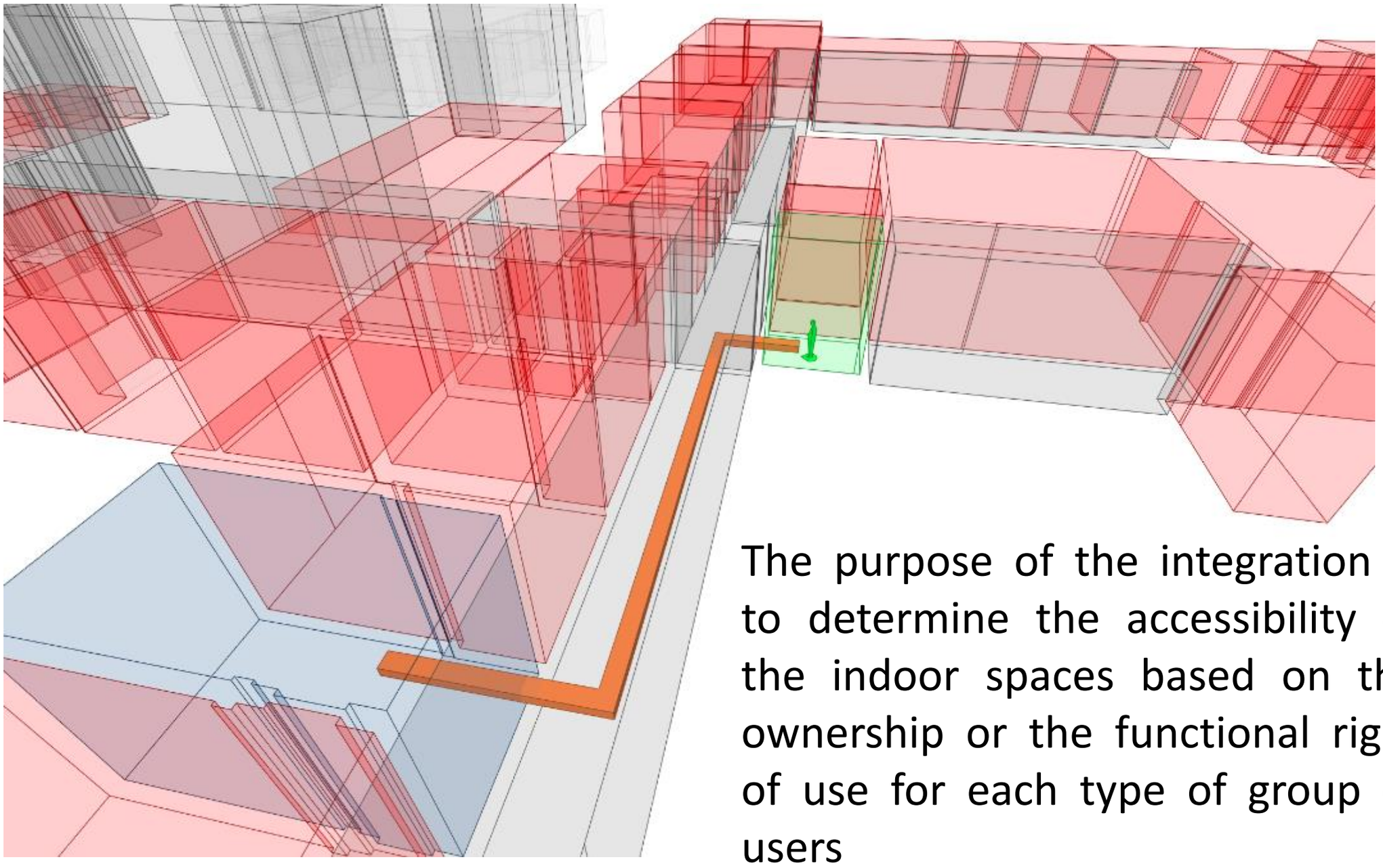
Deriving the technical model for the indoor navigation prototype based on the integration of IndoorGML and LADM Conceptual Model

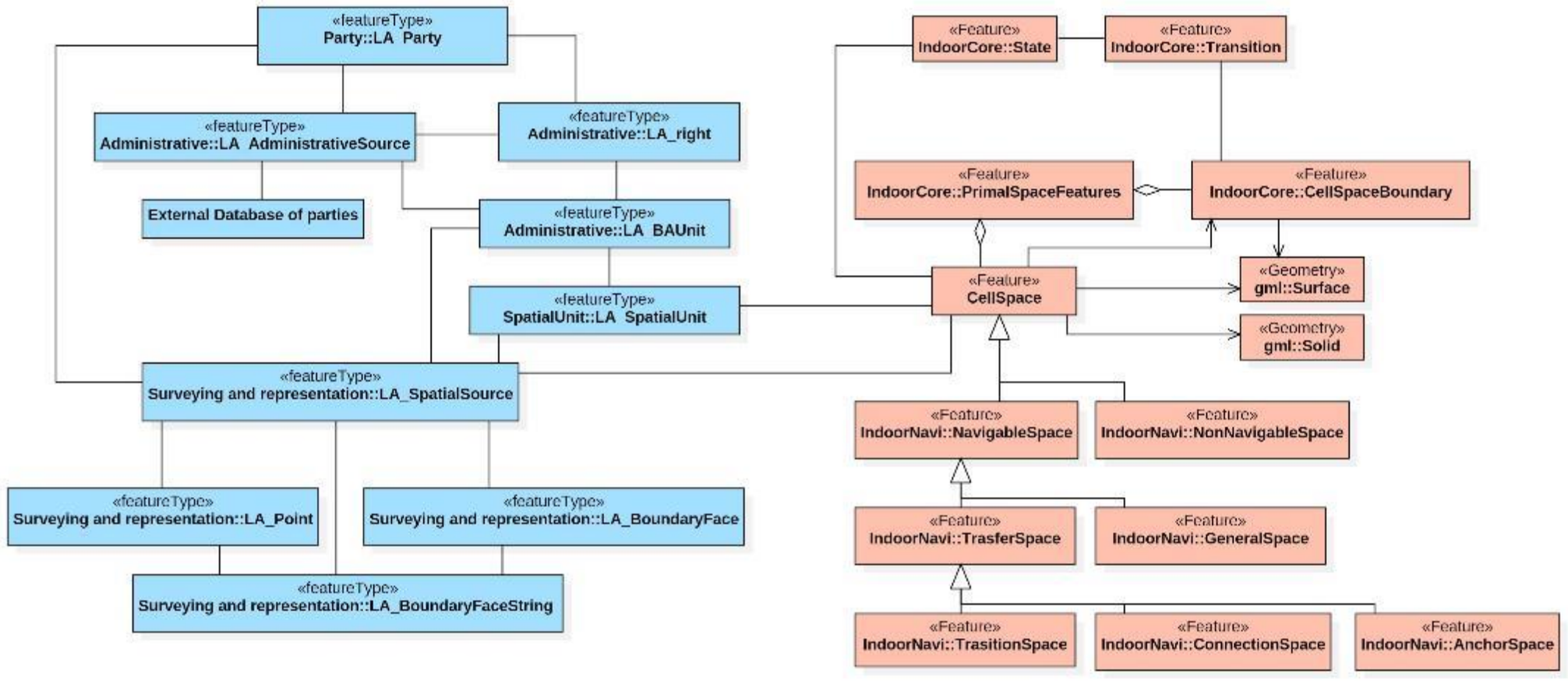
Abdullah Alattas, Peter Van Oosterom, and Sisi Zlatanova



The purpose of the integration is to determine the accessibility of the indoor spaces based on the ownership or the functional right of use for each type of group of users

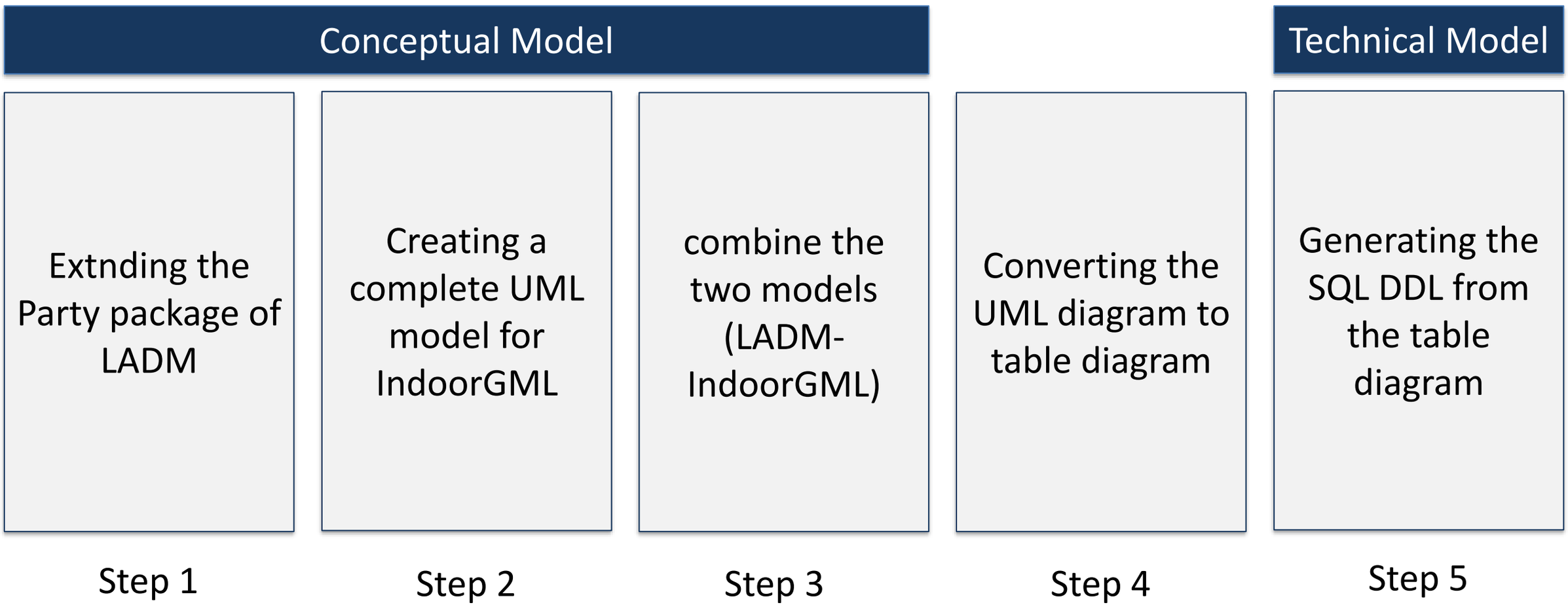
Motivation:

Is to **assess** the conversion of **LADM-IndoorGML conceptual** model to **technical** model

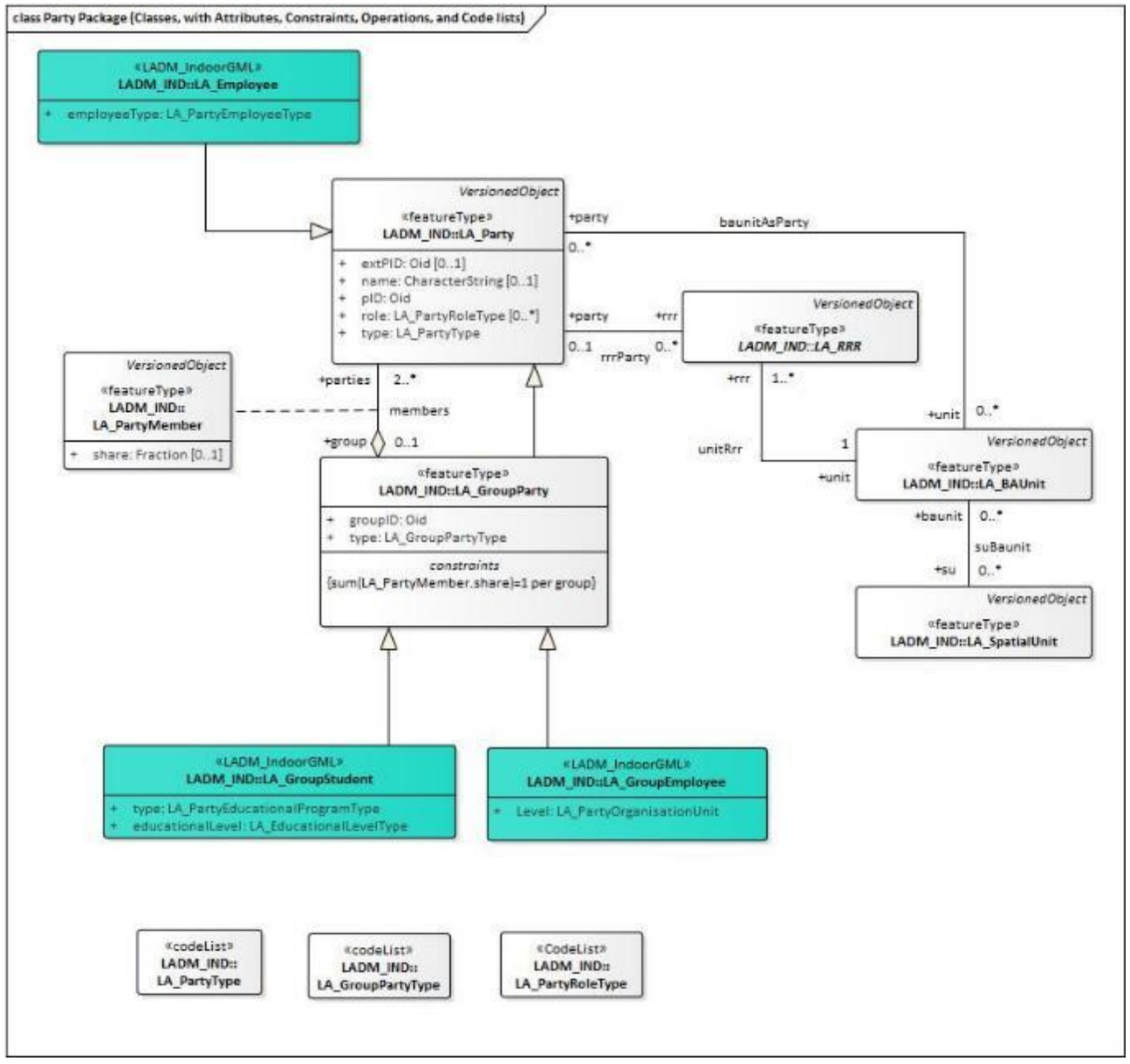


The integrated model of LADM (blue) and IndoorGML (red) (Alattas et al. 2017)

There are five steps from conceptual model to technical model:



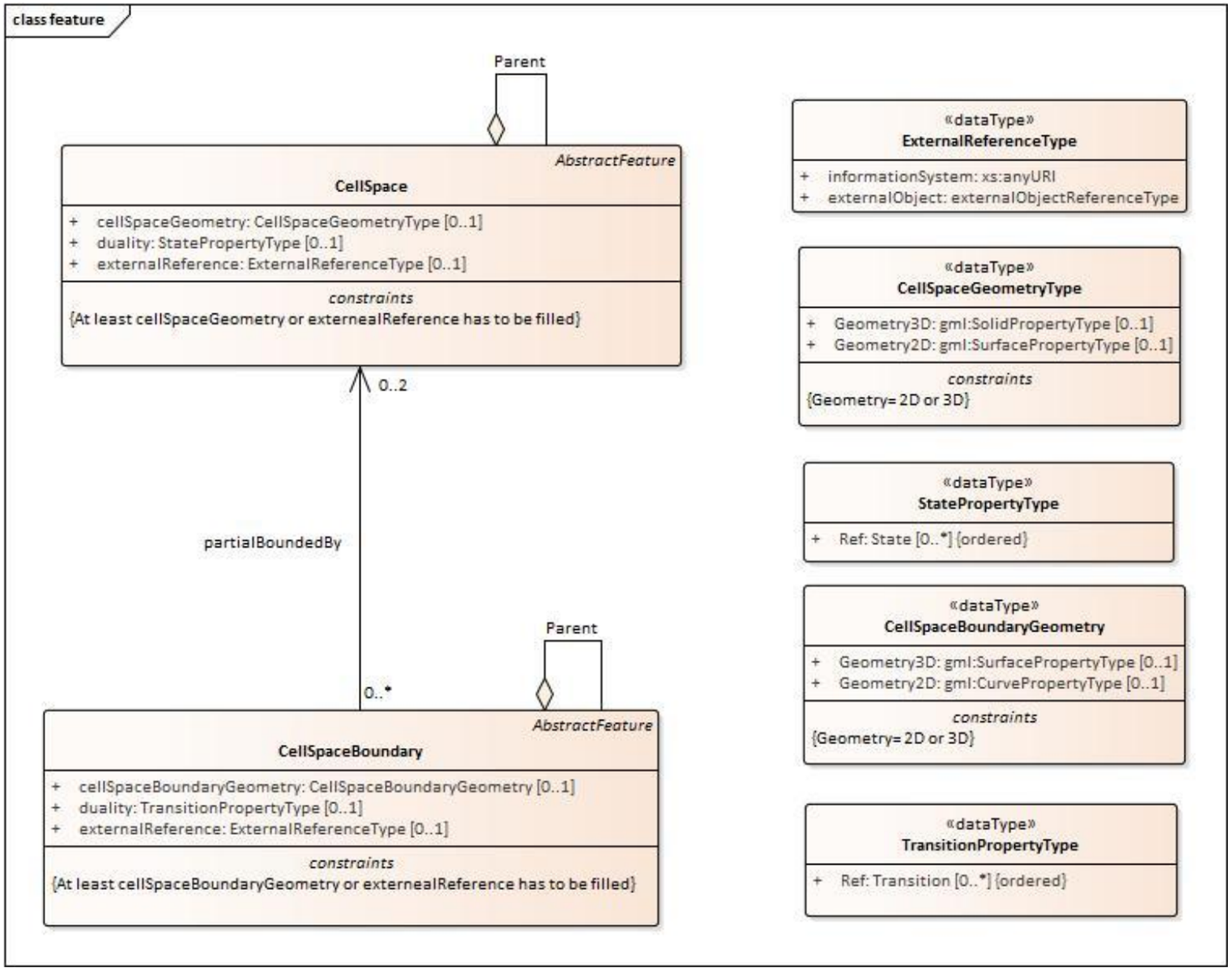
Step 1: LADM UML Model



Party class diagram contain the LADM-IndoorGML new classes (in blue the new classes)

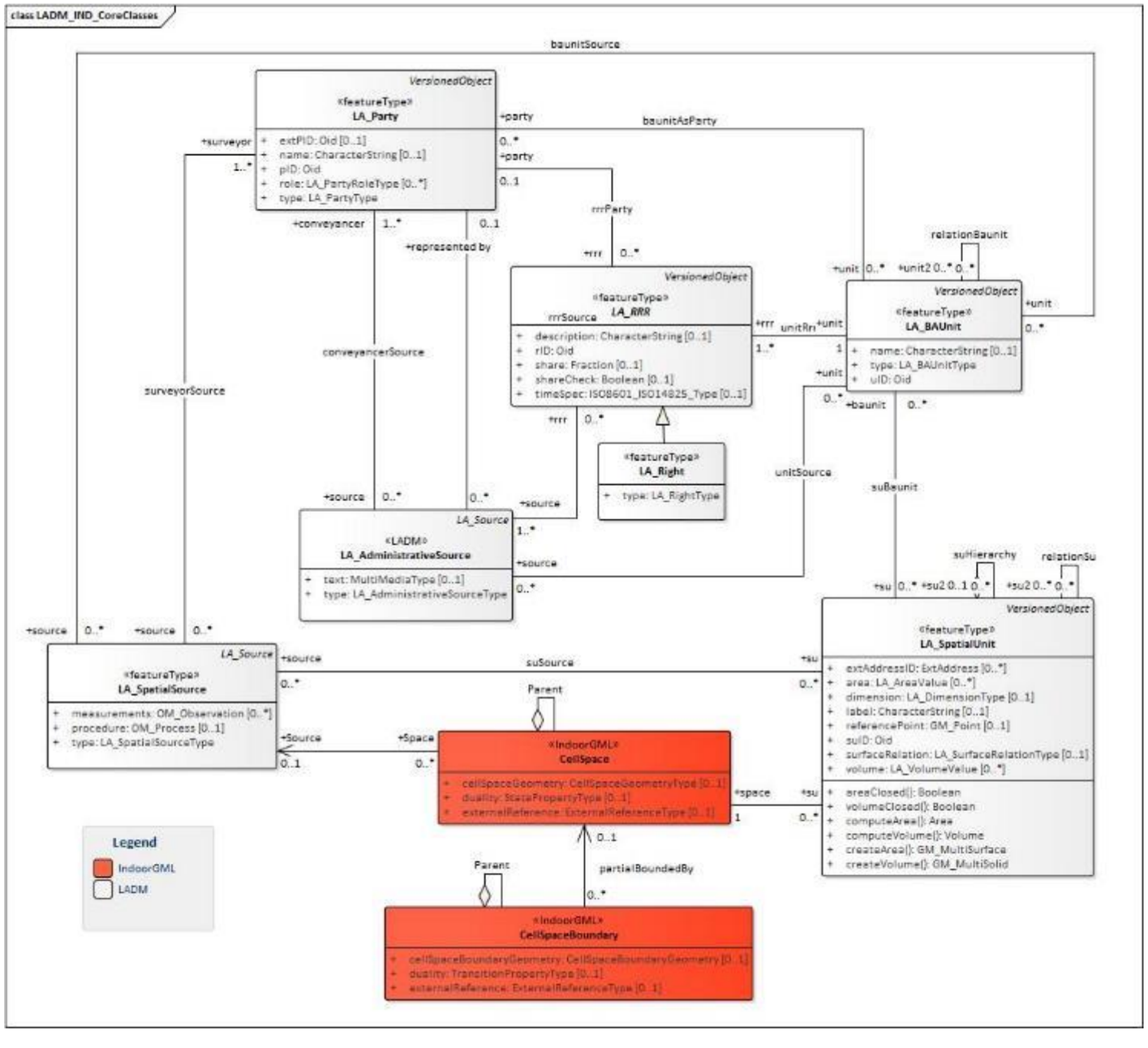
Step 2: IndoorGML UML Model

- The current version of the standard does not have a complete UML model.
- Therefore, the XML/GML schema and the underdevelopment Java classes of IndoorGML has been used to derive the attributes into UML model (class diagram) and to make the conceptual model complete.
- The code engineering tool in Enterprise Architect has been used to generate the UML classes from the Java classes (via reverse engineering)



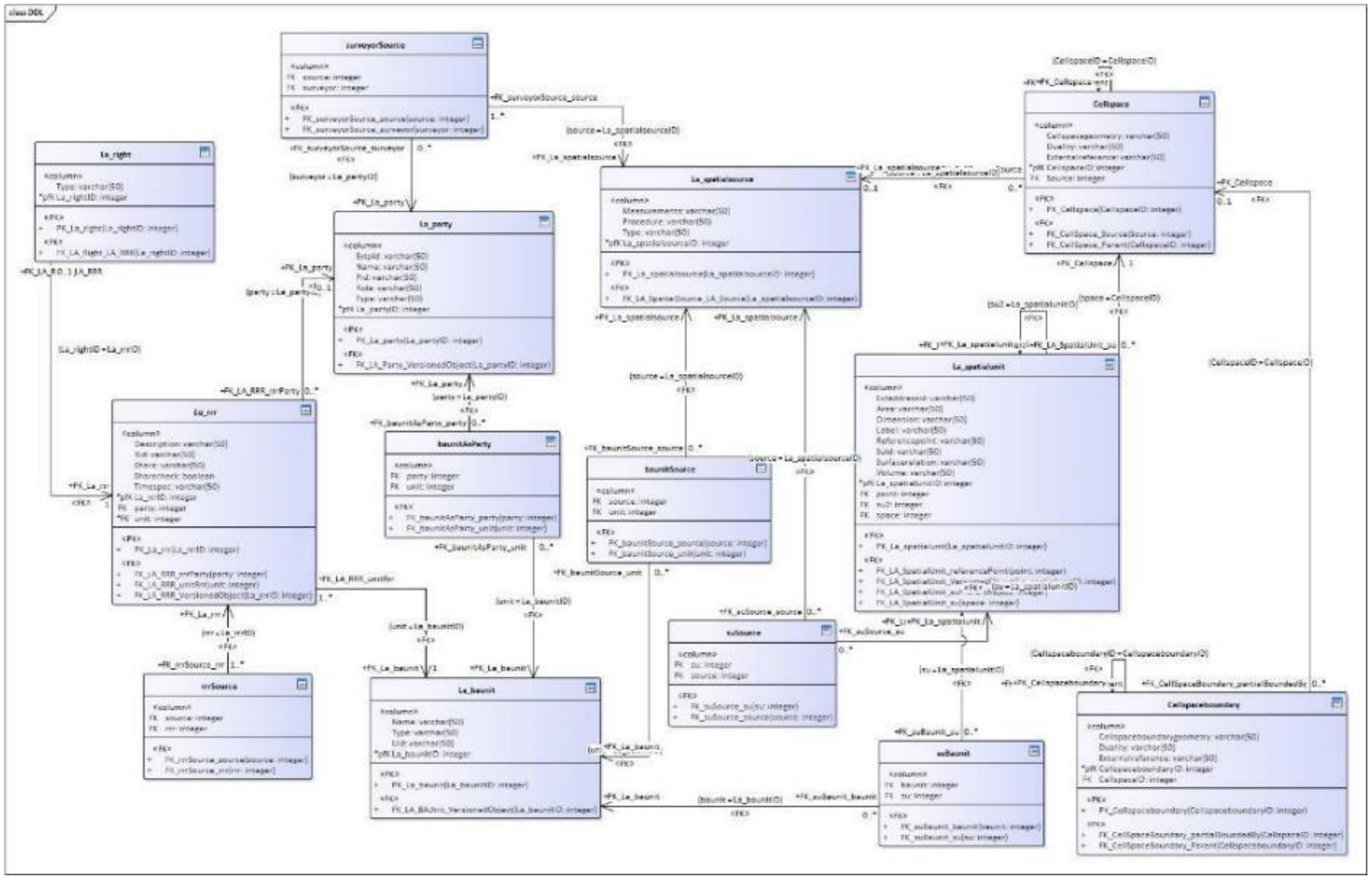
Creating attributes for CellSpace class and CellSpaceBoundary

Step 3: LADM-IndoorGML conceptual UML Model



Complete LADM-IndoorGML UML diagram

Step 4: Generating Tables Diagram from the UML Diagram



Many aspects will be analysis in the next slides

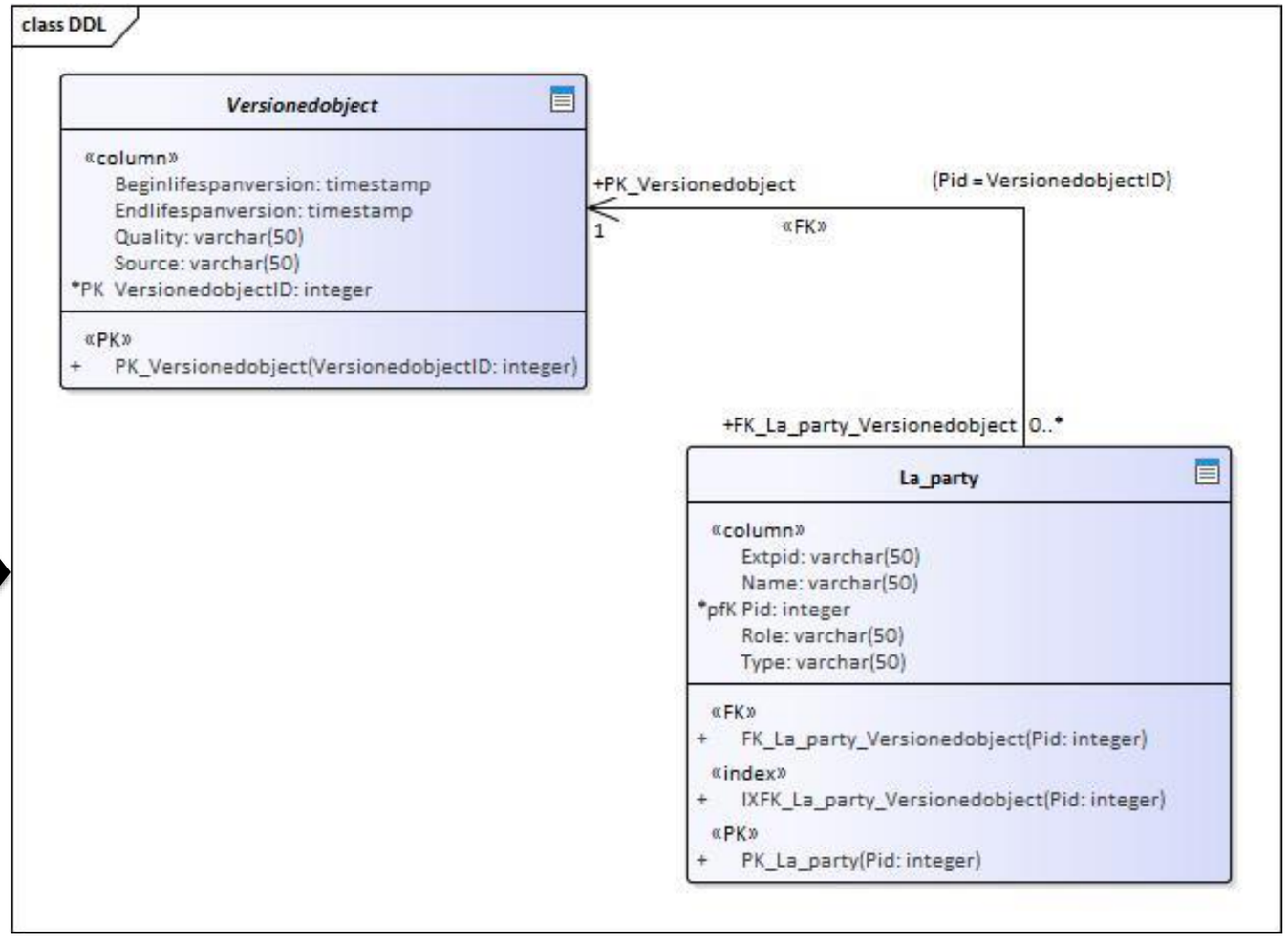
Step 4.1: Inheritance

There are three options to inherit the attributes:

- By using a **flat model** which mean all the attributes of the parents are copied to the children classes. This option makes the model to contain the same attributes over and over agian in all the children classes

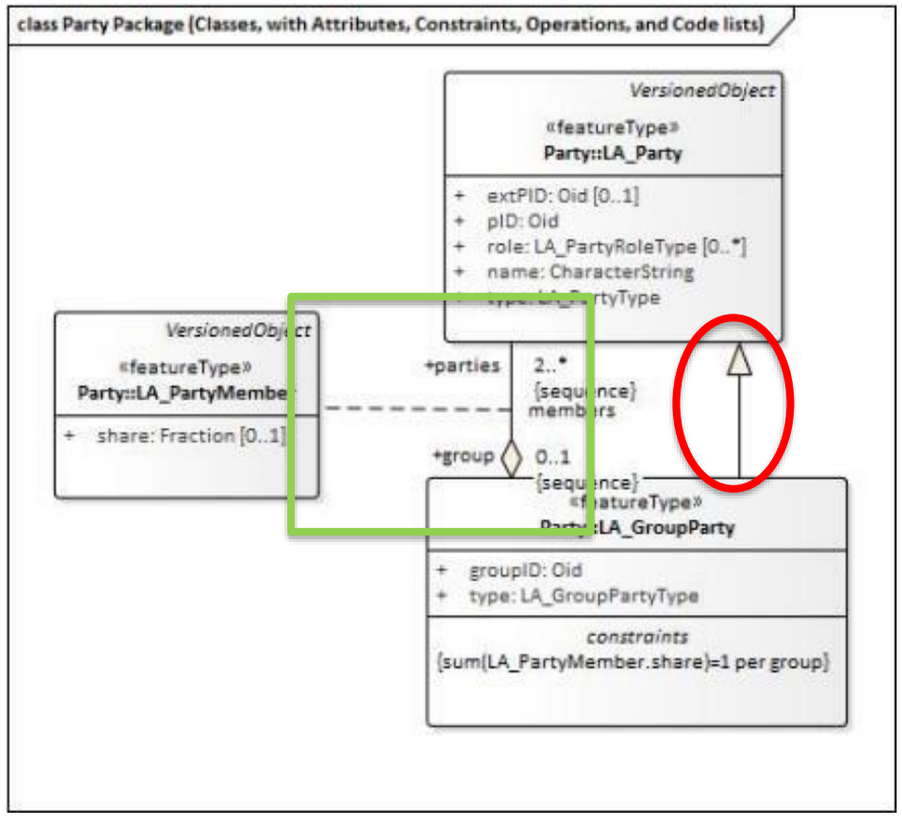
- The second option is to create a **super class** that is used as a reference class and the children classes refer to the super class by using **PK** and **FK**. This option has been used in this case to avoid the duplication of the attributes all over the model.

- The third opion is to define the **children class** as a subclass of the parent class by defining the parent class as object which will allow the children classes to store the attributes of the parent class.

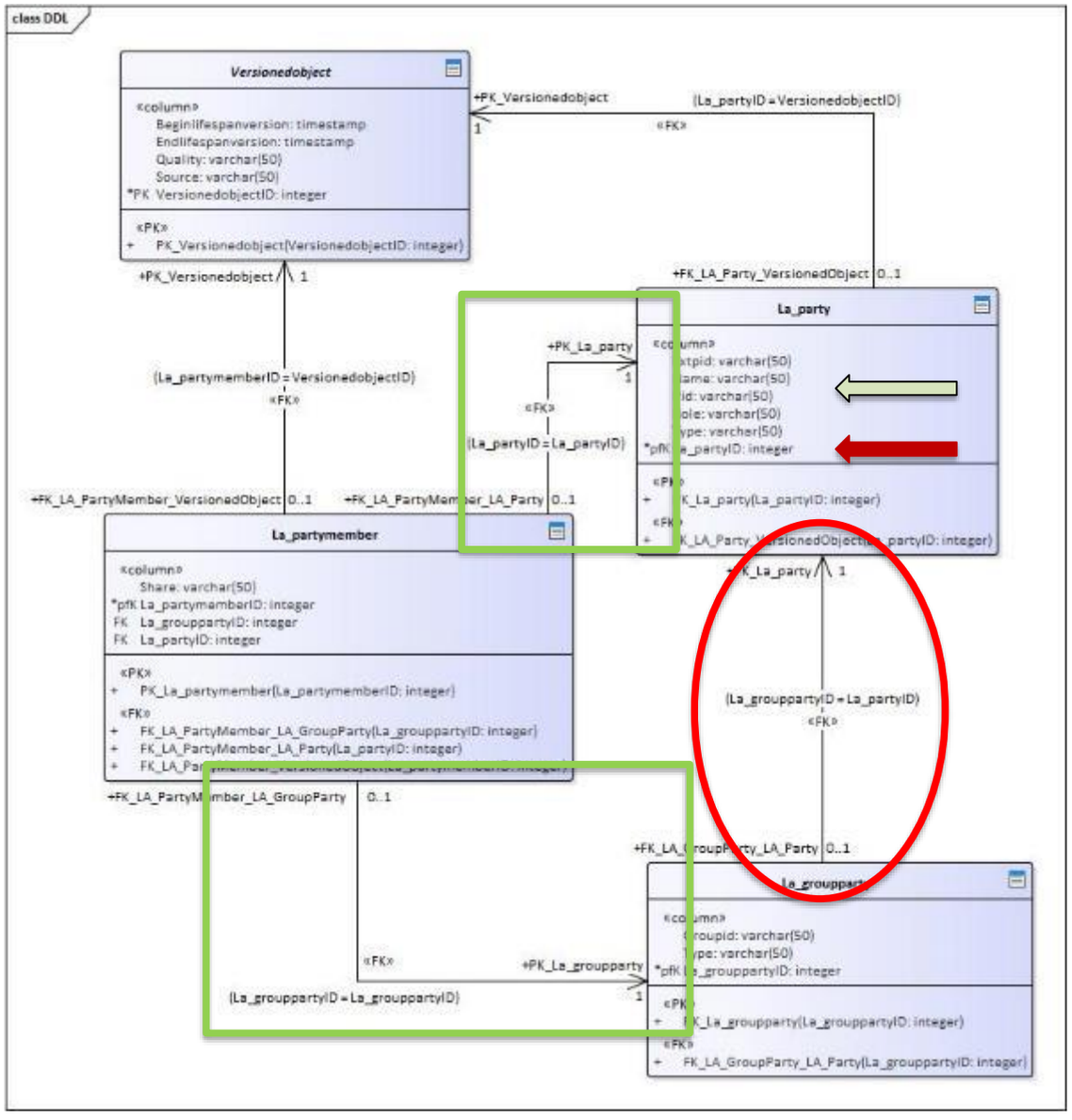


Generalization asscoiation

Step 4.2: Primary key and a Foreign key and Association Class

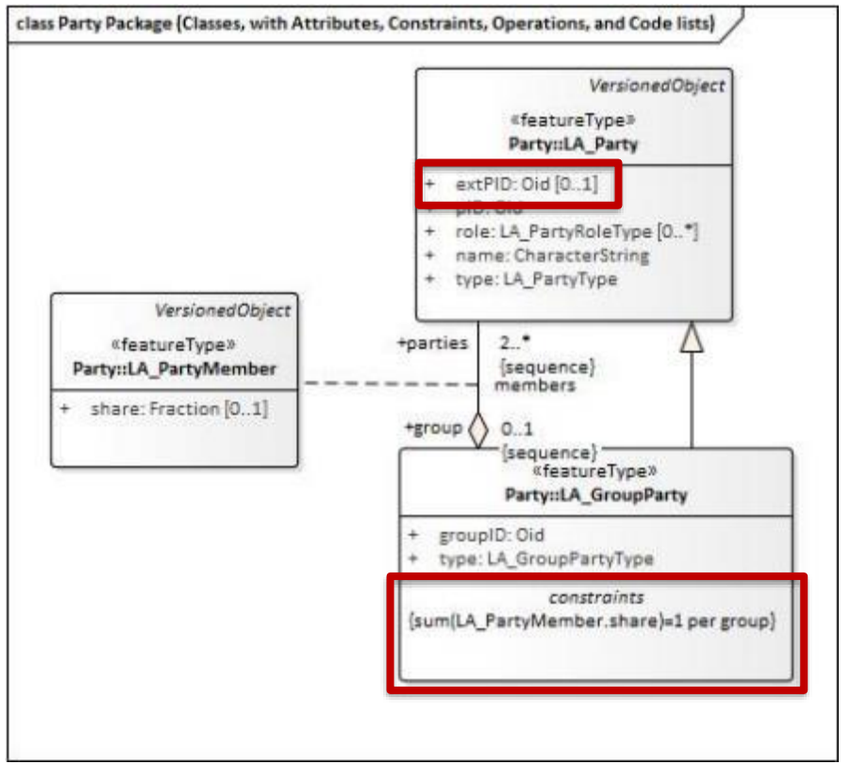


LADM Party Package shows that the party class has an ID attributes

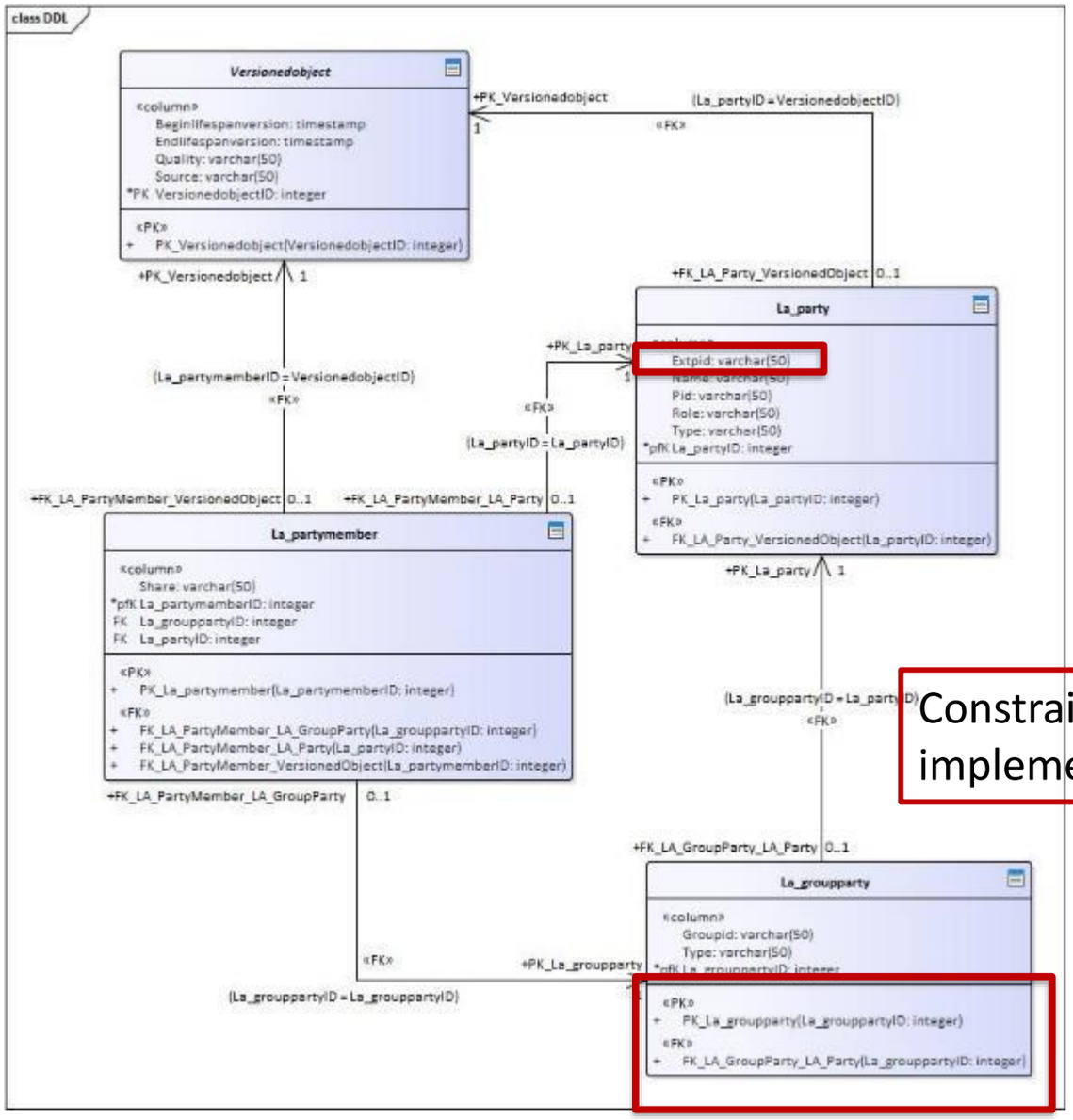


Additional Primary key has been created for each table and did not use the ID attribute that has been already defined in the class diagram

Step 4.3: Attributes Multiplicity and Constraints



The multiplicity for the association between the tables, left figure shows class diagram, right figure shows the tables diagram



Constraint are not implemented

Step 4.4: Data Type

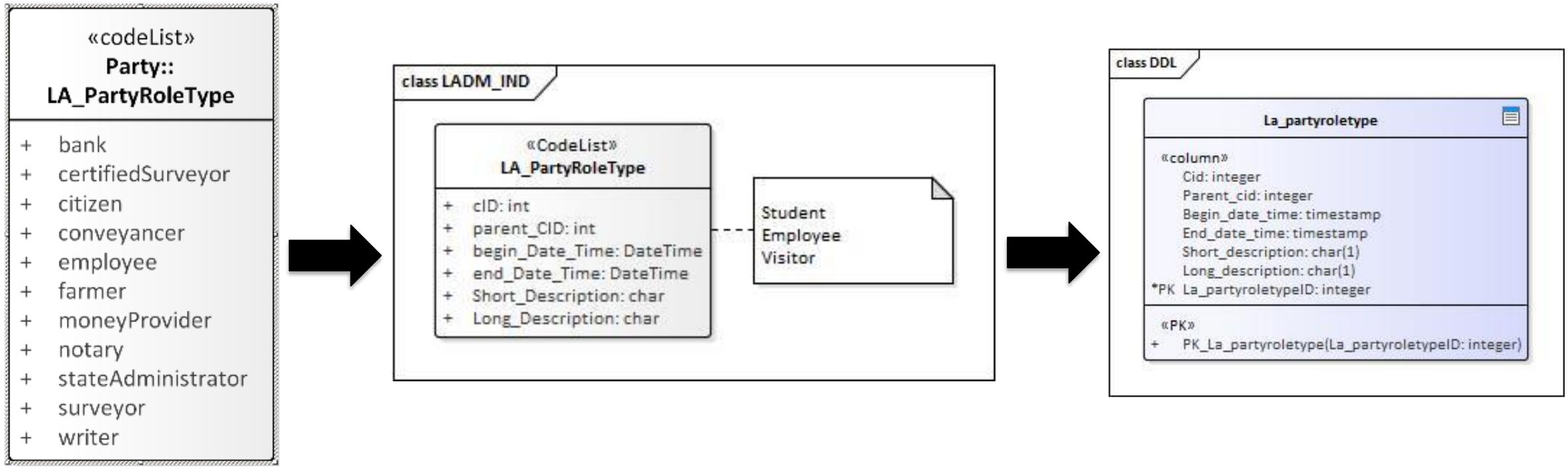
The **Oid** is a **user defined data type from ISO 19152** that used as an identifier for the objects. It consists of local identifier and nameSpace that used as an identifier for the data source. The **transformation model** does not consider the **Oid** and **Fraction** as data type and it replaces them with ”**varchar**” in the table diagram **based on the cabibility of the target plat form such as postgrass or oracle**

Step 4.5: Spatial Data Type

The **spatial classes** of the conceptual model such as **LA_SpatialUnit**, **LA_Point**, **LA_BoundaryFace**, and **LA_BoundaryFaceString** have a spatial data type for some of their attributes such as **GM_Point**, **GM_MultiSurface**, and **GM_MultiCurve**, however, the software does not realize their type and select a ”**varchar**” as a data type in table diagram **based on the cabibility of the target plat form such as postgrass or oracle**

After the transformation, the software offers a list of spatial data types that could be used such as **geometry**, **geometry collection**, **linestring**, **multilinestring**, **point**, **multipoint**, **polygon**, and **multipolygon**. If the spatial data type does not include in the list, the only way is to define the type manually in the **SQL code**.

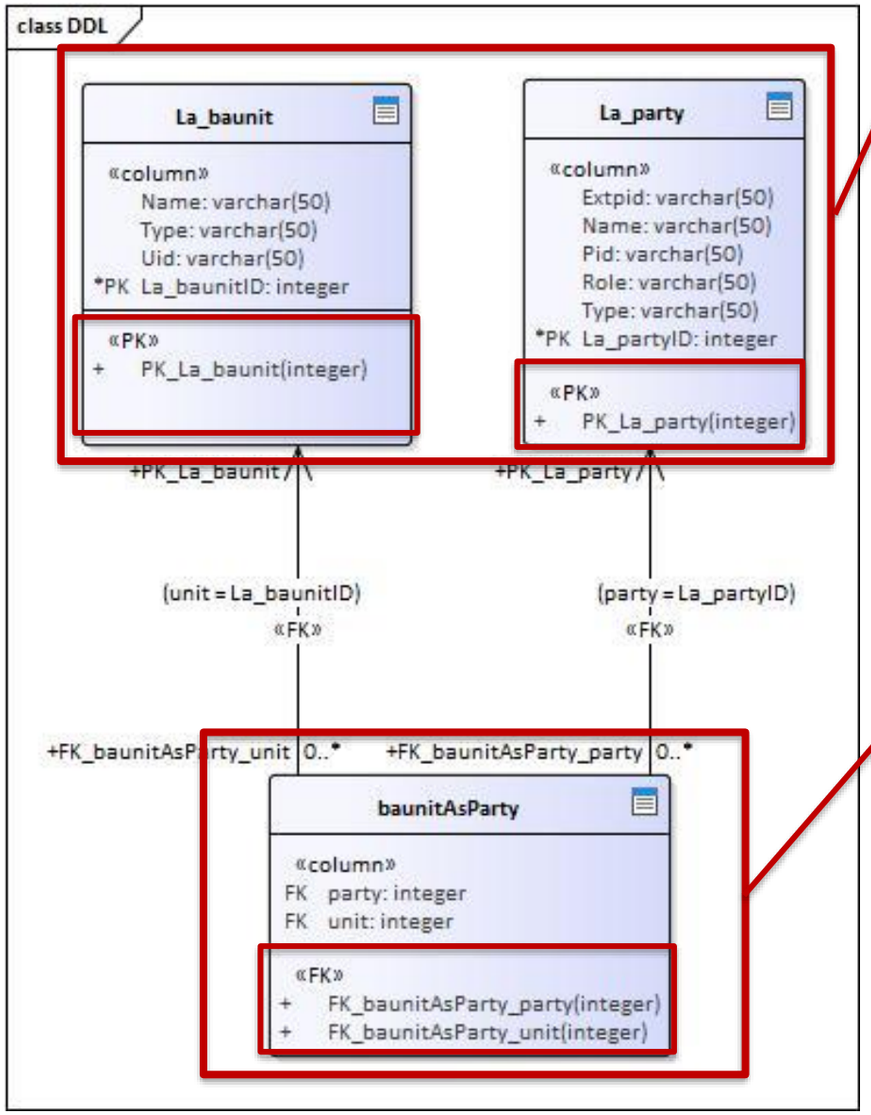
Step 4.6: Code List Classes



The code list class in conceptual model, left shows the normal class structure, right shows the implementation structure/ hierarchy and versioning for the code list

Step 4.7: Indexing

The transformation create an B-tree index in automated way and there is no additional options to generate R-tree index or index that refer to special object



The two classes have a direct association in the class diagram

A new class that has been created because of the relationship is many to many between the two classes

The index that the transformation provides in automated way

Step 5: Generating SQL from UML Table Diagram

```
CREATE TABLE La_party
(
Extpid varchar(50) NULL, -- the identifier of the party in an external registration
Name varchar(50) NULL, -- the name of the party
Pid integer NOT NULL, -- the identifier of the party
Role varchar(50) NULL, -- the role of a party in the data update and maintenance process
Type varchar(50) NULL -- the type of the party
 )
;
```

The SQL code for LA_Party table containes the notes that have been added to the conceptual model

Conclusions

- Goal: Assessing the conceptual model, however, the focus of this research has been shifted to cover the transformation issues.
- ISO TC211 used to prepare LADM-IndoorGML UML model.

Enterprise Architect software: still has many issues that need attention and manual adjustment:

- All classes from different packages have to included in the new package to ensure correct transformation to table diagram,
- A new unique attribute as PK is created even though existing unique ID has been modeled in UML,
- PK affects the associations between the tables,
- Multiplicity of the classes diagram has been changed.
- The multiplicity that relates to the attributes is not considered,
- a B-tree index is automatically created,
- Spatial index has to be defined manually in the SQL code.

Thank you