-d4f7814f37c7

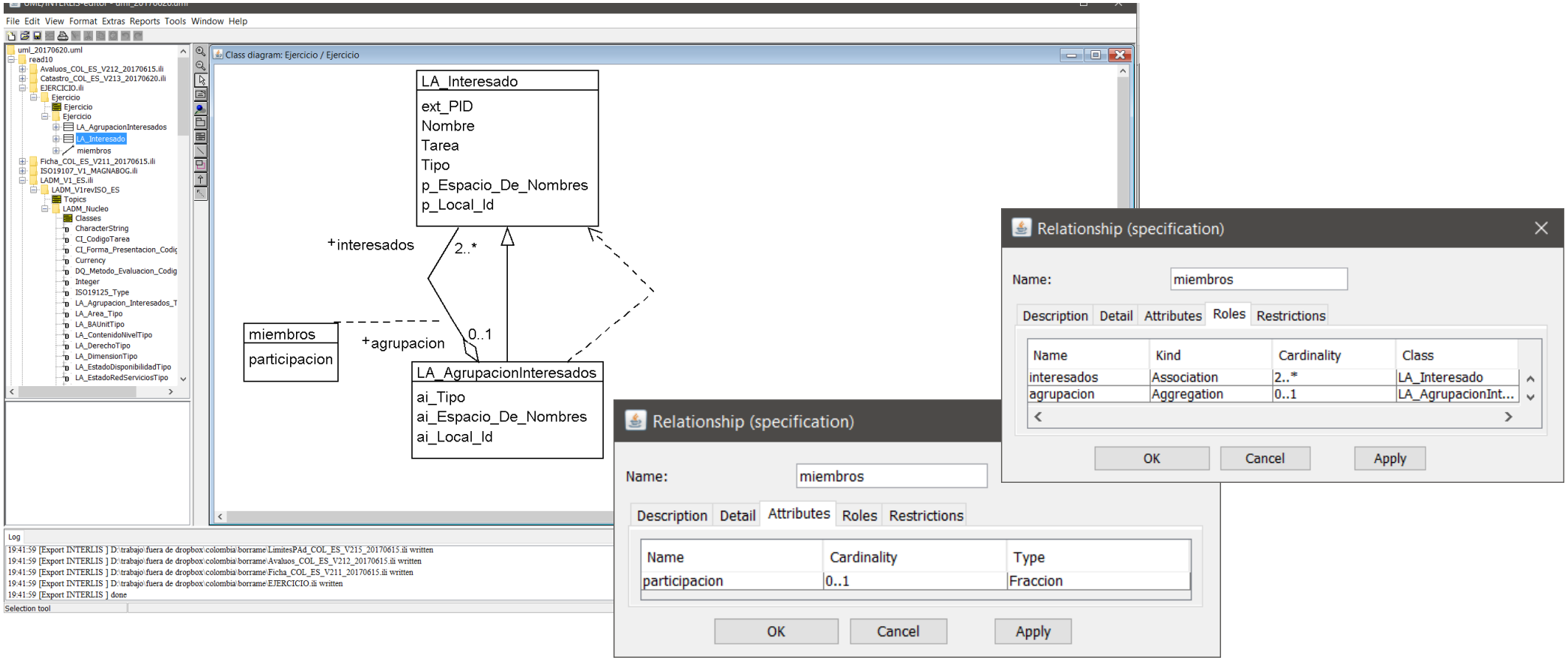
LandNetwork

Hands-on
Using modelling tools

Tools: UML/INTERLIS Editor

https://github.com/claeis/umleditor

https://github.com/SwissTierrasColombia/umleditor



LandNetwork

Reverse Engineering

- Create an UML Model representing attached .ili file

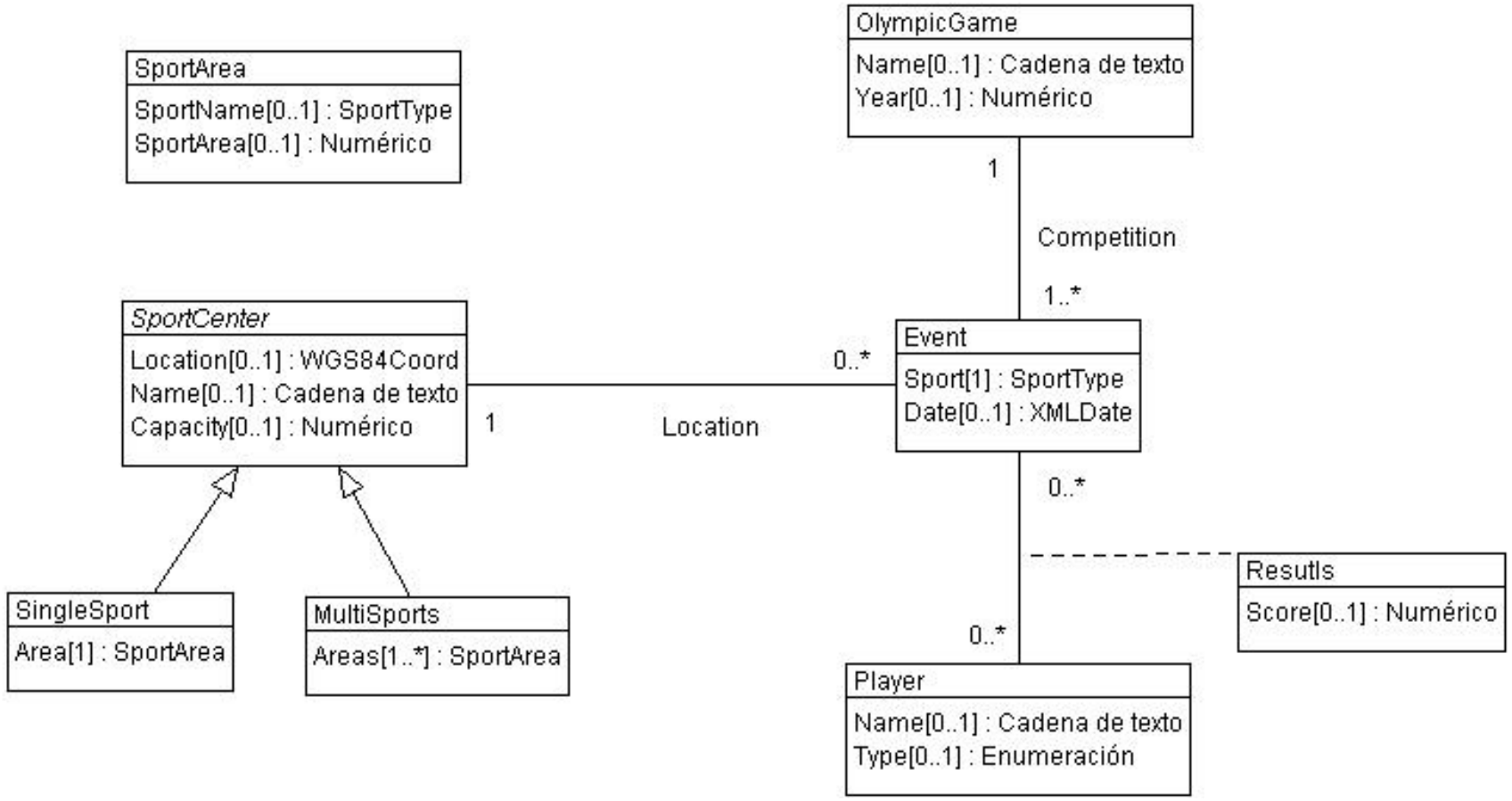
```
INTERLIS 2.3;

MODEL OlimpycGames (en_US)
AT "mailto:Fabian@localhost"
VERSION "2021-06-23"  =
  IMPORTS CoordSys,Units;

  REFSYSTEM BASKET
  BCoordSys ~ CoordSys.CoordsysTopic
  OBJECTS OF GeoEllipsoidal : WGS84;

  DOMAIN

    SportType = (
      Athletics,
      Boxing,
      Archery,
      Swimming,
      Gymnastics,
      Ski
    );
```

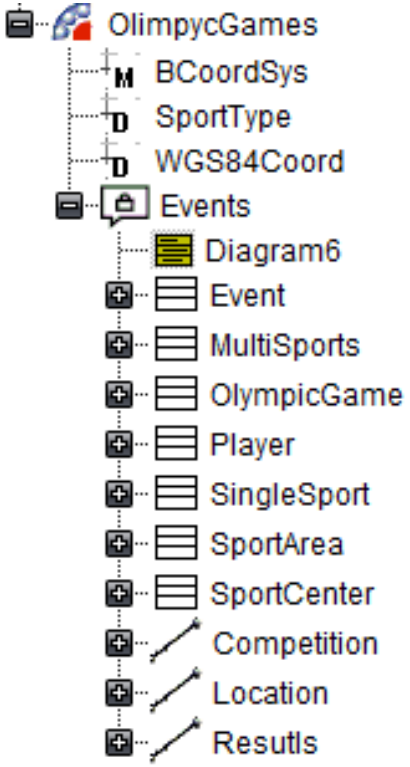
```
  WGS84Coord = COORD -90.000 .. 90.000 [Units.Angle_Degree]{WGS84[1]}, 0.000 .. 359.999 CIRCULAR
  [Units.Angle_Degree]{WGS84[2]};

TOPIC Events =

  CLASS Event =
    Sport : MANDATORY OlimpycGames.SportType;
    Date : INTERLIS.XMLDate;
  END Event;

  CLASS OlympicGame =
    Name : MANDATORY TEXT*50;
    Year : MANDATORY 1896 .. 2500;
  END OlympicGame;

  CLASS Player =
    Name : TEXT*150;
    Type : (
      MultiPlayer,
      SinglePlayer
    );
  END Player;

  STRUCTURE SportArea =
    SportName : OlimpycGames.SportType;
    SportArea : 0.00000 .. 1.00000 [Units.km2];
  END SportArea;
```

```
CLASS SportCenter (ABSTRACT) =
  Location : OlimpycGames.WGS84Coord;
  Name : TEXT*150;
  Capacity : 0 .. 1000000;
END SportCenter;

CLASS MultiSports
EXTENDS SportCenter =
  Areas : BAG {1..*} OF OlimpycGames.Events.SportArea;
END MultiSports;

CLASS SingleSport
EXTENDS SportCenter =
  Area : MANDATORY OlimpycGames.Events.SportArea;
END SingleSport;

ASSOCIATION Competition =
  olympics -- {1} OlympicGame;
  competition -- {1..*} Event;
END Competition;

ASSOCIATION Location =
  event -- {0..*} Event;
  center -- {1} SportCenter;
END Location;
```

```
    ASSOCIATION Resutls =
      event -- {0..*} Event;
      player -- {0..*} Player;
      Score : MANDATORY 0 .. 10;
      Reward : MANDATORY 0 .. 10000000 [Units.USD];
    END Resutls;

  END Events;

END OlimpycGames.
```

Expected Result





LandNetwork

Extending a Paralympic Model

- Specialize some sports

  Athletics : 100m, HalfMarathon, Marathon

  Swimming: Butterfly, Freestyle
- Minimum reward will be 1.000 USD
- Player: add disability attribute and its corresponding domain
- Sport center: include HasWheelChairAccess attribute (BOOLEAN)
- Some pictures for events will be nice!

Lines, surfaces

Syntaxis

```
Line =
  POLYLINE Form
    Vertex [Intersectiondef]

Form =
  WITH (STRAIGHTS,ARCS)

Vertex =
  VERTEX CoordinateType

Intersectiondef =
  WITHOUT OVERLAPS > Tol
```

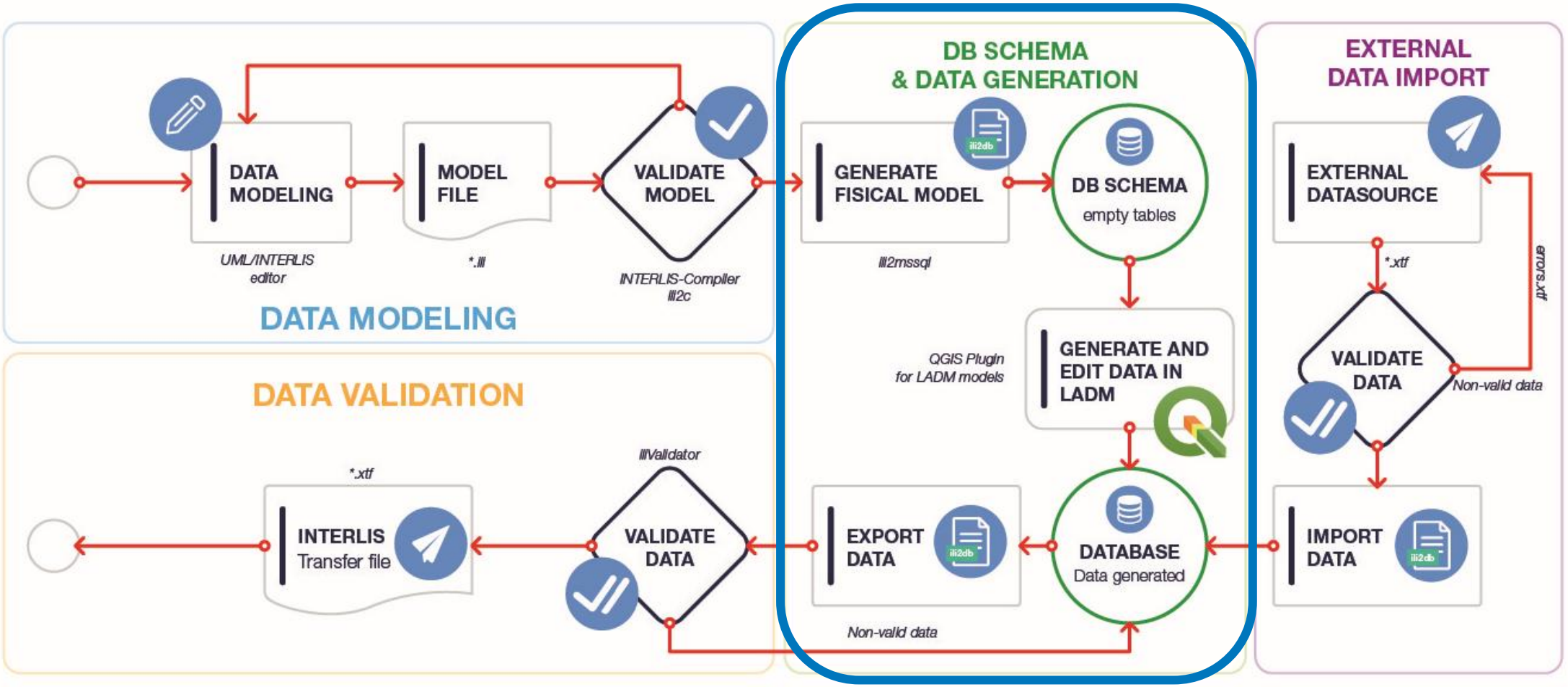
Example

```
Geometry: POLYLINE WITH (STRAIGHTS,ARCS)
  VERTEX ZH_Coord;

Surface: SURFACE WITH (STRAIGHTS) VERTEX GeometryCHLV95_V1.Coord2
    WITHOUT OVERLAPS > 0.005;
```

LandNetwork

Moving from conceptual to physical models

Typical INTERLIS implementation Workflow



Mejía et al., 2017

LandNetwork

Ili2db main functions

create DB schema based on ili model

- --schemaimport
- options to control the mapping

Data import

- --import
- options to define what happens with existing records

Data export

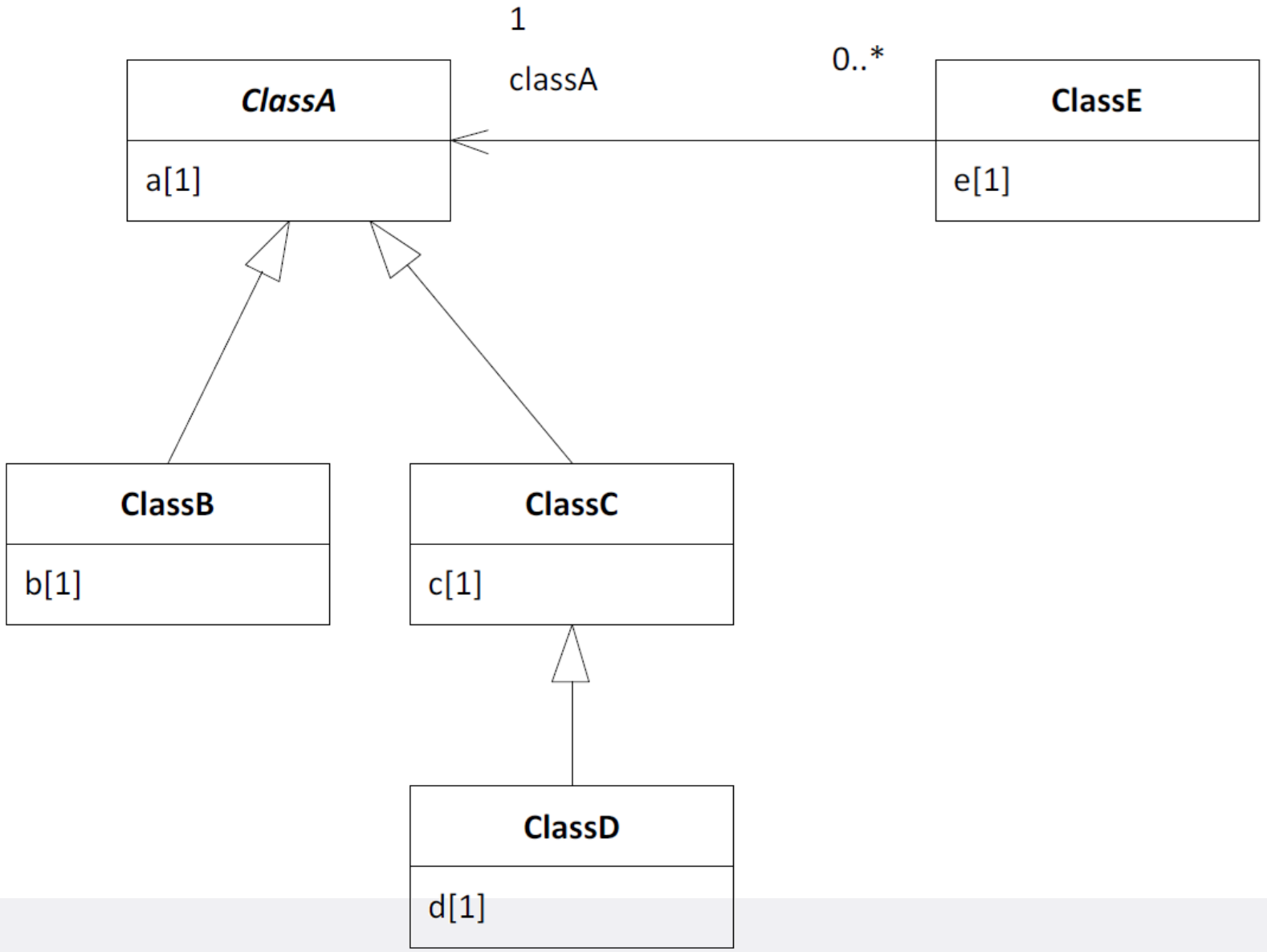
- --export
- options to define what records are exported

Inheritance Mapping

Common patterns

- New Class
- Super Class
- Sub Class
- New+Sub Class

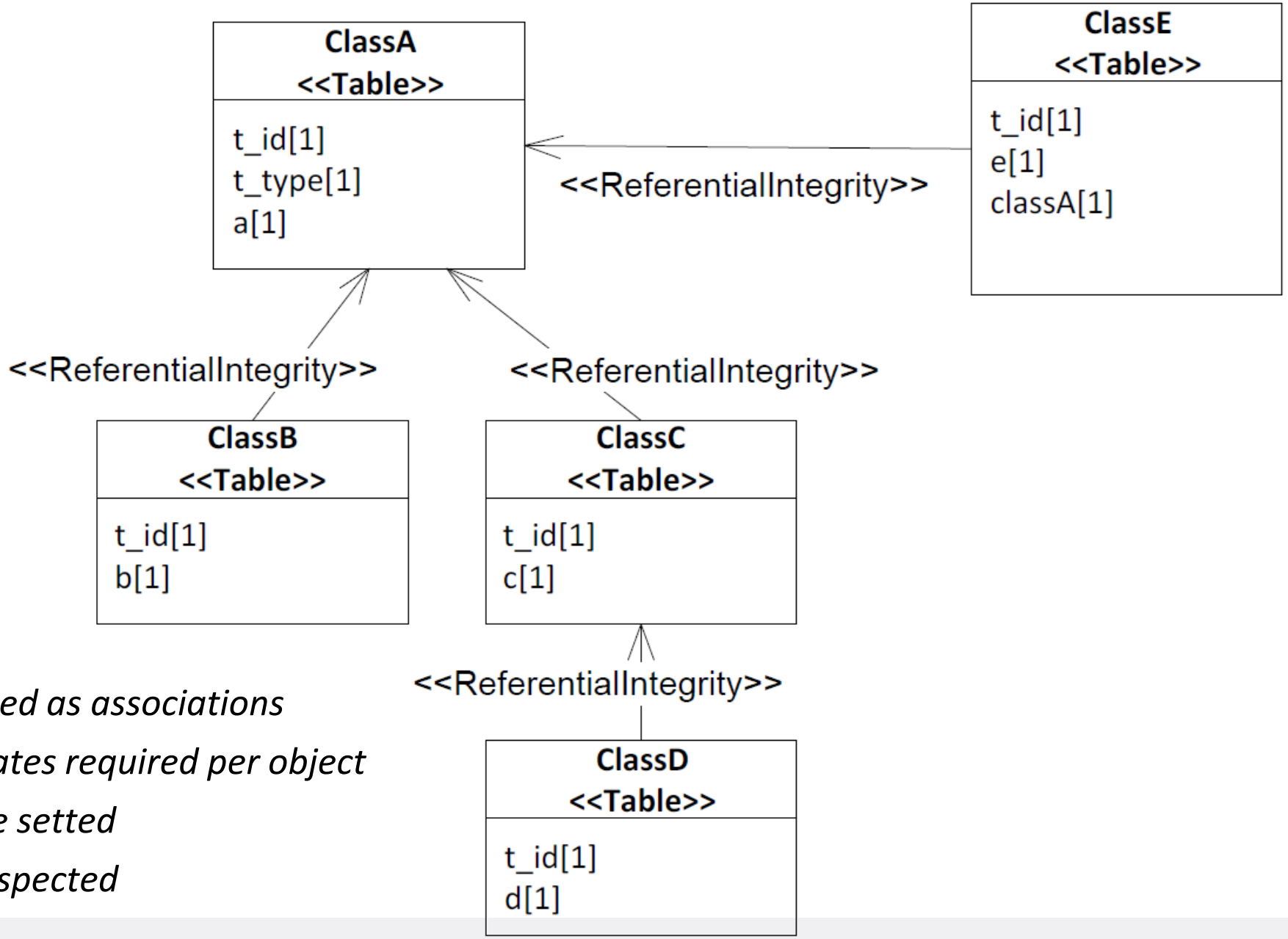
Sample Model



LandNetwork

New Class

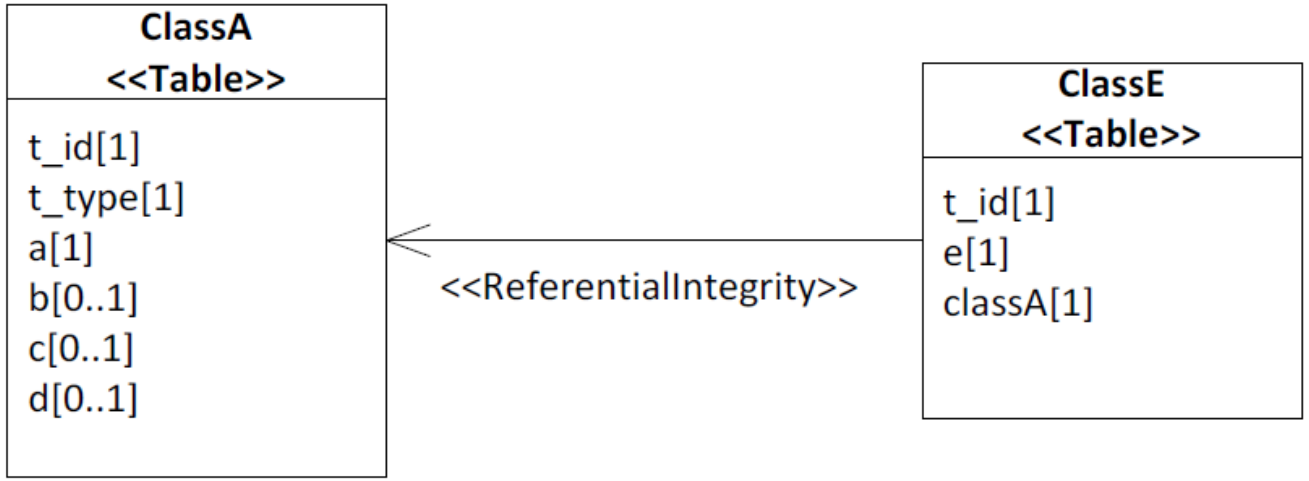
```
ClassA.t_type: (
                    ClassB,
                    ClassC,
                    ClassD
                )
```



- *Specializations are mapped as associations*
- *Multiple inserts and updates required per object*
- *Not null attributes can be setted*
- *Referential integrity is respected*

Super Class

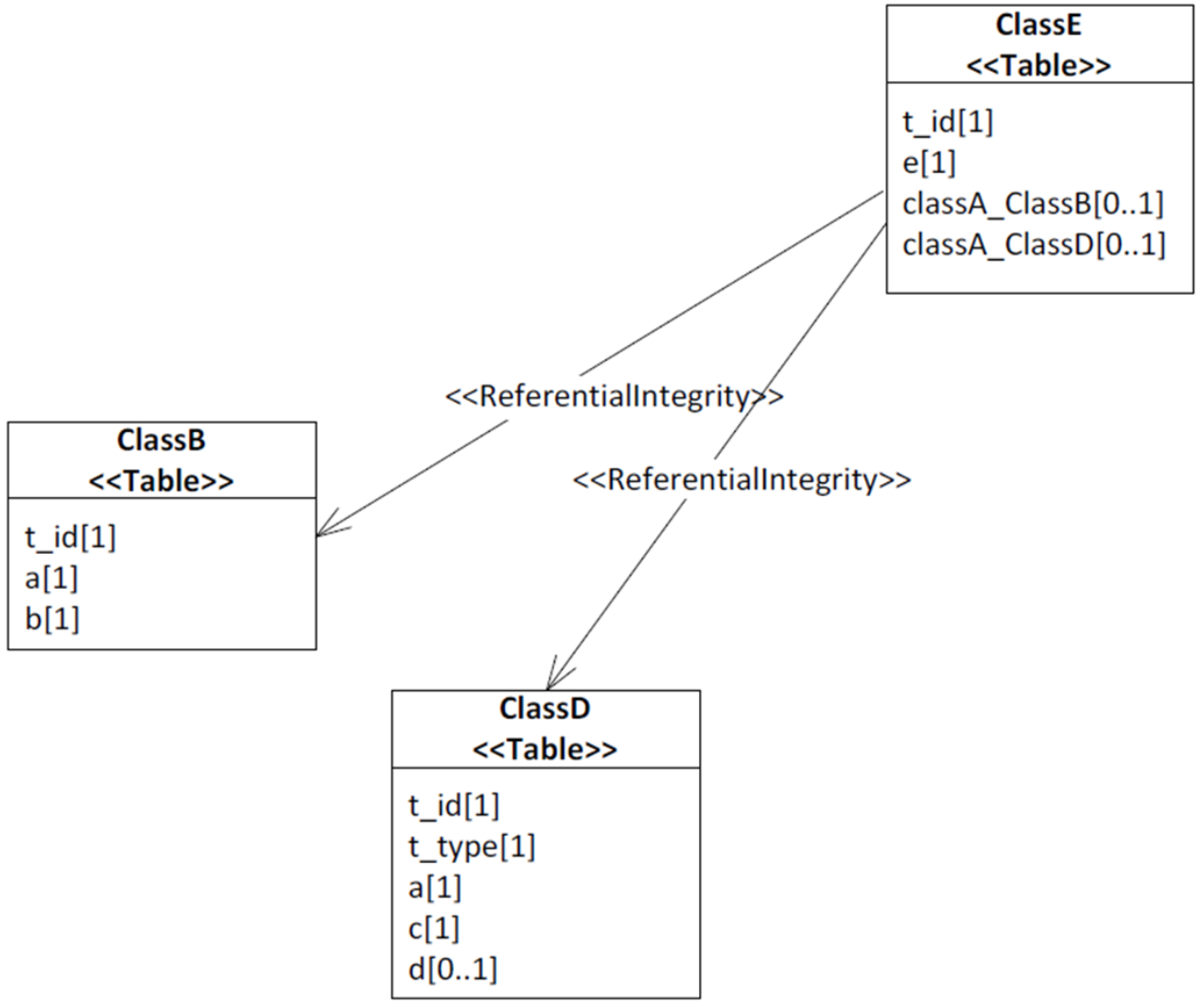
```
ClassA.t_type: (
                    ClassB,
                    ClassC,
                    ClassD
                )
```



- *Missing Not null constraints*
- *Less tables and associations (easy to use)*

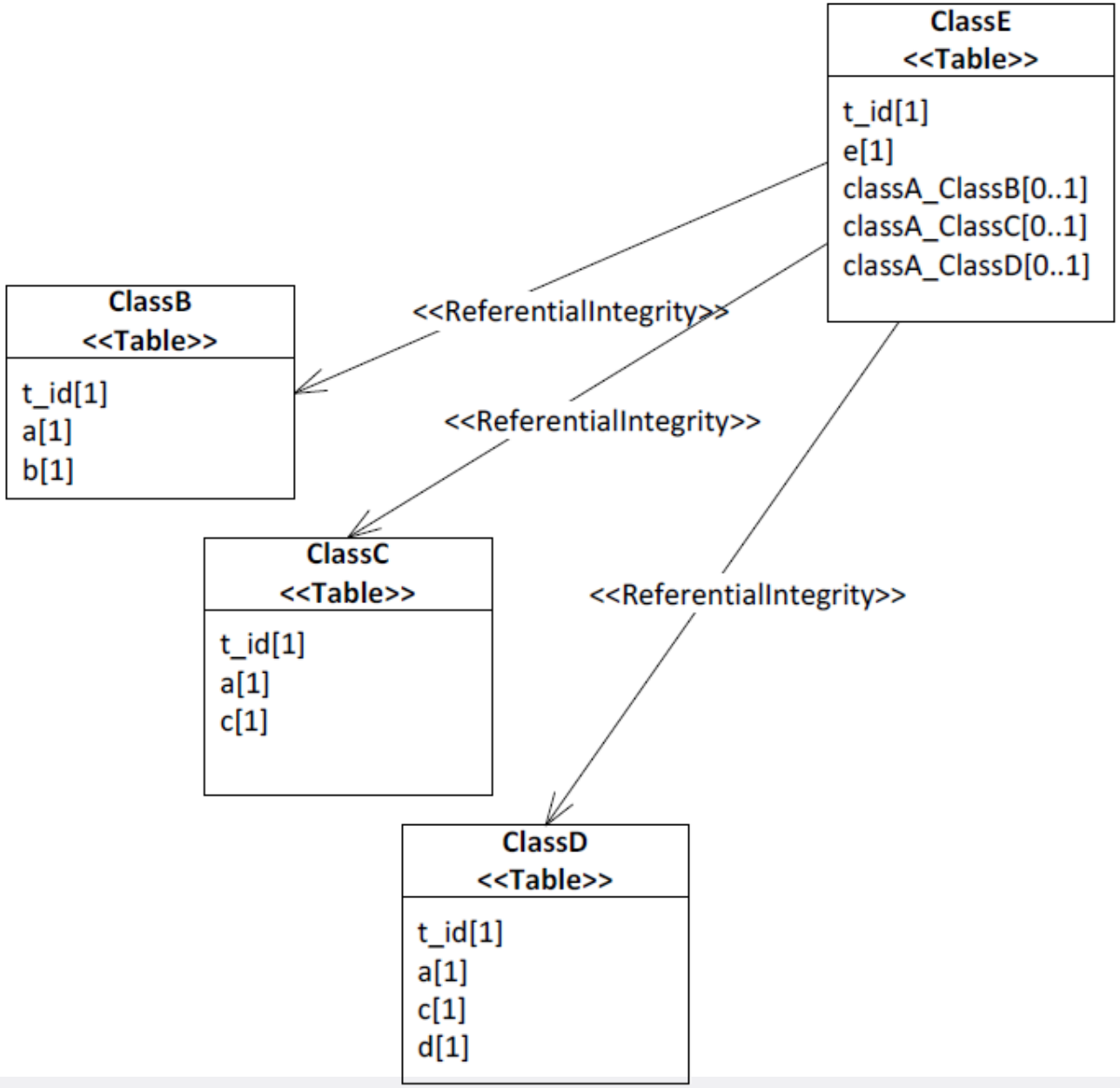
Sub Class

```
ClassD.t_type: (
                    ClassC,
                    ClassD
               )
```



- *Missing Not null constraints*

New + Sub Class



- *Missing Not null constraints*
- *Referential integrity is respected*

Ili2db smart Mapping

--noSmartMapping

- All classes are mapped using New Class Strategy

--smart1Inheritance

- Abstract classes without associations are mapped using Sub Class
- Abstract classes with associations and no concrete Super Class, use New Class
- Concrete classes without concrete Super Class, use New Class
- All other classes are mapped using Super Class

--smart2Inheritance

- Abstract classes Mapped using Sub Class
- All Concrete classes mapped using New and Sub Class

Custom mapping

Enumerations

- --createEnumTabs (create an additional table per enumeration type)
- --createSingleEnumTab (create one single additional table for all enumeration types)
- --createEnumTxtCol (create an additional redundant column: same value as dispName in enum tables)

Structures

- --coalesceMultiSurface (refactors GeometryCHLV03_V1. MultiSurface to a column with type MULTISURFACE)
- --expandMultilingual (refactors LocalisationCH_V1. MultilingualText to five columns for de, en, fr, it, rm)

Constraints

Constraints

- --sqlEnableNull (all columns of Interlis attributes are nullable independent of MANDATORY)
- --createFk (create a foreign key constraint on colums that reference other tables)
- --createUnique (maps INTERLIS-UNIQUE-constraints to SQL-UNIQUE-constraints)
- --createNumChecks (create SQL-CHECK-constraints to validate the numeric range)

Geometry

- --defaultSrsAuth auth (SRS authority, if not evaluable from the Interlis model. Default: EPSG)
- --defaultSrsCode code (SRS code, if not evaluable from the Interlis model. Default: 21781)
- --strokeArcs (stroke arcs on –import)
- --oneGeomPerTable (create helper tables, if the Interlis class has more than one geometry attribute, so that the SQL tables have not more than one geometry column)

LandNetwork

INTERLIS Constraints

Constraints

- Rules that must be followed by a combination of values or certain objects
- Are part of the definition of Classes, Associations or Views

Mandatory

Applies to all objects of the class, constraint is evaluated for each one.

```
MandatoryConstraint = 'MANDATORY' 'CONSTRAINT'
                      Logical-Expression ';'
```

```
CLASS Employee =
  firstName : MANDATORY TEXT*30;
  isRetired : BOOLEAN;
  salary : 0..250000 [CHF];
  MANDATORY CONSTRAINT
    !! If employee is not yet retired,
    !! a salary must be defined
    isRetired==#true OR DEFINED(salary);
END Employee;
```

Constraints

Unique

Describes a combination of attributes that must be unique across all object on the basket

```
UniquenessConstraint = 'UNIQUE' GlobalUniqueness ';'.

GlobalUniqueness = UniqueEl.

UniqueEl = ObjectOrAttributePath { ','
                    ObjectOrAttributePath }
```

```
CLASS Employee =
   firstName : TEXT*30;
   lastName : TEXT*30;
   taxIdentificator: Mandatory TEXT*10;
   !! All Employees have a unique Tax ID
   !! All employees have a unique name
   UNIQUE taxIdentificator;
   UNIQUE firstName, lastName;
END Employee;
```

Constraints

Set

Used to describe more complex rules that must be followed for a group (set) of object inside the basket

```
SetConstraint = 'SET' 'CONSTRAINT'
                                Logical-Expression ';'.
```

```
CLASS Parcel =
   type: (private, public, ….);
   Geometry: SURFACE ...;
   SET CONSTRAINT WHERE type = #public :
      areAreas(ALL, UNDEFINED, >> Geometry);
END B;
```

All public parcels (type = Public) should be an aggrupation without overlaps and gaps (Tessellation)

Functions

- Used in constraints to add logic to validations
- Usable to return complex calculations

Custom function workflow:

Define the function -> use the function -> implement the function (external tools)

Predefined functions Use qualified name (INTERLIS.len())

Text Functions

- FUNCTION len (TextVal: TEXT): NUMERIC;
- FUNCTION lenM (TextVal: MTEXT): NUMERIC;
- FUNCTION trim (TextVal: TEXT): TEXT;
- FUNCTION trimM (TextVal: MTEXT): MTEXT;

Element count Functions

- FUNCTION elementCount (bag: BAG OF ANYSTRUCTURE): NUMERIC;
- FUNCTION objectCount (Objects: OBJECTS OF ANYCLASS): NUMERIC;

Functions

Check Type Functions

- FUNCTION myClass (Object: ANYSTRUCTURE): STRUCTURE;
- FUNCTION isSubClass (potSubClass: STRUCTURE; potSuperClass: STRUCTURE): BOOLEAN;
- FUNCTION isOfClass (Object: ANYSTRUCTURE; Class: STRUCTURE): BOOLEAN;
- FUNCTION isEnumSubVal (SubVal: ENUMTREEVAL; NodeVal: ENUMTREEVAL): BOOLEAN;
- FUNCTION inEnumRange (Enum: ENUMVAL; MinVal: ENUMTREEVAL; MaxVal: ENUMTREEVAL): BOOLEAN;

Misc. Functions

- FUNCTION convertUnit (from: NUMERIC): NUMERIC;
- FUNCTION areAreas (Objects: OBJECTS OF ANYCLASS; SurfaceBag: ATTRIBUTE OF @ Objects RESTRICTION
  (BAG OF ANYSTRUCTURE); SurfaceAttr: ATTRIBUTE OF @ SurfaceBag RESTRICTION (SURFACE)): BOOLEAN;


Thank you
Fabian.mejia@landnetwork.ch