Transformational Design Exercise

This exercise is an extension of the data modelling exercise done earlier on in the module. The idea here is to further understand the principles behind transformational design and the MDA approach, and at the same time include in the modelling process aspects that are specific to spatial data (i.e., geometries).

The objectives of this exercise are:

- *Understand the extension mechanisms of UML*,
- *Create a conceptual model with geometric primitives*, and
- *Apply the Model Driven Architecture design approach*

As always the files that are needed to carry out the exercise can be found in BB.

Concepts

The text below presents a short review of important concepts, which form some of the basis of the exercise.

The Model Driven Architecture (MDA)

The MDA is a systematic design approach for the development of software systems. It provides a set of guidelines for the structuring of specifications, which are expressed as models. MDA is based on modelling separately technology-independent and technology-specific aspects of a system. These aspects are taken into consideration in separate schemas.

The most important aspect of the MDA approach is the explicit identification of Platform-Independent Models (PIMs) which can be implemented on different platforms via Platform-Specific

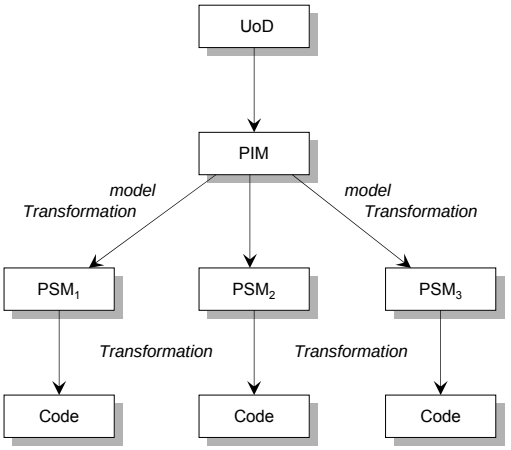

Figure 1: The Model Driven Architecture approach

Models (PSMs). An essential feature of the MDA framework is to make transformations between models (schemas) automatic. This means that a transformation tool takes a PIM and transforms it into a PSM. In a follow-up step, the transformation tool transforms the PSM to code. Figure 1 shows how a UoD is represented in a unique PIM, which in turn can be transformed into a PSM, based on a platform of choice. A second transformation derives code for the implementation of the PSM.

UML Stereotypes

Stereotypes are one of three extensibility mechanisms[1] in the Unified Modelling Language (UML). They allow designers to extend the vocabulary of UML in order to create new model elements. These new elements are derived from existing ones, but have specific properties that are suitable for a particular problem domain or specialised use. In our case, this means that we can add modelling elements to represent spatial data types, topological rules, etc. By using stereotyped elements we can make these, application specific concepts, appear as primitive building blocks of any UML tool.

Figure 2: Stereotype representation

Graphically, a stereotype is rendered as a name enclosed by guillemets (≪ ≫) and placed above the name of a modelling element. Alternatively, a stereotyped element may be indicated by a specific icon (see Figure 2). The icon image may even replace the entire UML symbol.

Since we are following the principles of the MDA, we need modelling elements to represent spatial concepts at the `platform-independent` phase of the design process. For this, we will use the stereotypes defined in the ISO standards[2]. Below, we briefly explain the added stereotypes.

The FeatureType stereotype

A feature is an abstraction for a real world phenomenon that has geometrical characteristics. ≪FeatureType≫ is a stereotype that is instantiated as a class that represent objects for which geometry properties are included in the model, e.g., parcels, lakes.

Enumerations and CodeLists

Enumerations are model elements used in class diagrams to list user-defined sets of named identifiers. These identifiers represent the values of the enumeration. These values are called enumeration literals. You can add enumerations to depict discrete sets of values. Models can include user-defined enumerations such as one that defines the days of the week. For example, an enumeration called `Day` would have enumeration literals `Monday`, `Tuesday`, `Wednesday`, and so on.

ISO defines a special type of enumeration, the so-called ≪CodeList≫. A code list is similar to an enumeration, in that one of a number of values is possible, but differs in intent, in that a

[1] UML extensibility has been study in the first bunch of exercises.
[2] See the ISO 19101, 19107 and 19109 documents for further reference.

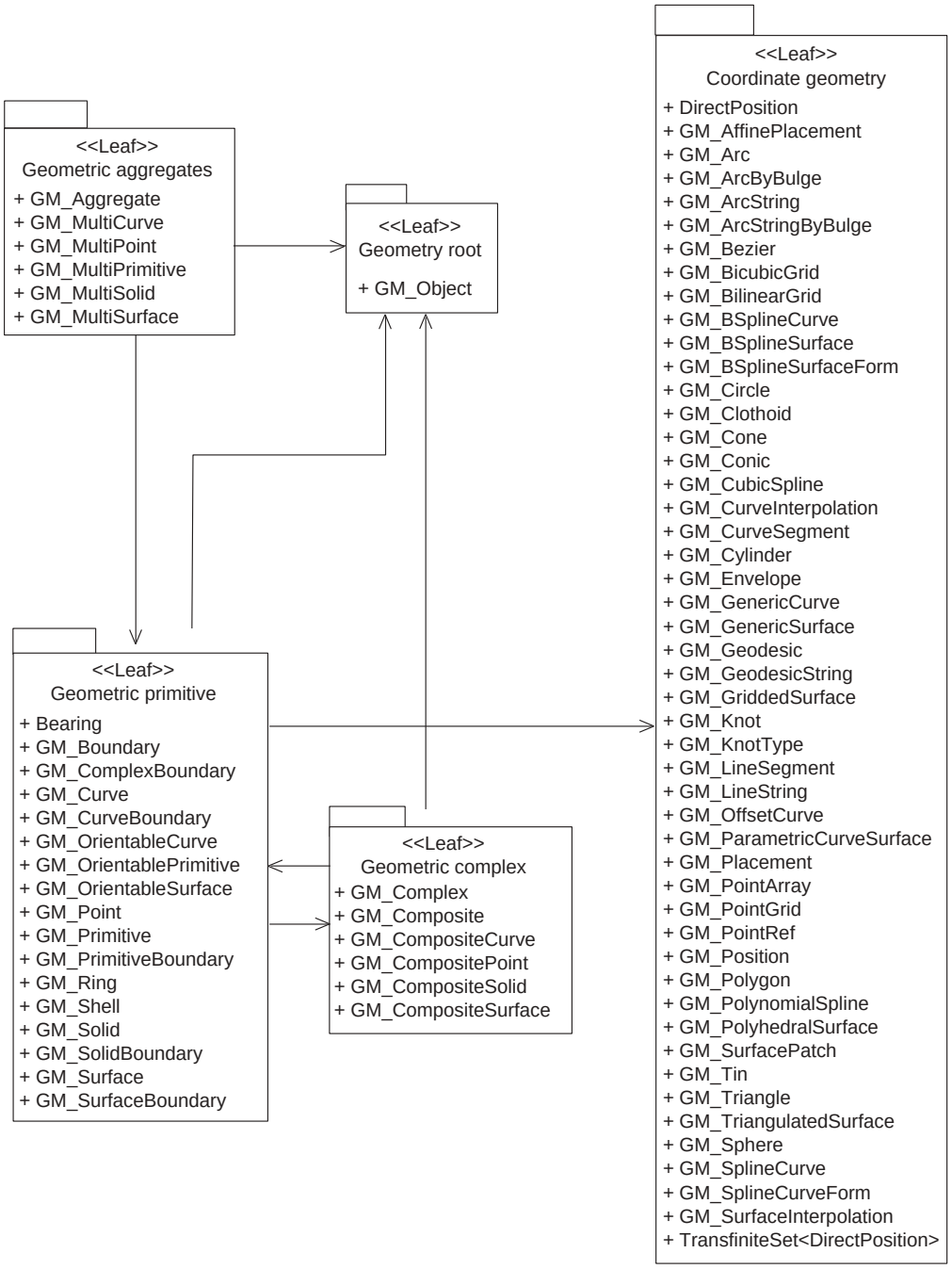

Figure 3: The Geometry Package

code list may be expanded over time. Most code lists are stored as numeric values, but some implementations use character strings. Please note that this is only an ISO subtlety that you may decide to disregard in your design process, but since in this exercise we want to be compliant, we follow ISO to the letter.

The Geometry Package

Several elements are required to completely cover the modelling aspects of spatial applications. ISO covers almost all of the issues, however, here we look at a very limited subset. ISO's modelling elements are organised into packages. The elements we need in this exercise are defined in the `Geometry Package`. The geometry package has several internal packages that separate primitive geometric objects, aggregates and complexes, which have a more elaborate internal structure than simple aggregates (see Figure 3). At the root of the geometry package sits the GM_Object which represents a combination of a geometry and a reference system.

The GM_Primitive is the abstract root class of the geometric primitives (see Figure 3). A geometric primitive (GM_Primitive) is a geometric object that is not decomposed further into other geometric primitives in the system. All other primitives imply some form of composition of geometric primitives to form more elaborated geometries.

In this exercise we will use some of these primitives as data types for our spatial attributes. The extension of Enterprise Architect available for you includes the following types: GM_Point, GM_LineString, GM_Polygon, GM_MultiPoint, GM_MultiCurve, GM_MultiSurface, and GM_Complex.

Transformational Design (Tutorial)

What we will do here first is to develop a PIM model to describe our Universe of Discourse. To be correct in terms of the platform independence of the model according to the MDA approach, we will use ISO standard data types at this stage. Secondly we will transform the PIM model into a PSM model using as the platform of choice PostgreSQL/PostGIS. Finally, the resulting PSM model will be transformed into actual code for the creation of the physical structure of the database.

Let us get to work. Start by creating a work directory for the exercise. Place inside the newly created directory the files provided in BB. Make sure to decompress the ".zip" file. These files contain the code that correspond to a spatial profile, an extension to UML that includes spatial modelling elements, and they also contain data required to configure individual EA projects, and some PIM to PSM transformation rules. These Files are used during the transformation process and as a consequence they need to be in the same directory as your EA project files.

EA configuration

Now we need to import the profile into EA. To do that start EA and from the `Settings` menu select the `MDG Technologies` option. A window appears with the list of all the current technologies available in EA. Here you can enable or disable any of the available technologies. To include our profile we only need to register the path where our files are. To do this click on the `Advanced` button, click `Add` and then `Add path` and navigate to the location were you put the profile files to add the path to EA. Click `OK` twice to accept the registration.

At this point you need to re-start EA for the new configuration to work. Note that the term "`MDG Technology`" is purely an EA feature. It is one of the EA ways to handle UML profiles and has

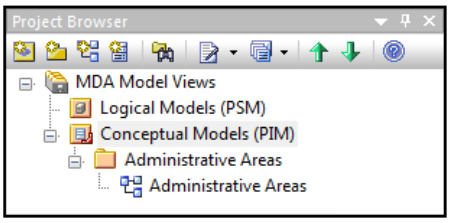

Figure 4: The Project Browser

nothing to do with the modelling process itself. The actual process of creating a UML profile falls outside the scope of this exercise. Once you restart EA you can see that a new technology is now available, the so-called `ISO-OGC Spatial`. You can see it by opening the '*`MDG Technology`*' dialog box again. The new profile will be available for you in any successive sessions of EA.

The project file

We can now start with the definition of a project, customised to realise transformations that include spatial modelling elements.

Create a new project, give it a name and make sure it is located (saved) in your work directory. The same directory as the spatial profile folder. Do not select any of the predefined models, Just close the `Model Wizard Window`. In the `Project Browser` select the 'Model' root, then press `F2` and rename it to '*`MDA Model Views`*'. Right-click and select `Add` and then the `Add View` option. In the dialog box type '*`Conceptual Models (PIM)`*' as the name of the view and make sure the `Class View` option is selected. We use this option because we are creating a conceptual model, just like in the first exercise. Repeat the view creation process to create a '*`Logical Models (PSM)`*' view, but this time select the `Deployment` option as the view type. If necessary expand the model views by clicking on the `+` sign. Now we need to define which of this views is going to contain the conceptual elements. This is achieved by defining which view is the namespace root. To do this, right-click on the '*`Conceptual Models (PIM)`*' view, navigate to `Code Engineering` and select `Set as Namespace Root`. The namespace symbol appears at the lower-right corner of the icon of the selected view.

At this point we need to specify the data types we will be using and the mappings between platforms (note that these mappings are specific for every single EA project[3]). To do that, from the `Project` menu navigate to `Model Import/Export` and then select `Import Reference Data`. In the dialog box that appears, use the `Select` button to select the file called '*`pgmda_DataTypes.XML`*' from the spatial profile directory. From the list that appears in the box, select '*`Spatial Data Types`*' and press `Import`. We now have access to spatial data types in our project. We still need to define the new data types as the default types for our project. To do this, from the `Code Engineering` toolbox select `PIM` for the code and `PostgreSQL` for the database platform. If the `Code Engineering` toolbar is not visible, then enable it by right clicking on the EA `toolbar` and in the menu that appears select `Toolbars` and then `Code Engineering`.

We continue by creating a package to contain the modelling elements of our exercise. In the `Project Browser`, right-click on the '*`Platform Independent Model`*' view, select `Add` and then `Add Package`. Type a name, '*`Administrative Areas`*', make sure that the `Automatically add new diagram` checkbox is enabled and then click `OK` twice. Your project should look like Figure 4 at the moment. To start creating elements, open the '*`Administrative Areas`*' diagram. We need to select the ISO-OGC toolbox. To the left of your EA window is the `Toolbox`, select `More Tools`

[3]You will need to repeat this step in any new EA project if you want to make use of these data type mappings.

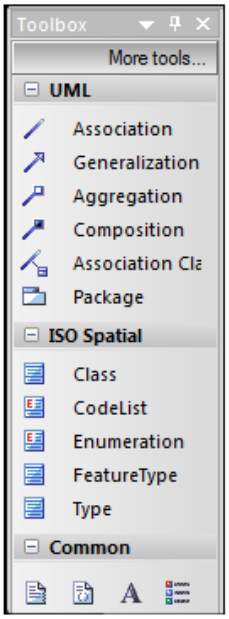

Figure 5: The ISO-OGC profile toolbox

and then '*ISO-OGC*'. Your toolbox should now contain the stereotypes required for the exercise (see Figure 5).

To carry on with the transformation we need a model. Your next task is therefore to recreate the model that appears in Figure 6. Go ahead and produce your diagram, select the appropriate modelling elements from the ISO-OGC toolbox and place them in the diagram canvas. The stereotype names in your diagram might look slightly different to those on Figure 6. This is because for your exercise, Enterprise Architect has been configured to display stereotype inheritance. Make sure not to use spaces in any of the elements' names. Notice that literals in enumerations and code lists are entered as if they are attributes but without an specific data type, instead, make sure that the `Is literal` checkbox is selected for each of the literals. The list below presents a summary of the requirements behind this conceptual model. The model reflects the needs of a government organisation that manages administrative data as follows:

- The country is divided into administrative areas of different types structured in a hierarchy. An administrative area is compose of other administrative areas. Administrative area may not be contiguous areas.
- Taxation is managed based on lots and building, both of which are located within a municipality. Taxes on buildings are calculated based on the buildings use.
- All data on building accessibility via the road network is stored and used for routing and personal car navigation.
- All official names are defined by an authority which is known to the public by its business credentials.

When defining attribute types, do not type or enter values in the 'Type' box of the attributes dialog box. Instead select the appropriate type from the drop list. If the type that you are looking for does not appear in the list yet, e.g. '*RoadCategoryType*', then you need to create the type

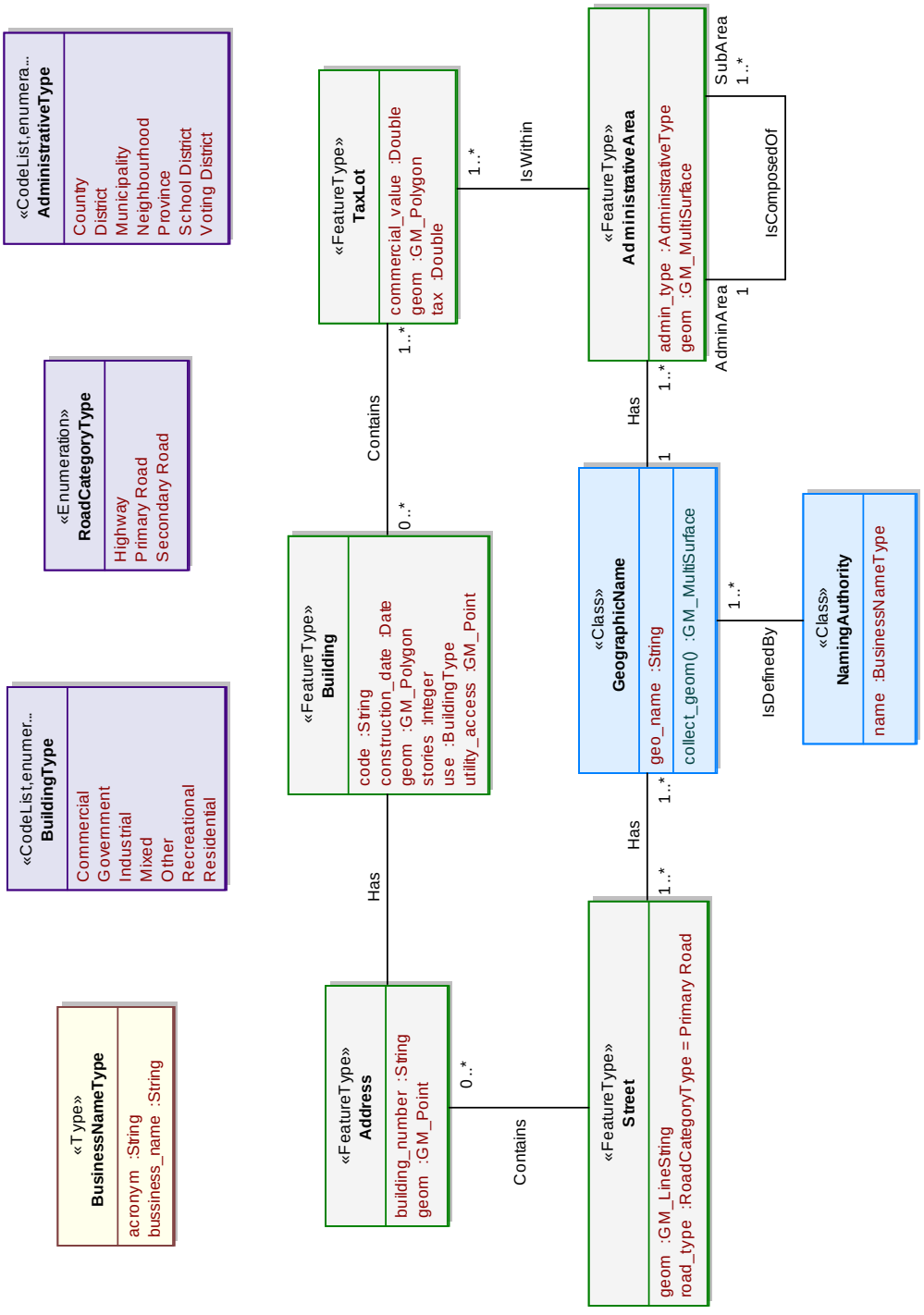

Figure 6: The administrative areas conceptual data model (PIM)

first, by selecting the correct element from the toolbox. Another reason for the data types that you need not appearing in the list is that you did not select the proper platforms in the `Code Generation` toolbar.

As you start working with the exercise yo may notice that your conceptual data types now offer '*GM_---point, multipoint, linestring, multicurve, polygon*' and '*multisurface---*', which are ISO datatypes. During the lectures we discussed OGC data types and we talked about '*multilinestring*' and '*multipolygon*', instead of '*multicurve*' and '*multisurface*'. This might not look quite consistent to you but this is what it is. Ant it is for now beyond our control. If I were in charge I would call the same thing with only one name. But there you go, I not in charge so, my apologies for the lack of consistency. The transformation template will take care of making the proper mappings between these types, you only need to be sure of picking the right one at the conceptual level.

Do try to understand the model and if necessary make modifications to it. Pay attention to the classes with geometric attributes. Explain the mechanism used to model administrative areas, such as state, province, etc. Your understanding of the PIM model is required so that you can make sense of the transformations that will be executed next.

Transformation from conceptual to logical model (PIM to PSM)

The transformation process will generate a set of PSM objects based on a given set of PIM objects. We will store the resulting objects in the '*Logical Models (PSM)*' view, but we need a package to contain them. Go ahead and create a package (with no diagram) in inside the '*PSM*' view and call it '*Administrative Areas*'. If by accident you created a diagram inside the new package, remove it before proceeding to the next step.

EA comes with its own transformation libraries, but we will not use those because they will not recognise our profile and our spatial data types. Instead we will use our own transformation libraryMost of the code used in this and the succeeding transformation steps was develop by Jan van Bennekom, a graduate from the GIMA MSc. course. If necessary save your '*PIM*' diagram before proceeding to avoid unexpected behaviour during the transformation.

To start the transformation, open the `Extensions` menu, navigate to `Transformation Tool` and then select the `Transformations` option. In the dialog box that appears, you can see the source and target packages for the transformation and a list of transformation task at the bottom left of the window.

All you really need to do is to select the PIM package that you want to transform, and the destination package where the resulting PSM elements should be placed. In our case:

- The PIM Package is `Administrative Areas`
- The output PSM Package located inside the `Logical Models (PSM)` view is also `Administrative Areas`

Select the appropriate source and destination packages for the transformation (be careful here!!!). Next select the option `1. Transform PIM->PSM` and then click on `Run Task`. The transformation will start and messages will be shown in the progress window. Once the transformation is finished, close the dialog box and check the '*PSM*' view to see the results.

If things went according to plan (or indeed according to our transformation template), then you should now have an '*Administrative Areas*' diagram and a number of '*PSM*' objects inside the

'*Administrative Areas*' package in the '*Logical Models (PSM)*' view. Expand the view and open the newly created diagram. Check the results to determine what did actually happen.

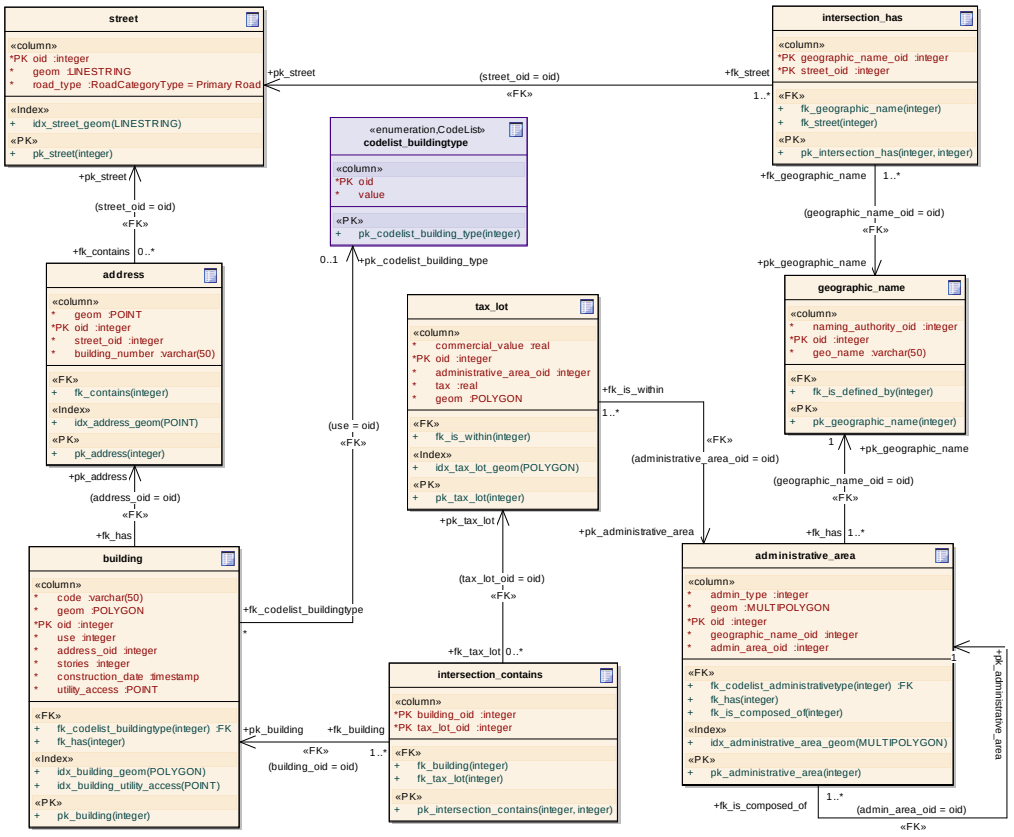

Figure 7: Part of the Administrative PSM model after the first transformation

Check the geometric attributes. See what happen to the enumerations and code lists. Try to find out if there are issues that need to be addressed.

Transformation from logical to physical model (PSM to code)

The only step that remains now is to produce the code that corresponds to the newly created '*PSM*' model. To do this, we use again the `Transformations` extension. This time we select option `2. Generate DDL/SQL Scripts PSM->PSM` from the menu at the bottom left. Once option 2 is chosen you get a new set of choices to make. First select the '*PSM*' package that you want to transform. Then we need to decide which reference system will be use for our geometries. Here we are assuming that we have a homogeneous design and all our geometries fall in the same reference system. We will assume that in this exercise is that the resulting database will contain data from The Netherlands, and therefore all the geometries will carry the SRID of the Dutch Reference System (EPSG:28992). Go ahead fill in the number in the corresponding box and then click on `Run Task` to carry out the second transformation. The result of this transformation step is a bunch of SQL files that should now be located in your working directory. Check them out using a text editor. These 'SQL' scripts may now use to create your own Spatial database in a PostgreSQL server. Some of the resulting 'SQL' files might be empty, for example the '*createviews.sql*' will be empty because we did not define any view in the model[4].

[4]If you were to have access to a server where you have sufficient privileges to administer a database system, you may use the resulting .sql files to populate your spatial database (create tables and relationships between,

Database design case study (Exercise)

For the second part of the exercise (the real exercise) you are asked to develop a conceptual schema, using UML class diagrams, that is a good reflection of the information content of the 'Company' database. The requirements document can be found in the sections ahead.

By the end of this exercise you should submit via Blackboard a compressed file (*.zip) containing the '*.eap' file of your project and one '*.sql' file containing the code (organised) that came as a result of the model transformation process. You should analyse the various pieces of code and define the correct sequence.

The tasks to perform

What you have to do in this practical is to derive a conceptual database schema. Below you can find the steps that you can follow to derive such a schema. Follow the steps carefully but do not take the sequence of steps too stringently. It is implicitly assumed that all the heuristics and subtle issues of conceptual data modelling, as they were discussed in class, will be taken into account in your exercise. Do verify with the slides on this. All of the steps listed below must be carried out:

1. Identify object populations and turn them into classes. Do not yet identify attributes or operations. Document the classes, and make sure you that there exist enough objects for each class in your UoD.

2. Consider whether you have any generalisation relations between classes. Do the subset test on their respective populations. Also pose the fundamental question of subpopulation disjointness and coverage of the super-population.

3. Identify the associations between classes, and provide their names, or roles, and multiplicities. Pose the question whether an association is an aggregation or perhaps even a composition, using the heuristics discussed in class.

4. Assign attributes to classes (names, datatypes), and consider whether they can be null-valued or not. If you do not know, allow them to be null-valued for the time being. If you want to assign an attribute to an association, by UML rules, you need to turn the association into an association class. Pose yourself the question with each attribute whether it is a derived or stored attribute.

5. Revisit every class, and determine any object constraint that needs to be included in the model. Find out how these can be administered in a UML class model, using EA. Do the same for all associations and association classes.

6. Revisit all you classes and add spatial attributes where you believe they are called for. Carefully consider whether you need a simple type 'LineString' or a multi type 'MultiLineString', for instance, and understand that your choice should be based on deep understanding of the phenomena that require representation.

7. In the end, perform the transformations discussed above, and study the PSM and Code that is derived from this.

constraints, indexes and types) in PostgreSQL+PostGIS database. Please note that the order in which the code is executed matters!

Requirements for the REAL ESTATE BROKER database system

The REAL ESTATE BROKER database would be used to keep track of data about companies involved in the property market around the world. These companies form part of a large multinational. The description of the actual UoD is as follows:

- Companies are organised into departments. Each department has a name, a unique number and a manager. The multinational keeps track of the date when an employee starts managing a department. A department operates in a number of offices, which may be located in different locations/places (buildings). These locations have identifying addresses and for each location the staff capacity (number of employees that can work at that location) is known. The company's headquarter is located in one of those locations.
- A department is responsible for a number of marketing projects. Each project has a name, a unique number and a scope (type of marketing e.g. residential, commercial). Each project is managed at one of the department's offices, typically that one closest to or inside the area covered by the project. Project areas are mutually exclusive. Project areas are surveyed (commercial surveys for investment[5]) based on subareas and not two contiguous areas should be surveyed at the same time, which requires an appropriate survey plan. These survey result in the identification of the best investment opportunities for each project area.
- In each of the surveyed areas, properties suitable for lease or purchase are identified, and their cadastral status recorded (this includes the geometry of the property, commercial price, owner etc.). One property can be assigned to different types of investment opportunities.
- Important data with respect to employees include: name, age, social/fiscal number, address, salary, gender and date of birth. An employee is assigned to one department but may work in multiple projects, which are not necessarily controlled by the same department. The number of hours per week that an employee works in each project should be recorded. It should be possible to determine the direct supervisor of each employee.
- There are four types of employees: assistant, technician, engineer and researcher. For assistants we keep track of their contract type which can be hourly, part time or full time. For technicians we record their specialties or skills. Engineers have different ranks therefore we record the name of their position. For researches we keep record of their degree or degrees they obtained in the university.
- Employees may not live in the same place/city where they work, therefore we need to keep track of their current address and permanent address.
- For insurance purposes it is necessary to keep track of the dependants of each employee. For every dependant's the first name, gender, date of birth and relationship are kept.
- Addresses should be used as a primary entry point for extracting information from the database. We should be able to identify, for example, the city or country where a project is, or the number of locations where only a few projects are running.

Gathering information on a database system

We are not doing this in our exercise but if we were to find out more details about the requirements of a database system we may use a selection of fact-finding techniques including interviews and observing the business in operation. Examples of the types of questions that we may ask include:

[5]Investment here refers to property lease or purchase

- What is the purpose of your Company?
- Why do you feel that you need a database?
- How do you know that a database will solve your problem?
- What type of data do you need to hold on objects of type X?
- What sorts of things do you do with the data on X, how do you report on it?"
- What transactions run frequently on the database?
- What transactions are critical to the operation of the organisation?
- When do the critical transactions run?
- When are the low, normal, and high workload periods for the critical transactions?
- What type of security do you want for the database application?
- Is there any highly sensitive data that should only be accessed by certain members of staff?"
- What historical data do you want to hold?
- What are the networking and shared access requirements for the database system?
- What type of protection from failures or data loss do we want for your database application?