Software and Systems Modeling (2022) 21:623–666
https://doi.org/10.1007/s10270-021-00901-1

THEME SECTION PAPER

Multilevel modeling of geographic information systems based on international standards

Suilen H. Alvarado[1] · Alejandro Cortiñas[1] · Miguel R. Luaces[1] · Oscar Pedreira[1] · Angeles S. Places[1]

Received: 21 June 2020 / Revised: 3 May 2021 / Accepted: 7 June 2021 / Published online: 2 July 2021

Abstract
Even though different applications based on Geographic Information Systems (GIS) provide different features and functions, they all share a set of common concepts (e.g., spatial data types, operations, services), a common architecture, and a common set of technologies. Furthermore, common structures appear repeatedly in different GIS, although they have to be specialized in specific application domains. Multilevel modeling is an approach to model-driven engineering (MDE) in which the number of metamodel levels is not fixed. This approach aims at solving the limitations of a two-level metamodeling approach, which forces the designer to include all the metamodel elements at the same level. In this paper, we address the application of multilevel modeling to the domain of GIS, and we evaluate its potential benefits. Although we do not present a complete set of models, we present four representative scenarios supported by example models. One of them is based on the standards defined by ISO TC/211 and the Open Geospatial Consortium. The other three are based on the EU INSPIRE Directive (territory administration, spatial networks, and facility management). These scenarios show that multilevel modeling can provide more benefits to GIS modeling than a two-level metamodeling approach.

Keywords Model-driven engineering · Multilevel software modeling · Geographic information systems

1 Introduction

Geographic Information Systems (GIS) support the processes of capturing, managing, visualizing, and analyzing data with a geospatial component [1]. GIS are used in many application domains, such as the management of transportation networks, logistics, supply infrastructures, or territory administration, among many others. Although in their beginnings GIS were used mainly by public administrations and engineering companies, the evolution of technologies for GIS development and the appearance of cheap mobile devices with GPS capabilities has extended the use of GIS to companies in very different domains.

Despite the differences in their functional scope, most GIS applications share a common set of concepts, standards, architecture, components, and technologies. For example, all GIS deal with spatial data types (such as points, lines, polygons, or variants of these basic types), coordinate systems, maps, layers (used to organize the information shown in maps), and operations to process spatial data. All these elements are defined in different standards from the ISO committee on geographic information/geomatics (ISO/TC 211[1]) and the Open Geospatial Consortium (OGC[2]), hence existing technologies for GIS development support them in the same (or very similar) way. Therefore, two different GIS are modeled and developed similarly, even if they have a different purpose and functional requirements. Also, due to the nature of the information managed in these application domains, some common structures appear repeatedly, such

Communicated by Adrian Rutle and Manuel Wimmer.

✉ Oscar Pedreira
oscar.pedreira@udc.es

Suilen H. Alvarado
s.hernandez@udc.es

Alejandro Cortiñas
alejandro.cortinas@udc.es

Miguel R. Luaces
miguel.luaces@udc.es

Angeles S. Places
angeles.saavedra.places@udc.es

[1] Universidade da Coruña, Centro de investigación CITIC, Database Laboratory, Elviña, 15071 A Coruña, Spain

[1] https://committee.iso.org/home/tc211.

[2] http://www.opengeospatial.org/.

as, for example, network structures (e.g., in domains such as road networks, telecommunications, or energy supply), or hierarchical decomposition of entities at different spatial levels (e.g., territory administration or facility management).

In model-driven engineering (MDE), models play a central and active role in the software development process, far beyond just describing the system. Models describe the software system, and they are artifacts that can be processed to be successively and automatically transformed into models at lower levels of abstraction, and, finally, into the source code of the system [2,3]. In MDE, a *metamodel* describes the elements that can be used in the models that conform to that metamodel. The traditional approach to MDE considers a fixed number of metamodeling levels. Typically, objects are described by models that define their state, behavior, and relations, and these models use concepts defined in a metamodel at a higher level of abstraction. At the same time, at the metamodel level, we can use elements defined in a metametamodel defined at a higher level. A common approach to MDE is based on the OMG's[3] *model-driven architecture* (MDA)[4], which defines four layers for software modeling: computational independent models, platform-independent models, platform-specific models, and system code. The OMG also defined a standard for *meta-object facility* (MOF) that defines the way to create *domain-specific modeling languages* (DSML), usually, through meta-modeling based on two levels of abstraction.

Working in one metamodel level implies balancing the scope of the metamodel and the flexibility of the solution. On the one hand, a simple metamodel would only define basic elements that could be potentially used in any other model. This solution would be simple but would force us to repeat the same information structures in different systems. On the other hand, creating a complex and rich metamodel that tries to define those information structures may be too rigid since it would be difficult to adapt to the particularities of a specific model. In [4–6], for example, we opted for creating a simple metamodel with just the basic elements common to any GIS (basic entities with a spatial component, layers, and maps). The resulting metamodel is very flexible, but it forces the designer to repeat many elements in different models.

A recent trend in MDE is *multilevel modeling* [7–9]. The idea of multilevel modeling is that the number of metamodeling levels is not fixed, so the designer can use the number of levels that better fit a particular domain. This approach aims at simplifying the complexity of the models through the separation of specific domain concepts that can be modeled at different levels. Multilevel modeling solves some drawbacks and restrictions that can occur in the traditional two-level modeling, which forces the description of the application domain in one level, something that can lead to an unnecessary complexity [7]. Despite the attention multilevel modeling is attracting, few works focus on its application to real scenarios. This has already been pointed out by de Lara et al. who mention in [7] that "there are scarce applications of multilevel modeling in realistic scenarios", and by Frank who also pointed out in [10] that "only little attention has been paid to applying multilevel modeling to particular domains".

In this work, we address the modeling of geographic information systems with a multilevel approach. Our main goal was to determine if a solution for GIS based on multilevel modeling could solve the drawbacks of a two-level solution, considering the following requirements: (R1) *Scope:* the set of models should be rich and include the typical elements in most GIS models; (R2) *Reuse of common structures:* the set of models must support the definition and reuse of common structures appearing in many GIS application domains; (R3) *Flexibility:* the solution must allow the designer to adapt high-level designs to the particularities of an application, and be easily extensible to incorporate new elements; (R4) *Realistic:* the solution must consider (not necessarily in an exhaustive way) scenarios extracted from existing proposals for the modeling of GIS; and (R5) *Generality:* the solution must not be limited to one particular case.

We present a proposal with a set of models that are based on international standards for geographic information systems. The first scenario applies multilevel modeling to the conceptual standards defined by ISO TC/211 and the implementation standards defined by the OGC to bridge the gap between these two sets of standards. The following three scenarios were extracted from the data specifications of the European Union INSPIRE Directive, which defines a set of models for information regarding resource and environmental management so that EU member countries can follow them to ensure interoperability. We have selected the application domains of territory administration, spatial networks, and facilities management. The four scenarios show the advantages that multilevel modeling may bring when compared with a two-level approach.

The motivation and contribution of this article are twofold. First, the models of the INSPIRE Directive of the UE provide a general design of GIS for different purposes in the public administration. That approach is based on well-known international standards and tries to provide a general solution to different GIS problems that may fit the needs of administrations and companies in different countries of the Union. However, that solution is based on deep and complex hierarchies of classes, which, based on our experience, should be extended even more to adapt them to the particularities of each country. In this article, we present an alternative solution using multilevel modeling and show that it is more flexible and easily adaptable to those particularities than a traditional

[3] Object Management Group: http://www.omg.org.

[4] Model Driven Architecture: http://www.omg.org/mda.

solution based on two levels. Second, we believe the solution we present also provides a real application scenario for an emerging modeling approach as multilevel modeling, which allows us to compare it with an existing design based on a traditional two-level approach, and that can contribute to understanding the advantages of this modeling technique in a real scenario.

The rest of the article is structured as follows: Sect. 2 presents background and related work on GIS, applications of multilevel modeling, and previous applications of MDE to GIS. In Sect. 3, we present our proposal for developing GIS under a multilevel modeling approach. We describe four scenarios with problems that appear in real-world GIS applications, we then present example metamodels based on international standards for each scenario, and we describe the advantages of using multilevel modeling. Section 4 presents a discussion and evaluation of the solution presented in Sect. 3. Finally, Sect. 5 presents the conclusions of the paper, the work we are currently undertaking, and lines for future work.

2 Background & related work

In this section, we first introduce GIS and describe the main features of these systems. Then, we review existing works on the application of multilevel modeling. Finally, we present previous applications of MDE and software product lines engineering (SPLE) to the GIS domain.

2.1 Geographic information systems

The main characteristic of Geographic Information Systems (GIS), which is also their main difference from regular information systems, is that they manage entities with a geospatial dimension. That is, an entity is defined by a set of attributes, each one being of a particular type. Commonly used data types, such as *String* or *Integer*, store alphanumeric information. In a GIS, there are specific data types to store a geometric structure in the space, like a point or a surface in a specific position in the world. The attributes of geographic data types can store, for example, the location of a building or a meteorological station, the paths of a road network, or the area covered by a forest. Having entities with such attributes allows an application to visualize them with maps instead of typical alphanumeric listings, but it also allows an application to make certain operations or analysis with these geographic data, such as getting the ten closest entities to a user position, or the most efficient path across a set of entities. Summing up, GIS have particular features and use specific technologies that allow us to collect, store, process, and visualize spatial information [1].

GIS were traditionally used by public institutions for administering the territory and managing public resources. Lately, the major advances in communication technologies and the technologies used in GIS development have increased the availability of GIS applications, and organizations from many domains are adopting GIS software. Moreover, the appearance of smartphones with GPS capabilities marked an important milestone in the development of GIS because gathering geospatial information is now cheap and easy for any company. This context has made it mandatory in some application domains to use a GIS-based solution to be competitive. For example, in warehouse logistics, GIS are needed to plan transportation routes in the most efficient way; in public transportation, to know which lines are overused or underused, and to decide how to change them accordingly; or even in social networks and advertisement, since knowing the position of the users and their publications enhance the information they collect to improve their algorithms or to enrich the data that afterward is used by ad services. Regardless of the application area or the purpose of each GIS, there are a set of features that are very common among them, such as digitizing geographic data, representing geo-located data in map viewers, the common tools related to these map viewers (from panning and zooming to measuring dimensions or objects within the map, or sorting the different layers), route calculation, etc.

GIS have changed a lot since they first appeared many decades ago [11]. At first, for a long time, each GIS application was developed ad hoc, totally independent from any other GIS application. The problem of that approach is that interoperability between GIS was not addressed in any way. Nowadays, that situation has changed thanks to two organizations, the ISO committee on geographic information/geomatics (ISO/TC 211[5]) and the Open Geospatial Consortium (OGC[6]), which have defined a set of evolving standards related to all levels of GIS. Most GIS software assets follow these standards, and therefore, GIS applications are quite similar, geographic data can be used in different GIS, and GIS components are, in general, interoperable.

2.2 Multilevel modeling and its applications

The traditional approach to MDE considers two metamodeling levels. In the metamodel level, the designer defines the main concepts of the domain. At a lower level, a domain-specific modeling language can be defined from the metamodel to allow the designer to create models of the system. A promising trend within MDE is that of *multilevel software modeling* [8,12,13]. In contrast to a more "traditional" approach, multilevel modeling does not fix the number of metamodeling levels, so the designer could use the number of levels that better fit a particular domain. This

[5] https://committee.iso.org/home/tc211.

[6] https://www.ogc.org/.

approach aims at simplifying the complexity of the models through the separation of specific domain concepts that can be modeled at several levels. Multilevel modeling solves some drawbacks and restrictions that can occur in the traditional two-level modeling [7]. As explained in [14], many modeling languages for this purpose have been proposed and, although they are different in some elements, they all share common features, such as considering that all classes at any level are also objects, and allowing for deferred instantiation of attributes.

Few works have presented applications of multilevel modeling in real scenarios: For example, Al-Hilank et al. [15] applied multilevel modeling in the context of development process improvement in the automotive industry to model the mappings between the concepts that describe the software development process and different quality standards. In Al-Hilank's work, multilevel modeling allowed to model the relations between domain concepts at different levels of abstraction. Frank [10] applied multilevel modeling in the development of systems and models to support IT management, so the concepts of the IT domain could be refined in successive levels. Similarly, Benner [16] applied it to model-based development of user interfaces. Benner's proposal allows to model elements of user interfaces in different levels of abstraction without using deep inheritance hierarchies. Nesic and Nyberg [17] applied multilevel modeling to data integration in the context of software product lines. In [18], multilevel modeling was applied by Rodriguez et al. to the modeling of colored Petri nets. In [19], Rossi et al. presented a multilevel modeling solution to the modeling of IoT applications for the detection of tourism flows, so different aspects and concerns of the system's architecture can be modeled at different levels. All these works share a common motivation: the need to represent abstract concepts in more than two metamodeling levels. Since the area of multilevel modeling is relatively new, some of these works have also mentioned issues such as the lack of tools.

de Lara et al. present in [7] a research work focused on "When and how to use multilevel modelling". The authors mention that "unfortunately, there are scarce applications of multilevel modeling in realistic scenarios […]". After analyzing a large set of metamodels from different sources, they identified many domains in which a multilevel approach could be more beneficial than a two-level approach, and they also identified a set of patterns where multilevel modeling may bring advantages. Frank also pointed out in [10] that "only little attention has been paid to applying multilevel modeling to particular domains".

2.3 Applications of MDE to GIS

Although most GIS applications are based on a common set of standards, technologies, and assets, they are usually developed "from scratch" with the help of these tools and libraries. This results in low productivity, long time-to-market projects, and high costs, especially in the maintenance and evolution stages. However, it remains clear due to all the exposed above that GIS is a more than adequate field to apply techniques of semi-automatic software development.

Some previous works have applied elements of MDE to GIS development. Regarding modeling and development, Lisboa-Filho et al. [20,21] proposed a UML profile to support GIS-related concepts in UML conceptual models that was used in [21] to generate spatial database. That UML profile was also used in an MDA architecture to generate the SQL DDL code for spatial databases [22]. Many GIS standards from ISO, OGC, and INSPIRE include metamodels covering different parts of a GIS. Kutzner [23] addressed the model-driven transformation of geospatial data according to different metamodels that can present differences between them.

In [4,5], we have already considered the application of automated software development to the GIS domain and we proposed an architecture and a tool for this purpose combining SPLE and MDE approaches. We analyzed the features of a generic family of GIS products and the components implementing these features, we designed a feature model [24] to represent this set of features, and we defined a traditional two-level metamodel to specify how the data model of the products of our family can be described. Afterward, we implemented a tool supporting the automatic generation of GIS products from these models. In [6], that metamodel was used to define a DSL for GIS development.

Even though our tool and the metamodels it handles are very flexible and complete, this design following the two-level modeling approach has some caveats. First, it forces us to define all possible elements of a GIS in a single metamodel. This does not allow us to reflect in the model common structures in this domain, such as defining entities that refine other entities at higher levels of abstraction. Second, since GIS manage entities with a spatial component in the real world, it is relatively common that some structures appear repeatedly with some adaptations and particularities. Working with a two-level approach does not easily allow us to take any advantage of this scenario. Defining those structures in the metamodel would allow us to use them directly, but they would be difficult to adapt to the particularities of a specific application.

As we will explain in Sect. 3, these disadvantages can be addressed by applying multilevel modeling. For example, let us assume we need to develop a GIS that allows a city manager to handle the road networks, the public transportation networks, and also the electricity, water, and telecommunication supply networks. As we will see in the next section, all these networks share the same structure, although they may differ in specific attributes of the network elements. Our

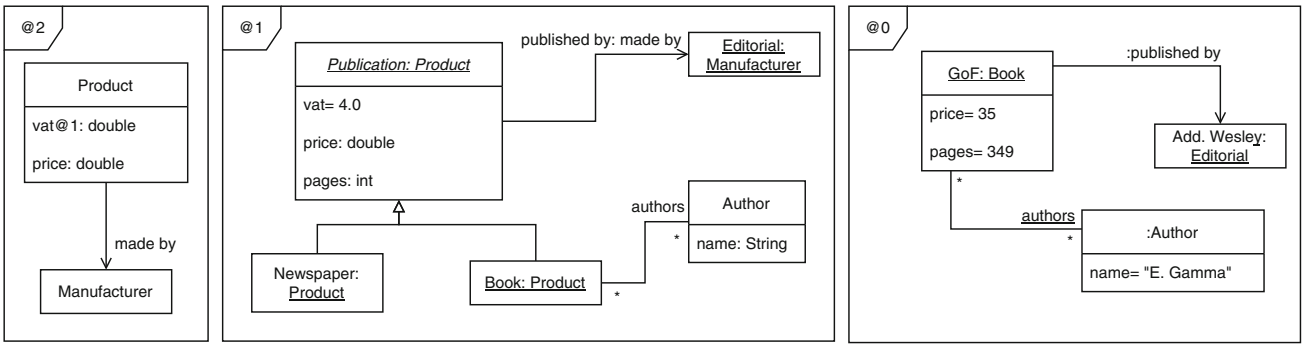

Fig. 1 Example models showing the notation used for multilevel modeling and common multilevel modeling patterns. Inspired by examples from [7]

proposal shows that applying a multilevel approach allows us to metamodel, at an intermediate level, the most common GIS structures, but allowing us to easily extend and adapt them, so we can use these structures to make simpler models in the lower level.

3 Multilevel modeling of GIS

We have identified four scenarios in which multilevel modeling provides a clear advantage over the traditional two-level modeling approach. In each of the scenarios, we describe how they are currently modeled in international standards from ISO, OGC, and the EU INSPIRE Directive, then, we present our proposal for modeling those scenarios with multilevel modeling, and we end with a discussion of its advantages compared with a two-level approach.

3.1 Multilevel notation and patterns

Before describing our proposal, we introduce the notation used in the multilevel models and a set of recurring patterns that will be addressed during the discussion of our multilevel proposals.

There is not a standard language for multilevel modeling yet. Lately, there has been an attempt to identify the most common characteristics of the existing proposals, or the requirements that these proposals target [14]. Tooling support for creating multilevel models is scarce, and the existing prototypes have certain limitations. Given that we have to represent several large models, for convenience we decided to use a standard UML tool to create them.

In this work, we express the multilevel models using the notation presented in [25]. We show an example of the notation in Fig. 1, inspired by examples from [7]. We use the symbol "@" to indicate the potency of the meta-classes or clabjects [26]. The *potency* of an element represents the number of meta-levels a property needs to be instantiated before we get a plain instance, and hence, we have to assign it a value [26]. If an element does not have an explicit potency indicated, it takes the potency of its container. Each element, class, or relationship that instantiates a higher-level element is underlined, and the instantiated element is indicated after the symbol ":". For our explanations along this section, we will use the concepts potency and classification level indistinctly, and when we refer to a meta-level with a particular potency assigned, we use also the symbol "@" in the text (e.g., *meta-level @5* means *the level assigned potency 5*, or *the set of elements with potency 5*). As an example, we describe Fig. 1, where we can see three meta-levels, assigned to potency 2 to 0. In meta-level @2, or meta-level assigned potency 2, we have a meta-class *Product* without any explicit potency. Therefore, it has potency @2, which is the one assigned to the meta-level. Its attribute *vat* has potency 1. Therefore, it must be assigned a value one meta-level below, when the meta-class *Product* is instantiated into *Publication* and *vat* takes the value "4.0". We can also see that the relationship *published by*, with potency 1, is an instance of the relationship *made by*, with potency 2.

When designing meta-level models, there are patterns that appear often, the same way that in traditional two-level modeling there are a well-known set of design and architectonic patterns. An effort to identify and describe these patterns was done in [7], where each pattern is shown with examples and compared to other two-level solutions. These recurring patterns appear on situations where multilevel modeling is adequate, and therefore, we will identify them in the example models we present in this section. The patterns are briefly described next, referring to elements shown in Fig. 1 to illustrate them:

– *Type-object* pattern: modeling types and instances of these types dynamically (e.g., the meta-classes *Publication* or *Book*).

– *Dynamic features* pattern: adding a feature to a dynamic type, and to all the instances of this new type (e.g., the attribute *vat*).
– *Dynamic auxiliary domain concepts* pattern: adding domain-related entities that have relationships with existing dynamic types (e.g., the meta-class *Author* and the relationship *authors*).
– *Relation configurator* pattern: allowing the configuration of a relationship (e.g., the relationship *published by*).
– *Element classification* pattern: dynamically create hierarchies of elements, allocating the features that are inherited by the child types (e.g., the hierarchy descending from the abstract meta-class *Publication*).

Finally, the classes that define geographic data types in the international standards are used as UML data types in the models (e.g., Fig. 3). Just like we can use *String* or *Integer* as the class of an attribute, we use geographic data types such as *Geometry* or *Point*. It may seem that if a class includes an attribute of a geographic type, this should be modeled as a relationship between the class and the class that represents the geographic type. However, we decided not to represent such relationships in that way for three reasons: (1) ISO/OGC UML models consider geographic classes as data types, even though they do not use the stereotype «dataType»; (2) in the logical and physical levels of a GIS, a geographic class works as a data type: It is not implemented using an association between objects in Java, but as a data type, and it is not represented using a foreign key to a table of geographic values in the database, but in the same row of the database; (3) the geographic attribute of the *SpatialEntity* meta-class is used and redefined in many other models, so if we represent it as a relationship it would worsen the legibility of these models.

3.2 Bridging the gap between conceptual and implementation standards in GIS

3.2.1 Overview

In recent years, there has been a great effort of standardization in the field of GIS. Two international organizations (ISO/TC 211 and OGC) have defined around a hundred standards for GIS that cover a multitude of aspects (e.g., conceptual models, logical models, physical models, web services). ISO/TC 211, being an organization composed of the standardization agencies of the member countries, has focused on the definition of conceptual standards aimed at providing solutions for general problems. On the other hand, OGC, being an organization composed of companies in the GIS sector, has focused on the definition of implementation standards aimed at solving problems of interoperability between tools and datasets. This has created a gap between the two sets of standards. Although the two sets of standards are focused on the same sector, they are not formally connected and they just have textual references between them. Multilevel modeling would allow to build a bridge between both sets of standards and have a more consistent and reusable domain description.

Consider as an example the standards that define a model for describing the spatial characteristics of geographic entities using vector geometries. The ISO standard *ISO 19107: Geographic Information - Spatial Schema* [27] defines primitive data types such as *GM_Point*, *GM_Curve*, and *GM_Surface*, aggregate data types such as *GM_MultiPoint*, *GM_MultiCurve*, and *GM_MultiSurface*, as well as many other data types. The data types are structured in a hierarchy that allows application schemas to use data types from the higher levels of the hierarchy to represent spatial entities whose spatial component can be of any of the data types. The schema described in ISO 19107 is conceptual, and it provides no implementation details. OGC Simple Feature Access (OGC SFA) [28], which is also an ISO standard (ISO 19125 [29]), offers the most popular implementation of ISO 19107, describing a common architecture for simple feature geometry. In practice, most GIS data models use the spatial data types from OGC SFA, like *Point*, *LineString*, or *Polygon* (some data types from OGC SFA match the names of ISO 19107 without the namespace prefix). Even though ISO and OGC have worked together on the definition of the standards (in fact, OGC SFA is also an ISO standard, ISO 19125 [29]), the connection between the standards is limited to an informative annex in ISO 19125 that "identifies similarities and differences" with textual descriptions without formal models (e.g., Annex A ISO 19125 [29] states that "MultiPoint in SFA-CA corresponds to GM_MultiPoint in Spatial Schema" but it does not provide a UML model).

3.2.2 Multilevel modeling solution

Figure 2 shows an excerpt of the metamodels that bridge the gap between the ISO conceptual model and the OGC implementation model. The complete models are included in Appendix A (Fig. 14 and Fig. 15). The left part of the figure shows part of the model for meta-level @6 that corresponds to the ISO *GM_Point* data type (a geographic point). The right part shows part of the model for the metal-level @5 that corresponds to the OGC *Point* data type. The OGC data types defined in meta-level @5 instantiate the data types defined by ISO at meta-level @6. We have also defined a class in meta-level @6 (*SpatialEntity*) that we use in the following models (Sect. 3.3, Sect. 3.4, and Sect. 3.5) to describe geographic attributes without specifying the specific type of geographic object until the application schema.

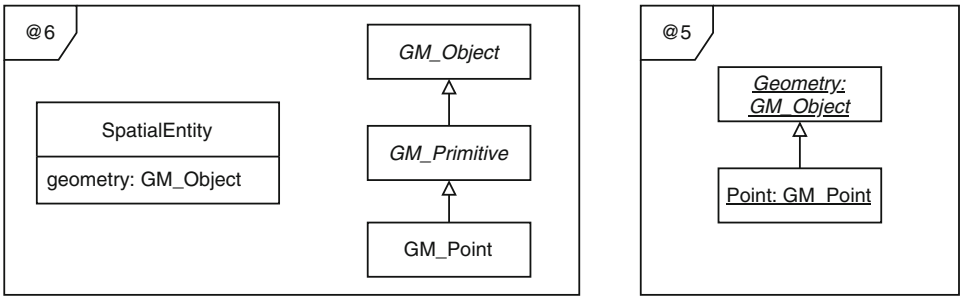

Fig. 2 Excerpt from the multilevel solution to bridge the ISO conceptual model and the OGC implementation model. Figures 14 and 15 in Appendix A show the extended models

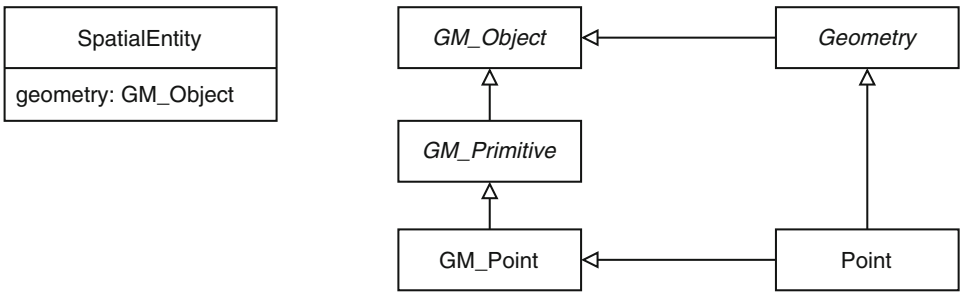

Fig. 3 Two-level solution to bridge the ISO conceptual model and the OGC implementation model

3.2.3 Discussion

If the ISO and OGC standards were defined following a multilevel approach, the connection between the models would be explicit and based on a model, rather than implicit and based on a textual explanation. New implementation standards based on the ISO conceptual standard could be defined (e.g., an implementation model that used Bézier curves instead of line segments in the definition of geometries). These new standards would remain connected with the ISO conceptual model, which would allow applying MDE techniques to all ISO and OGC standards (e.g., using model transformation techniques to convert data between the models).

Furthermore, defining a *SpatialEntity* class at meta-level @6 would allow the decision of the specific data type of an attribute to be deferred until the definition of the application data model. This would allow intermediate models (such as those defined by INSPIRE, see Sect. 3.3, Sect. 3.4, and Sect. 3.5) to be independent of the implementation, and hence allowing developers to design applications without selecting the implementation technology.

Figure 3 shows a two-level model similar to the one presented in Fig. 2. The connection between ISO 19107 and OGC-SFA datatypes has to be made through inheritance, which would reduce the readability of the model. Furthermore, multiple inheritances would cause difficulties in languages that do not support it, in addition to all the traditional difficulties of multiple inheritance in object-oriented programming languages. Furthermore, the *SpatialEntity* class would have to be associated with *GM_Object*, and although a particular application might use a subclass of *GM_Object*, it cannot be explicitly reflected in the application schema.

Regarding the complexity of the models, the number of classes in the multilevel models and the 2-level models is the same because the goal is to replicate the same set of data types. However, the number of associations is much lower because in the multilevel models we instantiate the ISO conceptual model data types into OGC conceptual model data types instead of using inheritance relationships. In particular, we avoid 8 inheritances. There are 6 other inheritances that would be avoided, but they are not shown in our models because we have not included the complete ISO 19107 model because it is a 239 page document in which a large number of data types are defined. As an example, *GM_Curve* is the root of a hierarchy of specialization that includes 17 children classes. The classes *LineString*, *Line* and *LinearRing* from the OGC-SFA model would inherit from some of these classes.

3.3 Ensuring interoperability in spatial data infrastructures

3.3.1 Overview

The European Parliament and the Council of The European Union approved on 2007 the Directive 2007/2/EC establishing an *Infrastructure for Spatial Information in the European Community* (INSPIRE), to create "a European Union spatial data infrastructure for the purposes of EU environmental policies or activities which may have an impact on the environment". This directive is a promising initiative in the management of spatial data in different countries regarding the interoperability and sharing of data.

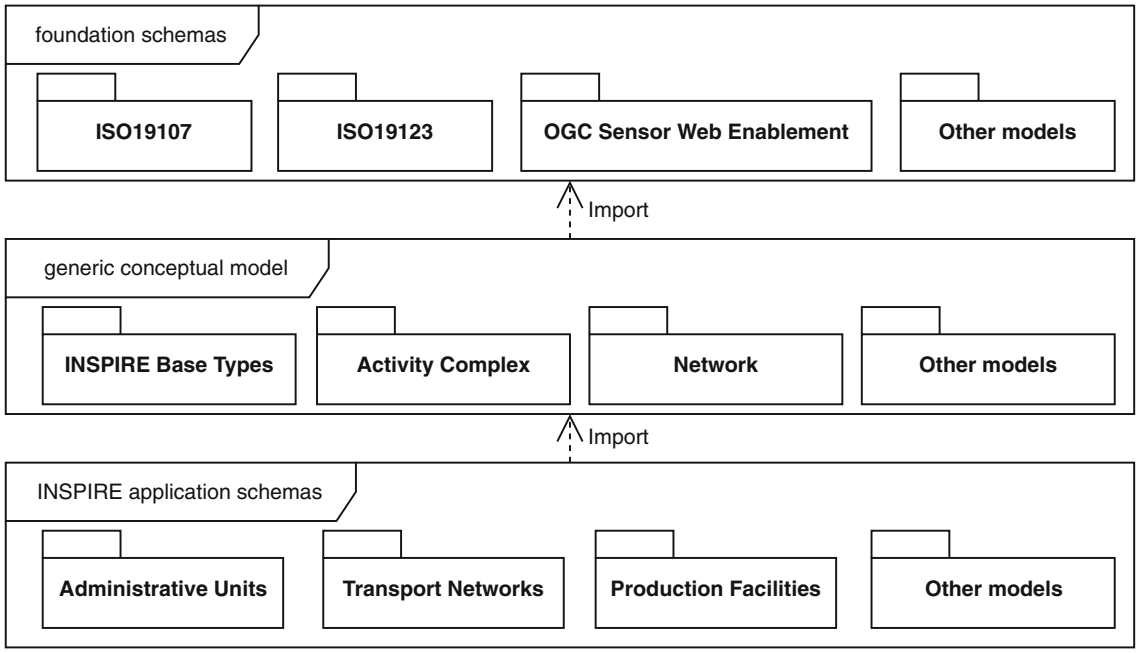

Fig. 4 Overview of the INSPIRE data specifications. The top layer is ISO/OGC international standards, the middle layer is INSPIRE common models, and the bottom layer contains the 34 INSPIRE data specifications

One of the main parts of INSPIRE is the technical guidelines (called data specifications[7]) that specify common data models to achieve interoperability of spatial data sets and services across Europe. Figure 4 describes the organization of these data specifications. The bottom layer of Fig. 4 contains the 34 data specifications that provide UML application schemas for the themes of interest for public administrations regarding resource management and information needed to monitor and define environmental policies. Some example data specifications are Administrative Units, Transport Network, or Production and Industrial Facilities. The middle layer of Fig. 4 contains the UML application schemas defined by INSPIRE that are reused across the 34 data specifications. As an example, INSPIRE defines a generic application schema for Networks that provides basic types that are extended in other data specifications such as the data specification for transport networks (it covers road networks and rail networks among others), or the data specification for Utility and Governmental Services (it covers water networks and electrical networks, among others). The top layer of Fig. 4 contains UML application schemas defined by third parties that are used by INSPIRE data specifications. For example, it contains the ISO standard *ISO 19107: Geographic Information—Spatial Schema* [27].

Considering the characteristics of INSPIRE we have mentioned, this is a clear example of a problem where multilevel modeling is well suited. One of the drawbacks of INSPIRE is its complexity. Across the 34 technical guidelines, we can find many elements that share a set of common attributes and relationships. Sometimes these common elements are not reflected in the technical guidelines, which can lead to repeating the same structures and patterns in different domains. In other cases, those commonalities are identified and considered in the technical guidelines, but through complex and deep inheritance hierarchies that can be difficult to specialize and adapt to the particularities of a specific country.

Consider as an example the application domain of territory administration. Its main responsibility is representing the boundaries of spaces and territories, both rural and urban. Typically, the administration of the territory divides the geographic space into administrative units, which can be composed of other administrative units, and so on. For example, Spain is divided into 17 *autonomous communities* and 2 *autonomous cities*. Each autonomous community is further divided into *provinces* that are divided into *municipalities*. Autonomous cities are not divided into provinces, and they contain a single municipality. On the other hand, Portugal is divided into 18 *districts* and 2 *autonomous regions*; both of them are divided into *municipalities*, which are themselves divided into *parishes*. Hence, each country has its own structure for its territory, with its own names, levels and restrictions.

[7] https://inspire.ec.europa.eu/data-specifications/2892.

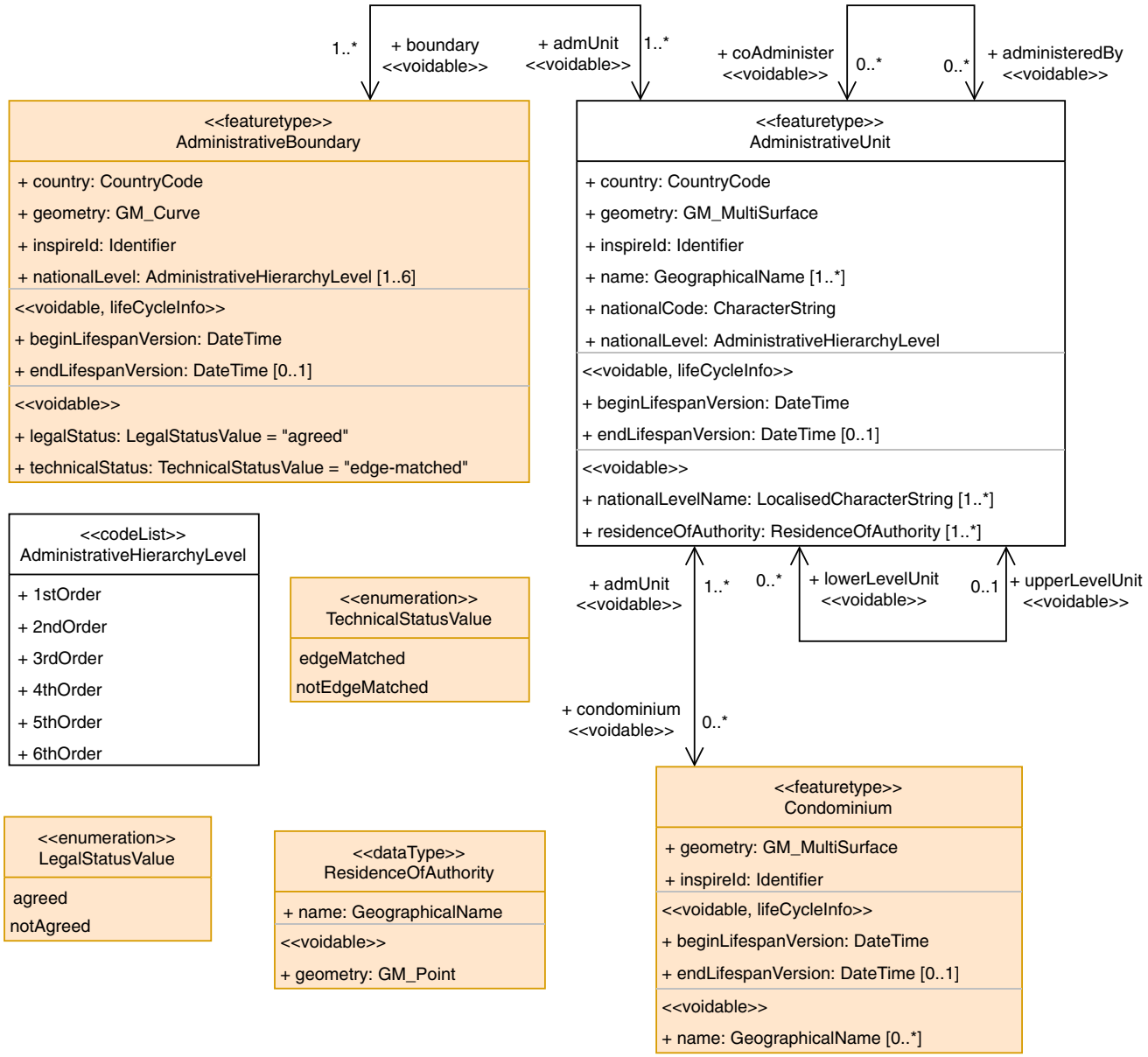

Fig. 5 INSPIRE Administrative units overview, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:1:2:1:7106

To support all this variability, INSPIRE has defined a generic model[8] that can be applied to all of them, independently of their particularities, shown in Fig. 5.

The class called *AdministrativeUnit* represents any part of the territory, from a whole country to a small village. The administrative hierarchy level is stored with a generic enumerated, where values go from *1stOrder* (country) to *6thOrder* (smallest administrative level of a country). Other attributes are the name (or names), the INSPIRE identification (a unique identifier within all Europe), and the geometry representing the surface of the territory. This class also stores information about the country to which it belongs, and an identifier within this country. Finally, there are some optional attributes to represent the version of each instance of the class, to store the *residence of authority* (usually, a capital city), or the name of the administration level of the instance within the country (for example, in Spain the administrative units of third level are called *Provincias*). The *ResidenceOfAuthority* has its own representation, which is very simple, just the name of the place and its geometry. Each administrative unit can aggregate a series of administrative units of a lower level. For example, a country and the regions of this country are

[8] INSPIRE Data Specification on Administrative Units—Technical Guidelines: https://inspire.ec.europa.eu/Themes/114/2892.

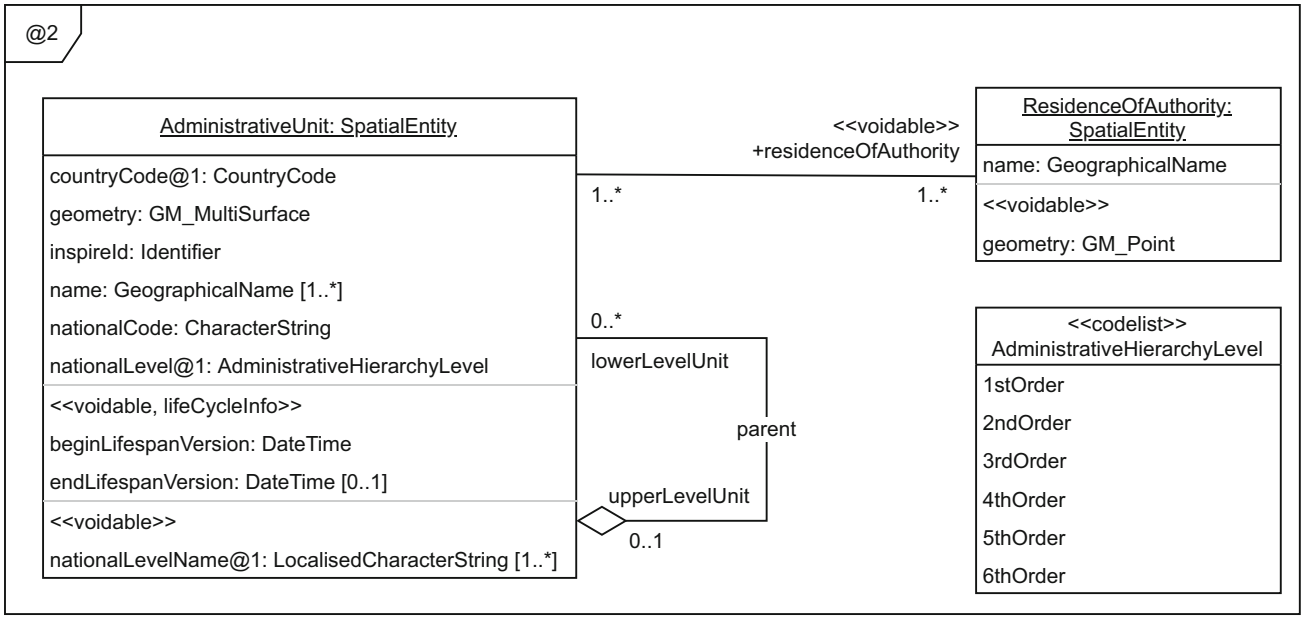

Fig. 6 Modeling multilevel territory administration—meta-level @2

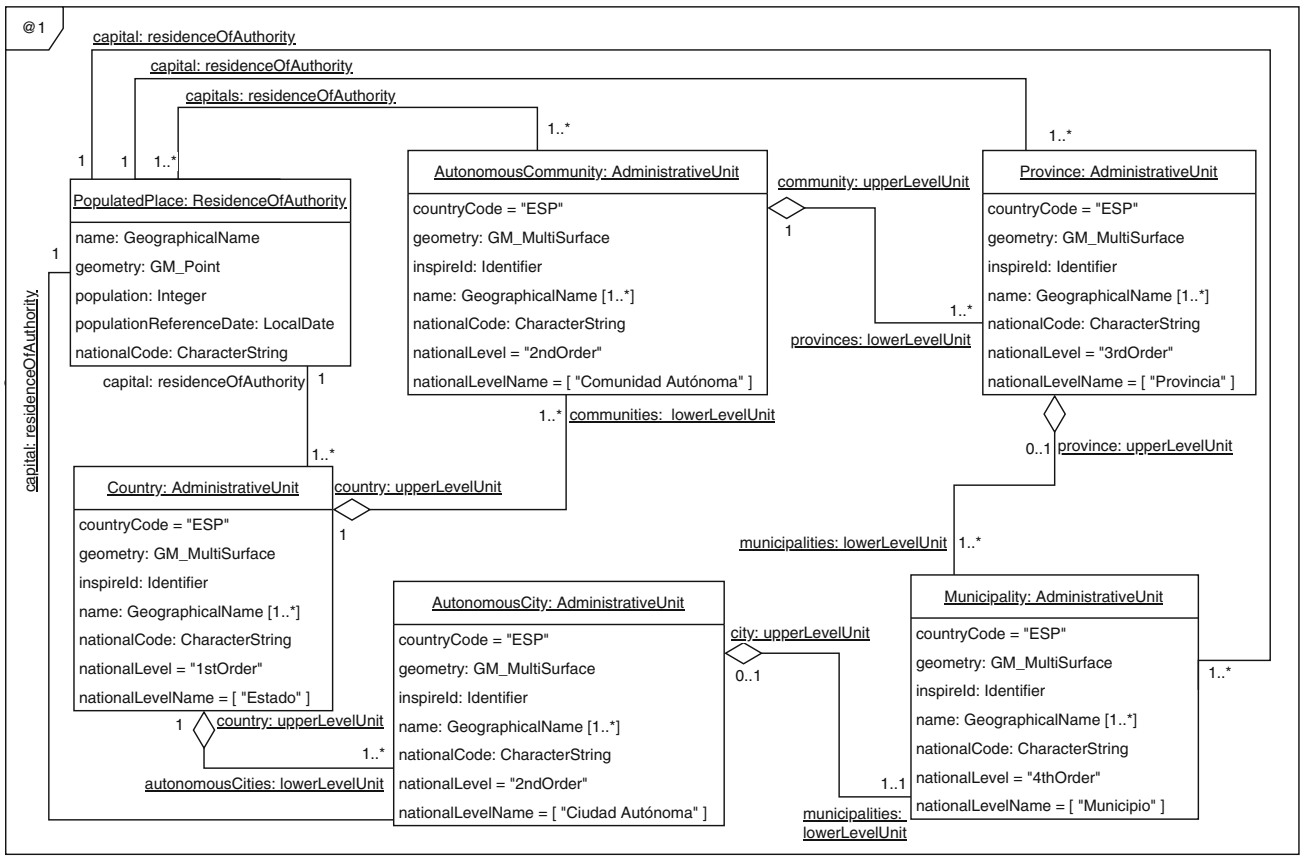

Fig. 7 Modeling multilevel territory administration—meta-level @1 for Spain

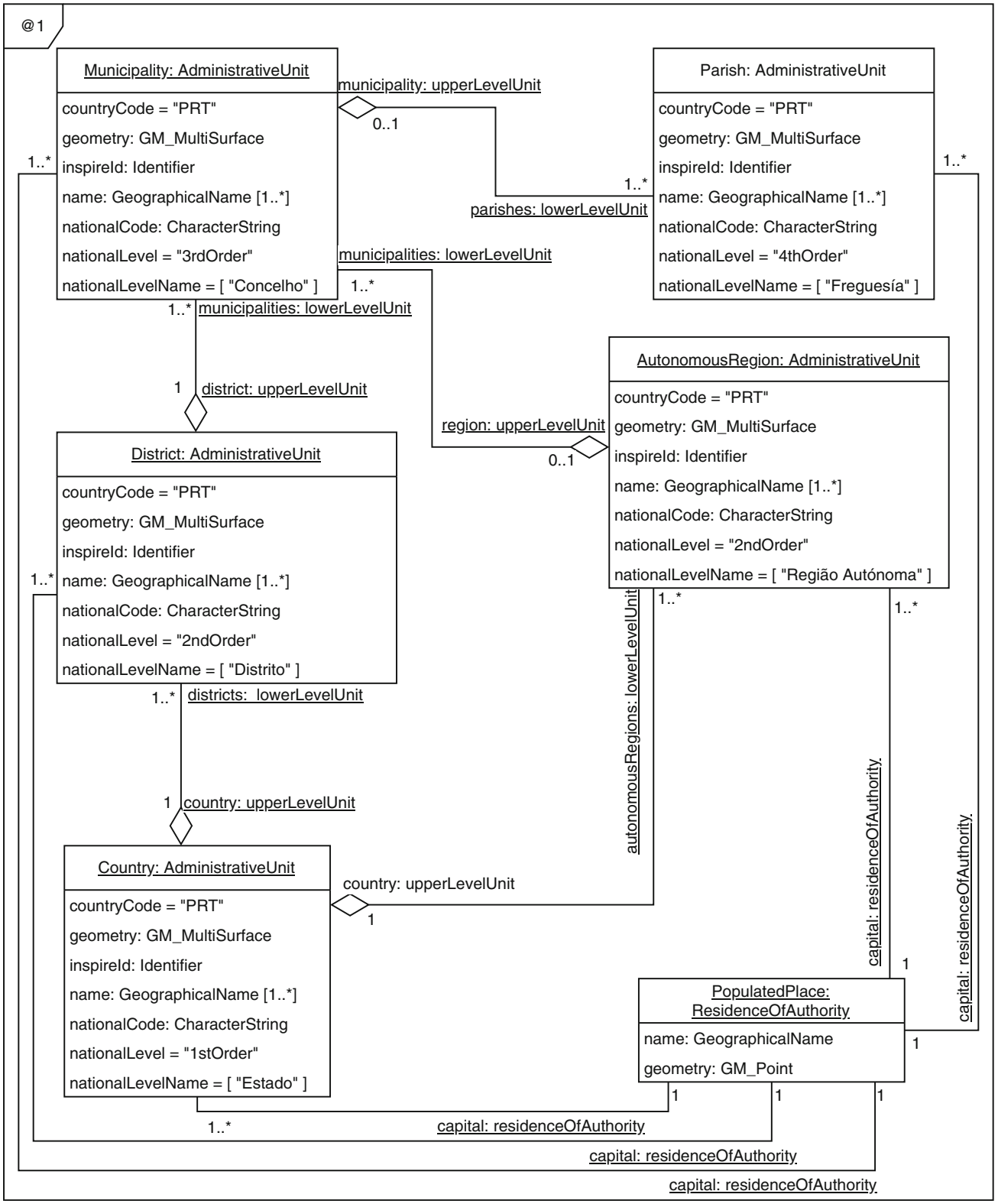

Fig. 8 Modeling multilevel territory administration—meta-level @1 for Portugal

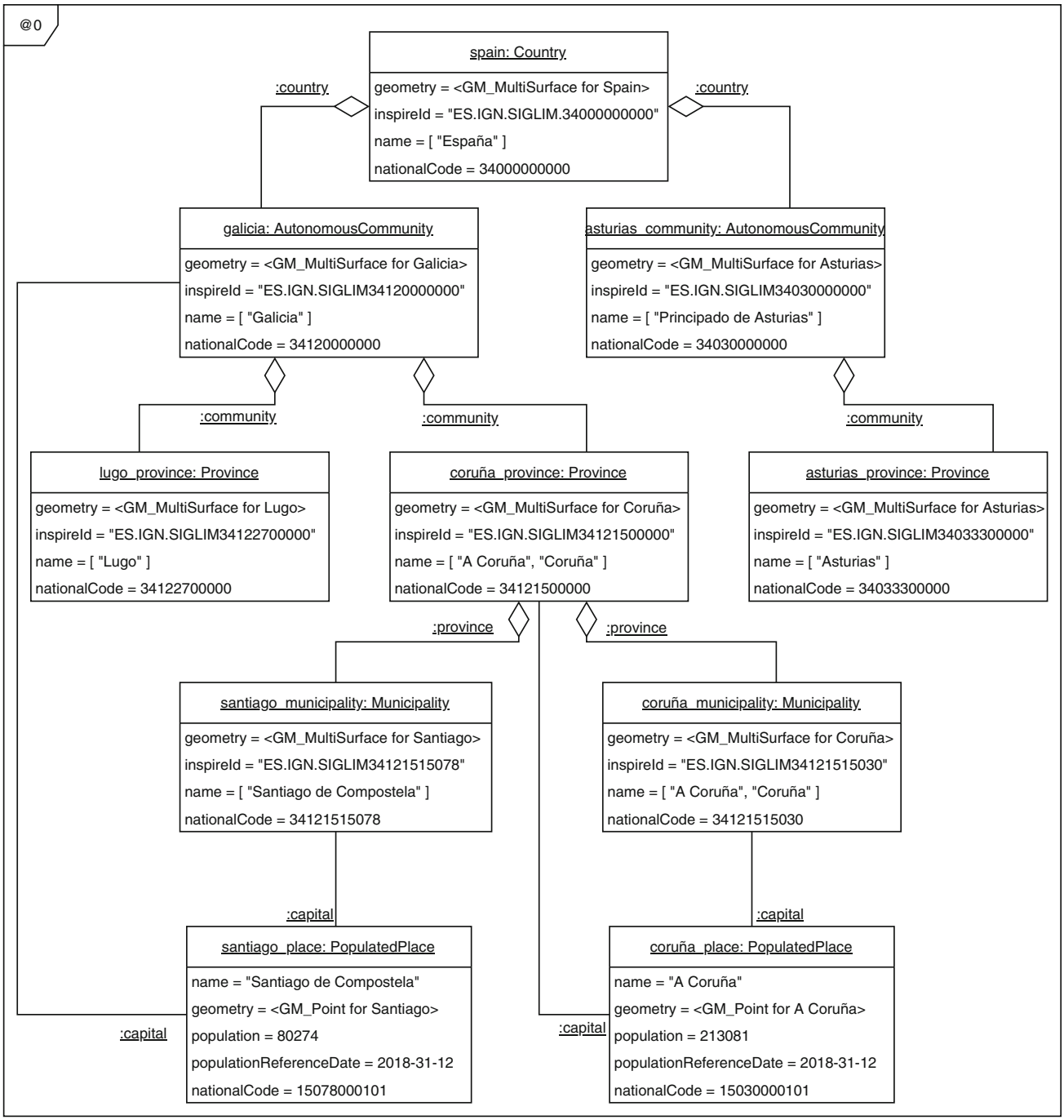

Fig. 9 Modeling multilevel territory administration—example of level 0 for Spain

related, being the former the *upperLevelUnit* and the latter the *lowerLevelUnit* of the relationship we can see in Fig. 5.

Given that the INSPIRE data specifications are abstract, each member country of the European Union is expected to adapt them to their particularities. Even though the model in Fig. 5 can be used to represent the hierarchical division of the territory of a country, it has some drawbacks. First, the semantics of the administrative division of a country are missing. For instance, it is possible that objects from an upper level (e.g., an autonomous community in Spain, a second-order division) aggregate objects from the wrong lower level (e.g., municipalities in Spain, a fourth-order division). Second, a member country may require that objects from each level of the administrative division have additional attributes. For example, autonomous communities and municipalities in Spain have specific legislation but provinces do not. Hence,

the classes for the second and fourth level in Spain require attributes describing the legislation but the class for the third level does not require the attribute.

3.3.2 Multilevel modeling solution

In order to solve these drawbacks, we use multilevel modeling. We define a meta-level @2 in Fig. 6 that is similar to the one defined by INSPIRE. This meta-level @2 model can be instantiated in each member country in such a way that it describes the semantics of its administrative division.

In Figs. 7 and 8, we show the model for the meta-level @1 of two particular countries: Spain and Portugal. Being both of the territorial structures very similar, there are certain particularities in the administration of these countries that are specified in these levels. Spain (Fig. 7) is composed by *AutonomousCommunities* that are composed by *Provinces*, and these ones by *Municipalities*. There are also *AutonomousCities*, composed each one by a single *Municipality*. Regarding the residence of authority linked to the administrative units, all have exactly one *PopulatedPlace* (that represents a city or village) as capital, except in the case of *AutonomousCommunities* because some of them can have more than one capital (e.g., the capital of the Canary Islands is shared by Santa Cruz de Tenerife and Las Palmas de Gran Canaria).

The territorial division of Portugal (Fig. 8) is a bit different. The first level of the administrative division (second order from the EU point of view) consists of *Districts* and *Autonomous Regions* (i.e., Azores and Madeira). Both of them are composed of *Municipalities*, which are themselves composed by *Parishes*.

The particularities of each country are explicit in the meta-level @1 metamodels by redefining the relationships between administrative units and residences of authority. Of course, any extra attribute required for modeling the administrative divisions of these countries could be added at this level (e.g., we have added a few attributes to *PopulatedPlace*), and many attributes are instantiated at this level, such as *countryCode* or *nationalLevelName*, simplifying meta-level @0 models. In Fig. 9, there is an example of the meta-level @0 model of an application based on the metamodel for Spain.

3.3.3 Discussion

The main advantage of applying multilevel modeling in this scenario is that the model of meta-level @2 can be instantiated in such a way that it can capture the semantics of each member country. Hence, the application schema of each member country forbids that objects from an upper level (e.g., an autonomous community in Spain, a second-order division) aggregate objects from the wrong lower level (e.g., municipalities in Spain, a fourth-order division). The same goal could have been achieved in a traditional two-level approach using inheritance, but it would require a complex inheritance hierarchy and the redefinition of the association between administrative units defined in the INSPIRE data specification. This solution would be less reusable and flexible than the multilevel solution that we propose.

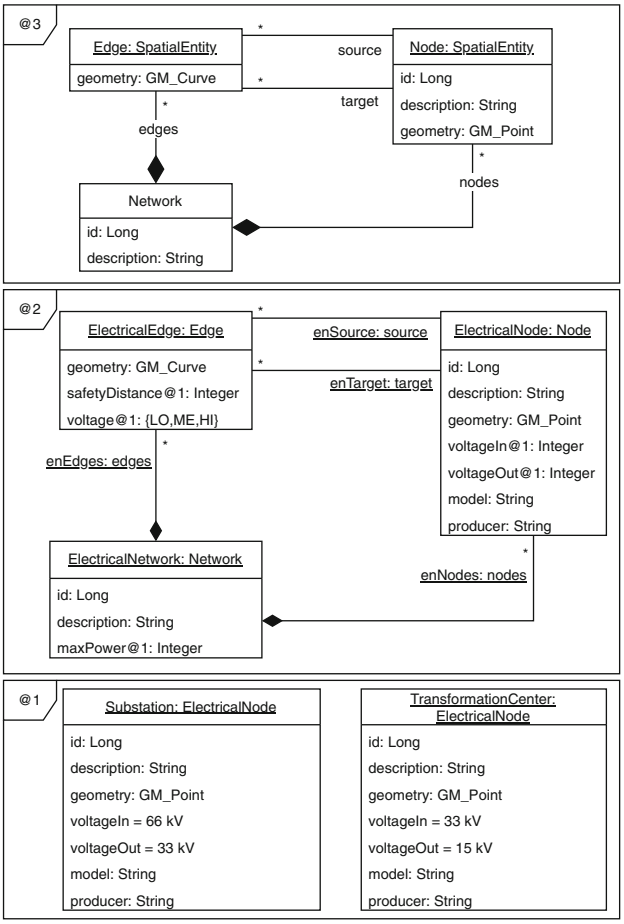

Fig. 10 Modeling multilevel spatial networks—a simple example

Another advantage is that member countries may now easily add additional attributes to specific levels of the administrative division modifying the meta-level @1 as desired. Again, the same goal could have been achieved using an inheritance hierarchy, but the resulting model would not be as clear.

Finally, the advantages described in Sect. 3.2.3 also apply in this scenario. First, the models of the member countries would be explicitly and formally connected, and hence, data transformation techniques could be used to achieve interoperability of the applications. Second, using the *SpatialEntity* class from meta-level @6 would make INSPIRE Data Specifications independent of the implementation and it would still be easy to select a data type from the implementation model in the application schema.

In the example models described, we can find several common patterns of multilevel modeling (see Sect. 3.1):

- *Type-object* pattern: for example, *AdministrativeUnit* is instantiated into different types in lower meta-levels, such as *Country* or *Municipality* (see Fig. 7).
- *Dynamic features* pattern: for example, the attributes included in meta-classes of the meta-level @1, such as *population* or *nationalCode* in *PopulatedPlace* (see Fig. 7). Also, almost every instance of *SpatialEntity* in all the examples includes new attributes, since the meta-class *SpatialEntity* is very generic. For example, *inspireId*, *name* or *countryCode* in *AdministrativeUnit* of the meta-level @2 (see Fig. 6).
- *Relation configurator* pattern: both the relationships *parent* and *residenceOfAuthority* defined in the meta-level @2 (see Fig. 6) are configured or redefined in the meta-level @1, changing the cardinality and the end classes in every case. For example, in the case of the association between *Province* and *AutonomousCommunity*, see Fig. 7.

Regarding the complexity of the models, the number of classes is the same in the INSPIRE models and in our proposal. However, the INSPIRE models would require a large number of inheritance associations (i.e., one for each administrative division of each member country) that are represented as instantiations in our models. Furthermore, the INSPIRE models would not easily represent the semantics of the administrative divisions of each member country because UML does not provide simple notation to specify that an association is specialized in a inheritance hierarchy (i.e., constraints must be used).

As a conclusion, the solution based on multilevel modeling is more expressive, flexible and simple. It also ensures interoperability between member countries because it ensures that the model of each country remains formally connected to the INSPIRE model while allowing the addition of country-specific semantics.

3.4 Modeling common GIS structures

3.4.1 Overview

Spatial networks are one of the most common data model structures in GIS. Many domains require modeling and processing different types of networks. Two of the most common domains are transport networks, such as those representing roads, railways, or flight routes, and resource distribution networks, such as electricity supply, telecommunications, or water supply networks.

From the most abstract point of view, a network is composed of nodes and edges. When a GIS is used to model networks, both nodes and edges of a spatial network are spatial entities because they represent real-world geographic features. Each node has a location, which is usually a point in the space (a *GM_Point* geometry). The edges of the network are defined by the two nodes they connect. If the edge is directed, one node plays the role of the *source*, and the other plays the role of the *target*. In most cases, edges are defined in the space by a line or curve (a *GM_Curve* geometry). Even though the data model for spatial networks is clearly understood (it was already defined in [30]), each GIS application schema has to redefine the same classes for networks composed of edges and nodes instead of referencing the spatial network definition as a common structure.

The idea of reusing the model that defines spatial networks was applied by the designers of the INSPIRE data specifications. Given that spatial networks are required by three data specifications in INSPIRE (namely, transport networks, hydrography, and utility and government services), they have defined a generic network model as part of its base models[9]. Then, each data specification extends the classes in the Generic Network Model to define a common set of base classes for transport networks[10], hydrography[11], and utility networks[12]. Finally, each data specification defines classes for specific network types extending the classes of the common model (e.g., the specification for transport networks defines classes for road railway, air, water, and cable transport networks, and the specification for utility networks defines classes for electricity, oil-gas-chemicals, water, sewer, and thermal networks). The result is a set of quite complex models with a very deep inheritance hierarchy that makes the model quite difficult to understand.

3.4.2 Multilevel modeling solution

Figure 10 shows a simple example of a multilevel solution for modeling networks with four levels. The upper level (not shown) consists of the meta-level @6 of spatial entities defined in Fig. 2. The meta-level @3 is used to model the generic structure for a network, independently of the nature of either the nodes or edges. *Node*s are spatial entities for which the geometry is a point. Besides, all nodes must have an identifier and a description. *Edge*s are spatial entities too, but their geometry is a line. In addition, edges have two

[9] INSPIRE Data Specifications—Base Models—Generic Network Model: https://inspire.ec.europa.eu/documents/inspire-data-specifications-%E2%80%93-base-models-%E2%80%93-generic-network-model.

[10] INSPIRE Data Specification on Transport Networks—Technical Guidelines: https://inspire.ec.europa.eu/Themes/115/2892.

[11] INSPIRE Data Specification on Hydrography—Technical Guidelines: https://inspire.ec.europa.eu/Themes/116/2892.

[12] INSPIRE Data Specification on Utility and Government Services—Technical Guidelines: https://inspire.ec.europa.eu/Themes/136/2892.

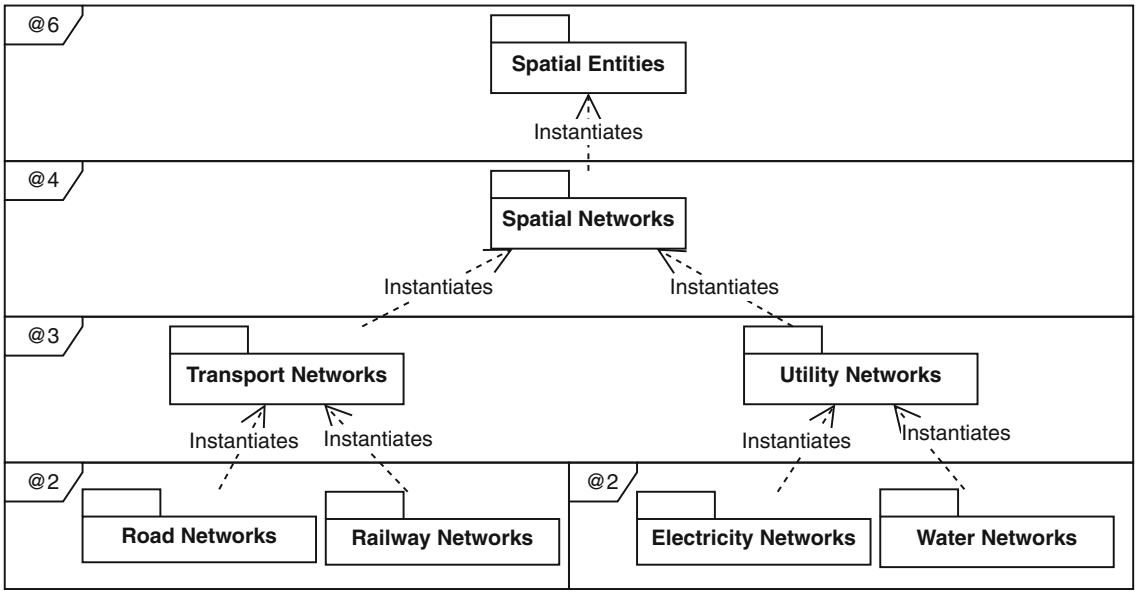

Fig. 11 Modeling multilevel spatial networks—overview

associated nodes (source and target). All these attributes are instantiated in meta-level @0.

The meta-level @2 shows the model of an specific type of network, an *Electricity Supply Network*. As we can see in the figure, the edges need to store data regarding the voltage they transport (which can be low, medium, or high) and the safety distance they must keep to buildings or trees. For each node, we must know the input and output voltages (which can be different, as it happens in the case of transformation centers). Other attributes to store for each electric node can be the model or the producer.

The design we have presented is extended in the next level, in which we define specific classes for specific types of nodes of the network that instantiate some of the attributes. For example, we define a *TransformationCenter* which is a special type of node that transforms medium-voltage electricity into low-voltage electricity, and a class *ElectricalSubstation*, which transforms high-voltage electricity into medium-voltage electricity. In this case, these classes instantiate the attributes *voltageIn* and *voltageOut*, since it is not be necessary to do it in the meta-level @0.

Figure 11 shows the overview of our multilevel modeling approach to the INSPIRE models. The topmost level (i.e., meta-level @6) is the level of ISO 19107 and the class *SpatialEntity* described in Sect. 3.2. The meta-level @4 corresponds to the generic network model defined by INSPIRE. The meta-level @3 corresponds to the common data models defined in INSPIRE for transport networks and utility networks. Finally, the meta-level @2 corresponds to the data models defined by INSPIRE for road and railway networks in the data specification for transport networks, and electricity and water networks in the data specification for utility networks. The INSPIRE data specifications include additional data models for networks that we have not included here for the sake of clarity (e.g., water transport networks, or sewer networks). The meta-level @1 is also left undefined in this paper because it should be defined by each member country with its specific requirements.

Figure 12 shows the meta-level @4 of our multilevel metamodel for modeling spatial networks that replicates the INSPIRE Generic Network Model[13]. A *Network* is composed by *NetworkElement*s, which can be *Node*s and *Link*s (edges), both of them being *SpatialEntities* with geometries. A *NetworkElement* can also be a set of *Link*s (to represent collections of related edges) or a sequence of *Link*s (to represent ordered collections of related edges). *NetworkProperty* can be used to reference a collection of *NetworkElement*s to apply properties to sections of the network. The reference can be applied to the whole *NetworkElement* or a part of it using a point and an offset, or using an offset and a length (i.e., the traditional concept of linear referencing in GIS). This is a design that allows, for example, to specify in a road network that the maximum speed for the first half of a road link is different than the one for the second half).

Figure 17 (in Appendix A) shows the meta-level @3 model corresponding to the INSPIRE model for transport networks[14] (see Fig. 16 in Appendix A), adapted to multilevel approach. As we can see, it redefines some meta-classes from

[13] INSPIRE Data Specifications—Base Models—Generic Network Model: https://inspire.ec.europa.eu/documents/inspire-data-specifications-%E2%80%93-base-models-%E2%80%93-generic-network-model.

[14] INSPIRE Data Specification on Transport Networks—Technical Guidelines: https://inspire.ec.europa.eu/Themes/115/2892.

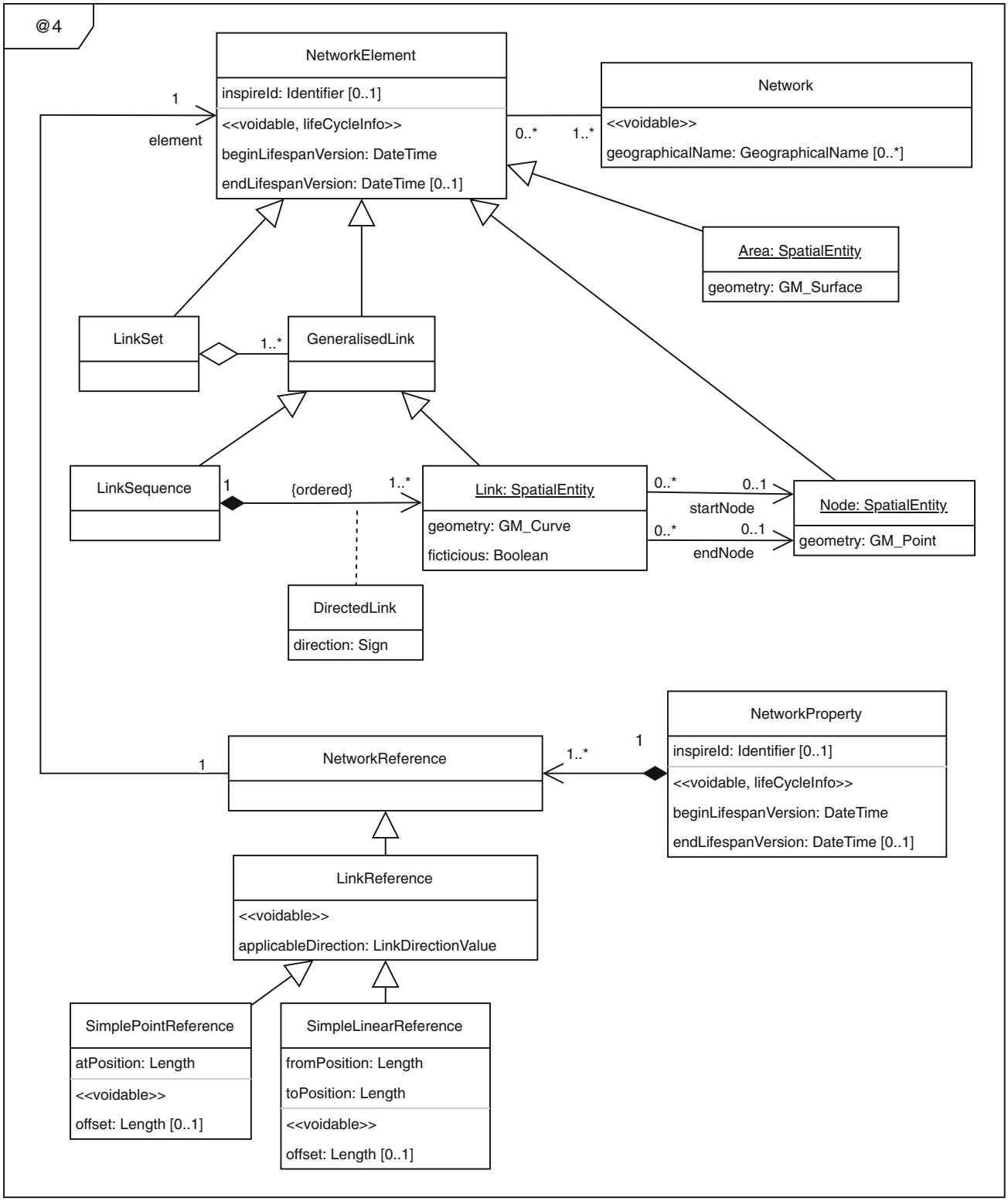

Fig. 12 Modeling multilevel spatial networks—meta-level @4

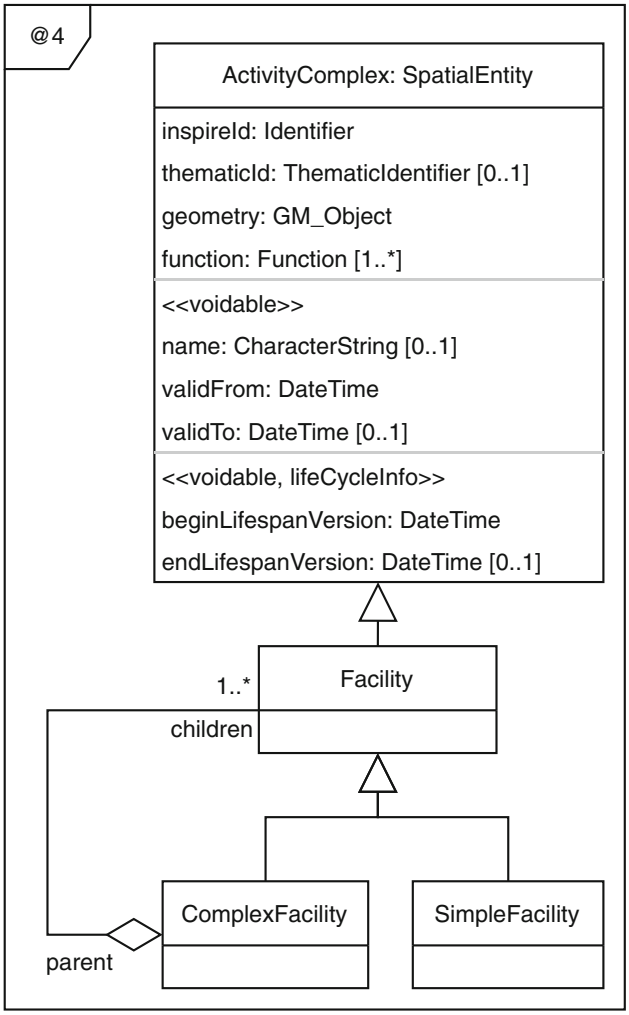

Fig. 13 Modeling multilevel facilities management—meta-level @4

the previous metamodel (Fig. 12), adding the specific elements of the transport networks such as the *typeOfTransport*, an attribute that defines the network nature that is instantiated in the next meta-level (@2), or the properties *MaintenanceAuthority*, *TrafficFlowDirection* and *OwnerAuthority*.

The INSPIRE specifications define a more specific model for each kind of transport network (see Figs. 18 and 20 in Appendix A). We have represented them as the metamodels of the meta-level @2, shown in Figs. 19 and 21 (in Appendix A). These models instantiate meta-classes of meta-level @3 as concrete meta-classes of level @2 (e.g., a *Road* class, a *ERoad* class, or a *RailwayLine* as an instance of *TransportLinkSet*). They also add new attributes when necessary (e.g., the European route number of an *ERoad* or the railway line code of a *RailwayLine*).

We have followed a similar approach for the case of utility networks[15] (see Fig. 22 in Appendix A). The model at meta-level @3 (Fig. 23, in Appendix A) defines the common set of classes used in INSPIRE for utility networks, namely an *UtilityLinkSet* as an instantiation of a *LinkSet* of the meta-level @4, and classes to represent cables, pipes and ducts as specializations of an *UtilityLinkSet*. The model at meta-level @3 also defines a class to represent appurtenances of the utility network as an instance of the a *Node* from meta-level @4. Then, the model at meta-level @3 for water networks (Fig. 25, which replicates the INSPIRE data specification of Fig. 24, both in Appendix A) instantiates the classes from meta-level @2 adding additional attributes to better describe the objects (e.g., the pipe diameter or the pressure). The same occurs with the meta-level @2 model for electricity networks (Fig. 27, which replicates the INSPIRE data specification of Fig. 26, both in Appendix A) that adds attributes to describe information such as the operating voltage or the nominal voltage of an electricity cable.

The proposal stops at meta-level @2 because it corresponds with the INSPIRE specification. A member country is expected to define a level @1 model that takes into consideration the specific requirements for its networks. Furthermore, if we take into account that the electricity networks of a country are built and operated by different companies, it would be possible to define an additional level below the INSPIRE level (that is, raising all the levels from Fig. 11 one level up), adding a new meta-level @2 with elements like those used in Fig. 10. This way, each company that operates an electricity network could define a meta-level @1 model compliant with @2 but taking into consideration the specific needs and functionalities of its information system.

3.4.3 Discussion

The advantages of multilevel modeling in this scenario are similar to those presented in Sects. 3.2 and 3.3: The multilevel model is simpler, it is more flexible, it ensures interoperability between INSPIRE member countries, and it can be easily extended to additional domains. Regarding the complexity of the models, the number of classes is similar because we have tried to replicate the models in the INSPIRE data specifications. However, the number of associations is much lower. In fact, the UML model in the INSPIRE data specification for transport networks use non-standard UML notation to avoid cluttering the diagram with multiple inheritances (e.g., the class *TransportArea* inherits both from *NetworkArea* and *TransportObject*, but this is represented as a small italic text above the UML stereotype instead of displaying the parent classes and the associations).

[15] INSPIRE Data Specification on Utility and Government Services—Technical Guidelines: https://inspire.ec.europa.eu/Themes/136/2892.

The following common patterns of multilevel modeling appear in the example models for networks (see Sect. 3.1):

- *Type-object* pattern: for example, the different types that instantiate *Node* or *Edge* in lower meta-levels, such as *TransportNode* or *Appurtenance* (see Figs. 17 and 23 in Appendix A).
- *Dynamic features* pattern: for example, the attributes added to meta-level @2 *ElectricalNode*, such as *model*, *voltageIn* or *voltageOut* (see Fig. 10).
- *Dynamic auxiliary domain* pattern: for example, in Fig. 10, there is a new meta-class in meta-level @3, *Network*, and a new association between it and instances of *SpatialEntity*. More examples of this pattern appear in the more complex example shown in Fig. 12, where instances of SpatialEntity have different relationships with domain-related elements such as *LinkSequence* or *Network*.
- *Element classification* pattern: for example, the hierarchy descending from the meta-class *RailwayNode*, with its subclasses *RailwayYardNode* and *RailwayStationNode*, which are all of them instances of *TransportNode* from the superior meta-level (see Figs. 17 and 21 in Appendix A).

As a conclusion, the multilevel models are general enough to be used outside the INSPIRE domain and to be applied in any domain that needs the definition of spatial networks. Hence, the multilevel model can be considered and used as a common structure.

3.5 Using common structures in unrelated domains

3.5.1 Overview

In the previous section, we have applied multilevel modeling in a domain that uses a common structure in the field of GIS. There are other application domains that are related but that are not usually modeled using the same common structure. The INSPIRE data specifications provide an example for this scenario. INSPIRE defines some data specifications in the domain of facilities management (e.g., environmental management facilities, agricultural and aquaculture facilities, production and industrial facilities, and buildings). In this domain, it is very common to have a hierarchy of facilities that contain lower-level facilities. For example, a manufacturing plant is usually divided into different facilities. Each facility may consist of different buildings and installations, each one may be in turn divided into different parts. Another example may be an agricultural installation divided into different units with different objectives. Unlike the approach taken with networks, the authors of the INSPIRE data specifications did not consider to provide a generic model for hierarchies of facilities. Therefore, each data specification takes a different approach to model this hierarchy.

3.5.2 Multilevel modeling solution

Figure 13 presents a multilevel solution to the problem that allows us to define a hierarchy of facilities using a composite design pattern that is then instantiated in different models that are able to capture the specific semantics of the INSPIRE data specifications.

The meta-level @4 (Fig. 13) of our solution defines a generic composite pattern to represent hierarchies of facilities using as base class of the composite the *ActivityComplex* class defined by INSPIRE as part of its base models[16] (see Fig. 28 in Appendix A). This class includes an identifier, a geometry (which we include instantiating the *SpatialEntity* class form meta-level @6) and one or several thematic identifiers and functions. It also includes some voidable attributes related to life-cycle and validity information of the instances, which can be used at the object level (meta-level @0) for convenience if the actual application requires them. Then, we define a composite pattern of facilities inheriting from *ActivityComplex* that can be used by models in lower meta-levels to represent the hierarchy of facilities for each specific domain.

Figure 30 (see Appendix A) shows our solution to model environmental management facilities that are used to handle environmental material flows, such as waste or wastewater flows, which is based on the INSPIRE data specification on utility and government services[17] (see Fig. 29 in Appendix A). The meta-class *EnvironmentalManagementFacility* instantiates *ComplexFacility* and redefines its relation since one of these facilities can manage a set of facilities itself. Some attributes of this domain are added, such as a *facilityDescription*, or the *serviceHours*.

In Figure 32 (see Appendix A), we have applied the pattern defined in Fig. 13 to the specification of agricultural facilities[18] (see Fig. 31 in Appendix A). The meta-class *Holding* represents the whole area and the infrastructures within it under the control of an operator to perform agricultural activities. Each *Holding* is composed by individual *Sites*. *Holding* instantiates the meta-class *ComplexFacility*, while *Site* instantiates *SimpleFacility*, being this a very straightforward example of the *ActivityComplex* composite.

[16] INSPIRE Data Specifications—Base Models—Activity Complex: https://inspire.ec.europa.eu/documents/inspire-data-specifications-%E2%80%93-base-models-%E2%80%93-activity-complex.

[17] INSPIRE Data Specification on Utility and Government Services—Technical Guidelines: https://inspire.ec.europa.eu/Themes/136/2892.

[18] INSPIRE Data Specification on Agricultural and Aquaculture Facilities—Technical Guidelines: https://inspire.ec.europa.eu/Themes/137/2892.

A much more complex metamodel is the one for production facilities, as we can see in Fig. 34 (see Appendix A). The data specification for this domain by INSPIRE[19] (see Fig. 33 in Appendix A) defines a class *ProductionSite* representing the surface where one or several *ProductionFacilities* are located. Each *ProductionFacility* is composed itself by a set of *ProductionPlots* (land or water portions destined to functional purposes), *ProductionBuildings* (artificial constructions within the production facility), and *ProductionInstallations*. The latter are composed themselves by *ProductionInstallationParts*, which are single engineered facilities that perform specific functionalities.

The last example is related to the representation of buildings in INSPIRE. The INSPIRE specification for this domain[20] (see Fig. 35 in Appendix A) is a bit more complex than the previous examples because it contemplates two different models to represent buildings: 2D or 3D. Therefore, the data specification defines in a first data model the generic attributes of constructions and buildings, which are then specialized in a 2D model by classes that represent buildings using 2D geometries and in a 3D model by classes that represent buildings using 3D geometries with different levels of detail. In our proposal (Fig. 36, see Appendix A), we have defined a model at meta-level @3 to represent the generic attributes of constructions (i.e., class *AbstractConstruction*) and buildings (i.e., class *AbstractBuilding*). Then, we have represented the composition of building parts into buildings instantiating the classes *SimpleFacility* and *ComplexFacility* from meta-level@4, respectively. Our proposal for the 2D version of the data specification is shown in Fig. 37 (see Appendix A). The class *Building* is instantiated into a class *Building2D*, and the class *BuildingPart* is instantiated into a class *BuildingPart2D*. Both of them are associated with a class *BuildingGeometry2D*.

3.5.3 Discussion

The main advantage of our solution is that we apply a well-known design pattern in meta-level @4 (i.e., a composite pattern, see Fig. 13) that is instantiated in the models of four different application domains. The use of the design pattern allows software engineers to take advantage of all its advantages (e.g., reusability and flexibility), while multilevel modeling allows software engineers to express the specific restrictions of each application domain.

The authors of the INSPIRE data specifications (almost) applied the composite pattern in the definition of the data specification for administrative units (Fig. 5). Hence, the model in the data specification is flexible enough to accommodate the specific administrative divisions of all member countries, but using a two-level solution prevents each member country from defining specific restrictions while conforming to the model in the INSPIRE data specification. However, the authors of the INSPIRE data specifications related to the management of facilities decided not to use a design pattern because they considered that representing the precise semantics of each domain was more important than using common and well-known structures. This decision makes the models more difficult to understand.

Our solution has the advantages of both alternatives. On the one hand, the model at meta-level @4 uses a composite pattern that provides flexibility and reusability. On the other hand, each of the models of each domain captures the semantics of the domain, in particular:

- The meta-level @2 for environmental management facilities does not define any specific restrictions on the composition, just like the data specification does.
- The meta-level @2 for agricultural facilities restricts the composition to two levels (i.e., holdings and sites) retaining the semantics of the data specification.
- The meta-level @2 for production facilities defines a hierarchy (i.e., nested containment) with four levels, just like the data specification does.
- Our proposal for buildings shows two advantages. First, the composite pattern of meta-level @4 is reused while keeping the semantics of the INSPIRE data specification. Second, while the INSPIRE data specification has to use a UML constraint to represent that the part of a 2D building has to be a 2D building part (because the inheritance hierarchy would allow any building part to be part of any building), our proposal explicitly represents the constraint by redefining the association in meta-level @2.

Regarding common patterns of multilevel modeling (see Sect. 3.1), the following appear in the examples of this section:

- *Type-object* pattern: for example, the different types that instantiate *ComplexFacility* or *SimpleFacility* in Fig. 34 (see Appendix A).
- *Dynamic features* pattern: besides the attributes added in the meta-class *ActivityComplex*, that instantiates *SpatialEntity* (see Fig. 13), we have new attributes for example in *EnvironmentalManagementFacility* with respect to *ComplexFacility* (see Fig. 30 in Appendix A).
- *Dynamic auxiliary domain* pattern: for example, new associations are created between instances of *Building*

[19] INSPIRE Data Specification on Production and Industrial Facilities—Technical Guidelines: https://inspire.ec.europa.eu/Themes/121/2892.

[20] INSPIRE Data Specification on Buildings—Technical Guidelines: https://inspire.ec.europa.eu/Themes/126/2892.

and *BuildingPart*, and a new meta-class that instantiates *SpatialEntity* (see Fig. 37 in Appendix A).
– *Relation configurator* pattern: for example, in Fig. 30 (see Appendix A), the association between *Facility* and its subclass *ComplexFacility* is redefined since *EnvironmentalManagementFacility* can be composed of instances of the same class.

The number of classes in these modes is similar to the models in the INSPIRE data specifications because we have tried to replicate them. Regarding the number of associations, our model does not require inheritance associations to *ActivityComplex*, thus reducing the complexity. The number of association between the classes is not reduced in our models, but considering that we have included a composite design pattern, we can say that we have improved reusability and flexibility without adding complexity.

4 Discussion and evaluation

In this section, we evaluate the proposal presented in Sect. 3 with respect to the requirements we described in Sect. 1 for the modeling solution: (R1) Scope, (R2) Reuse of common structures, (R3) Flexibility, (R4) Realistic, and (R5) Generality. Also, we summarize the multilevel metamodeling patterns that occur in the solution and the advantages that we have identified. Finally, we summarize the advantages of the multilevel modeling solutions with respect to their two-level counterparts.

Scope (R1), flexibility (R3) and generality (R5) As we explained in previous sections, working with one metamodel level requires balancing the trade-off between the scope of the metamodel and its flexibility. Adding common elements and structures to the metamodel would enforce all the models to use those elements and common structures as they were defined, and updating them would require updating the metamodel.

As an example, in the case of networks with a spatial component, in a two-level metamodeling solution, all our models would have to conform to the network definition at the metamodel level, and adding new features or admitting potential modifications on that network definition would force us to redefine the metamodel and, probably, add unnecessary complexity to it. Trying to consider all those particularities in one meta-model would be far from flexible. Moreover, the adaptations needed for two different applications could be contradictory, which would pose a problem to the usability of the metamodel. This is the reason why metamodels such as the one presented in [4–6] leaves out of the metamodel some elements of the domain that are interesting.

Something similar happens in the case of territory administration. In the metamodel level, we may define that a hierarchical decomposition of the territory follows a composite pattern. While this is the general case, the specific legal context of two countries may specify a more specific structure. For example, the territory can be structured in autonomous communities, provinces, and municipalities in Spain, but it may follow a slightly different schema in other countries. Trying to reflect all those specificities in the same metamodel would lead to an artificially complex solution. We can find similar problems in the scenario of facility management.

A multilevel modeling solution allows us to define common elements that may be necessary for different applications while having the flexibility and generality to adapt them to the particularities of each project in other metamodel levels.

The solution presented in Sect. 3 defines at each level elements at a specific level of abstraction that can be refined or adapted in metamodels at lower levels. For example, in the case of networks we were able to first define a generic network structure, then refine it to the case of transportation networks, and then adapt it again to the particular case of road and railway networks. In territory administration, we considered a scope that can be adapted to the particular case of any country, and in the case of facilities management, we were able to consider different types of facilities (environmental, agricultural, industrial, and buildings). Therefore, the solution based on multilevel modeling allowed us to address a wider scope. In the multilevel solution, extending the scope of the metamodel does not necessarily imply less flexibility. Therefore, in the multilevel solution, the flexibility and the generality of the metamodel are not determined by the scope of the metamodel (R1, R3 and R5).

Scope (R1), reuse of common structures (R2), and generality (R5) The reuse of common structures may not be relevant in other application domains, but it is especially convenient in GIS. The common information structures we have modeled emerge naturally from the information managed in these systems and its organization in the real world. In the four scenarios we considered in our proposal, we were able to identify those information structures and to model them in a general way that was then adapted to the particularities of specific cases. More specifically, our solution covers the following common structures of the GIS domain:

– *ISO conceptual model and OGC implementation model:* we propose in our solution to apply multilevel modeling to the definition of international standards in GIS. This is a very ambitious proposal since ISO and OGC have proposed 81 and 70 standards each (although not all are related to each other, and there is overlap between the standards). However, it cannot be denied that carrying out this task would meet the requirements of scope, reuse of common structures and generality.

- *Territory administration:* our solution models a general decomposition of the territory for administration purposes (just like the INSPIRE solution does) but allows us to specialize it to the particular territory administration of different countries (in our case, Spain and Portugal).
- *Networks:* from an abstract definition of a network, our solution considers the cases of transportation networks (in our case, road and railway networks), and also utility networks (in our case, electricity and water pipes networks). The solution could be easily extended to consider other types of networks.
- *Facilities:* as in the previous examples, an abstract facility information structure was then refined for managing environmental, agricultural, industrial, and building facilities.

Scope (R1) and Realistic (R4) We consider the solution we have presented realistic (R4) and covers a wide scope (R1) based on existing proposals for GIS that are currently being used in the industry. One of our goals in this work was to create a model solution according to real existing proposals for GIS modeling. Thus, we considered the INSPIRE Directive, a very complete set of models that cover the most typical application areas of GIS. INSPIRE defines 34 data specifications, and our goal was not to model INSPIRE completely. Therefore, we focused on selecting representative examples that show in a real example the benefits of a solution with multiple meta-modeling levels.

As we see in Sect. 2, the INSPIRE data definitions support a large set of concepts and many of the relationships among them. However, the models of INSPIRE can be quite complex. The multilevel models we presented in Sect. 3 provide a much simpler solution, with each level concerning one level of abstraction that can be easily adapted to specific needs of applications at lower levels.

Considering the purpose of INSPIRE and that it is thought to be used in different countries, one of the benefits we appreciate in a multilevel modeling solution in GIS is that it allows us to instantiate attributes, operations, and relationships at different levels. This is particularly important in complex model structures such as the one proposed by INSPIRE.

One of the advantages of multilevel modeling is the capability of deferring instantiation. This is very adequate to our context since certain elements of the models need to be defined depending on the particularities of the application context. For GIS applications based on INSPIRE specifications, the application context depends not only on the requirements of a specific application but also on the country. With our multilevel approach, the particularities of the country can be expressed in a meta-level lower than INSPIRE, but there is still a place for expressing the actual application requirements in the data model of the application, in meta-level @1.

Multilevel metamodeling patterns We analyzed the presence of the different multilevel metamodeling patterns described in [7]: type-object, dynamic features, dynamic auxiliary domain, relation configurator, and element classification. Table 1 shows the patterns that occur in each of the scenarios, we are using all the patterns at some point of the solution, and all the scenarios use at least two multilevel metamodeling patterns.

Summary of the advantages The advantages of multilevel modeling in the GIS domain can be summarized as follows:

- Many organizations are defining models for many aspects of GIS that are strongly related between them. However, these models are completely independent and they are disconnected. If a multilevel modeling approach were used, the models would be formally connected enabling many advantages related to MDE (e.g., applying model transformation techniques).
- Multilevel modeling reduces inheritance in the models. This improves the readability of the models and reduces the probability of requiring multiple inheritance and hence avoiding its problems.
- Interoperability can be improved using multilevel modeling because the organizations that define standards can propose metamodels that can later be instantiated in a more precise metamodel at a lower-level standard organization. The INSPIRE directive is a paradigmatic example in which the European Union would define a metamodel for each of the themes of the European spatial data infrastructure, that would, in turn, be instantiated in a new metamodel in each of the member countries, that would finally be instantiated in an application schema (or even in a new metamodel at a lower-level administrative division of the member country).
- Well-known common structures of GIS applications could be proposed as metamodels that could later be instantiated in the metamodel of specific GIS development tools to be finally instantiated in specific application schemas.
- Software engineering design patterns (e.g., the composite or the strategy design patterns) could be applied to application domains that are currently unrelated to transfer the benefits of the design patterns to the application domain. This would add flexibility to the application domains and it would reduce the complexity of the models and the learning curve of engineers and developers, facilitating the use of standard models instead of ad-hoc solutions for each problem.
- Some modeling constraints can be modeled as first-class entities in multilevel modeling instead of being external expressions in a different language or simple annotations.

Table 1 Occurrence of multilevel metamodeling patterns in the scenarios

Patterns	GIS Scenarios		
	Administration	Networks	Facilities
Type-object	x	x	
Dynamic features	x	x	x
Dynamic auxiliary domain		x	x
Relation configurator	x	x	x
Element classification		x	

5 Conclusions and future work

Geographic information systems manage entities with a geospatial component that plays a central role in the system. Even if two GIS applications have different functional scopes, they will share a set of common concepts, data types for representing geometries, spatial structures (such as territory decompositions or spatial networks), and a set of technologies based on international standards published by different organizations, such as ISO or OGC. These characteristics make GIS a suitable application domain for MDE.

In this work, we have addressed the modeling of GIS following a multilevel modeling approach. More specifically, we have presented a multilevel modeling solution for GIS considering different scenarios: harmonization of basic conceptual and implementation models from ISO and OGC, territory administration, spatial networks, and facilities management. These scenarios have been selected from the European Union's INSPIRE Directive, which defines a set of models for information regarding resource and environmental management so that EU member countries can follow them to ensure interoperability. In each of these scenarios, we have shown how typical elements and structures present in many GIS applications can be modeled in abstract levels to be refined and instantiated at lower levels. We have used these examples to show how multilevel modeling can be applied to bridge the gap between conceptual and implementation standards in GIS, ensuring interoperability in spatial data infrastructures, modeling common GIS patterns, and applying common structures to unrelated domains.

Based on that previous experience and the analysis presented in the discussion sections of this work, and although the set of models presented in this article does not pretend to be exhaustive and is just a part of all the potential elements included in a GIS, we consider that it shows that the application of multilevel modeling in this domain can lead to simpler, more flexible, and more expressive solutions when compared with a two-level approach. The multilevel solution allows us to define metamodels with a larger scope and richness that can be later adapted. It also allows us to model common information structures that appear repeatedly in GIS in a way that allows us to redefine or adapt them to the particular needs of an application. The models defined by INSPIRE provide a solution following a traditional modeling approach, but extending and adapting those models to the particular needs of each country or organization would not always be possible, or it would imply extending inheritance hierarchies that are already very complex. Also, the solution we have presented is based on existing standard models for GIS, which shows that multilevel modeling can lead to a good solution in a realistic view of the domain of geographic information systems. Furthermore, by using INSPIRE models to create a multi-level model for GIS, we have shown that international standards for information systems are a promising application domain for multilevel modeling approaches.

Acknowledgements We would like to thank the reviewers for their valuable comments and suggestions.

Funding This work has been partially funded by grants: MICIU/FEDER-UE, MAGIST: PID2019-105221RB-C41; MICIU/FEDER-UEBIZDEVOPSGLOBAL: RTI-2018-098309-B-C32, Xunta de Galicia/FEDER-UE, ConectaPeme, GEMA: IN852A 2018/14; MINECOAEI/FEDER-UE Datos 4.0: TIN2016-78011-C4-1-R; MINECOAEI/FEDER-UE Velocity: TIN2016-77158-C4-3-R; CITIC research center funded by XUNTA and EU through the European Regional Development Fund-Galicia 2014-2020 Program, grant ED431G 2019/01. Funding for open access charge: Universidade da Coruña/CISUG. Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

A Appendix–Complete models

See Figs. 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36 and 37.

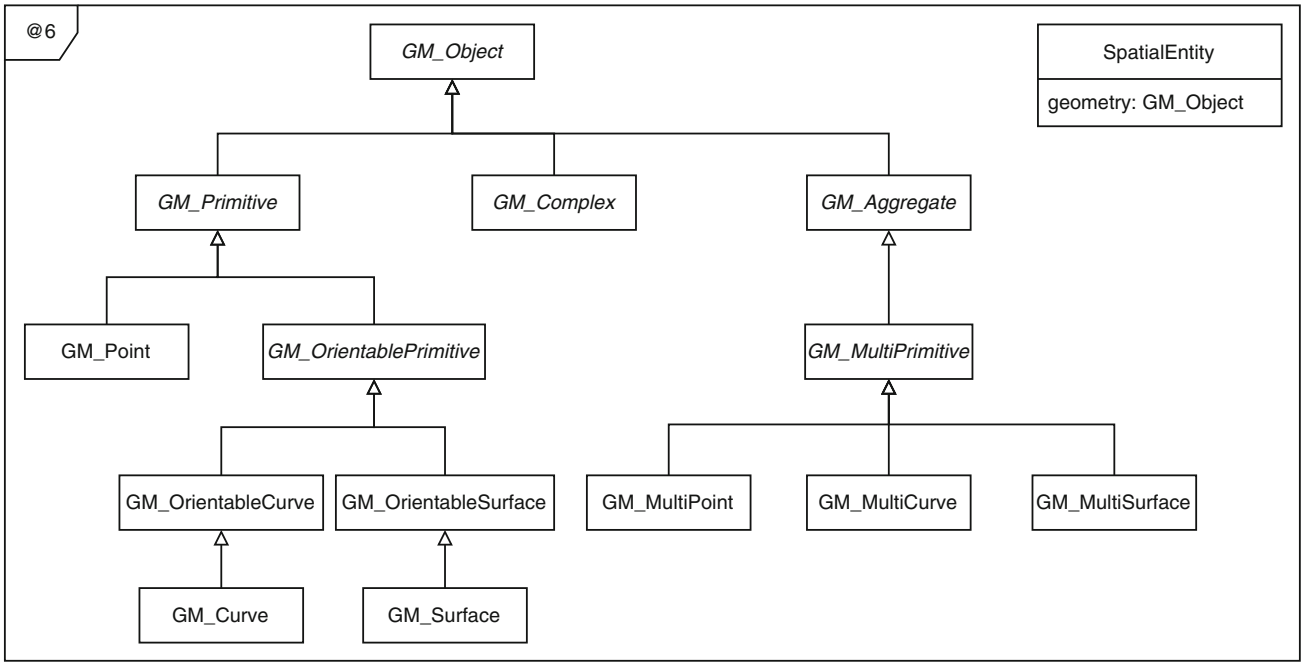

Fig. 14 ISO 19107: Geographic information—spatial schema

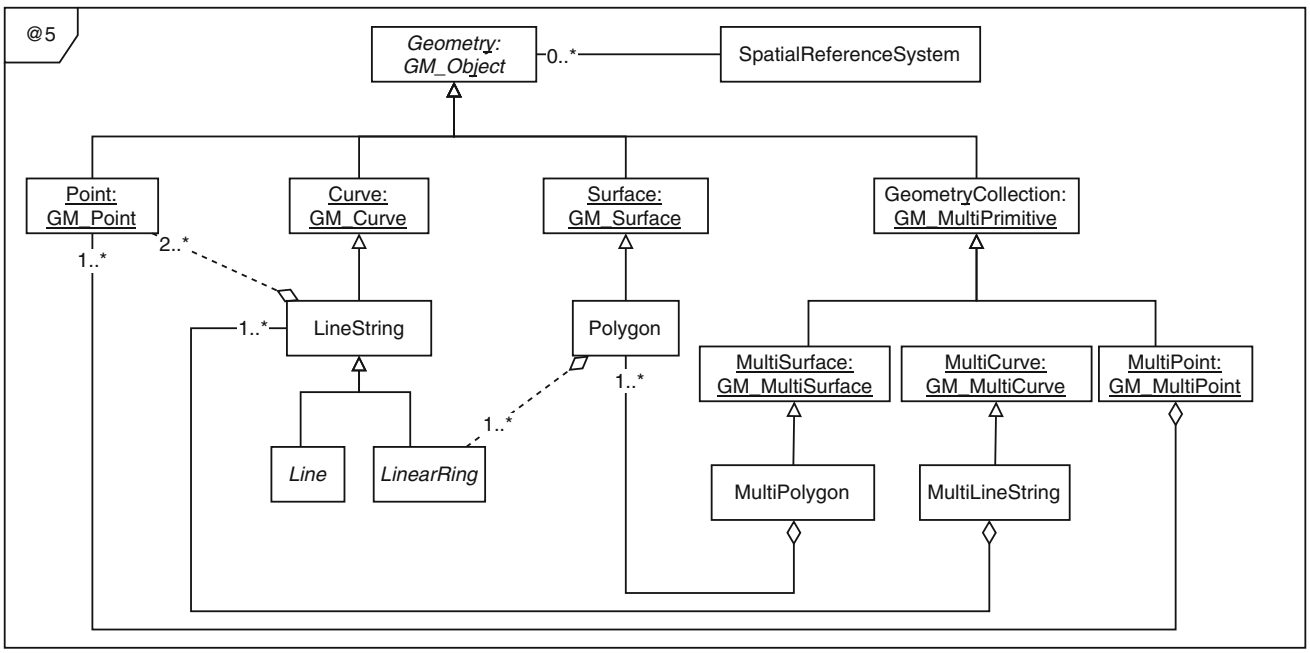

Fig. 15 OGC Simple feature access (OGC SFA)

The scope of this work does not include aspects related to model transformation, code generation, or other implementation aspects, which remain as future work.

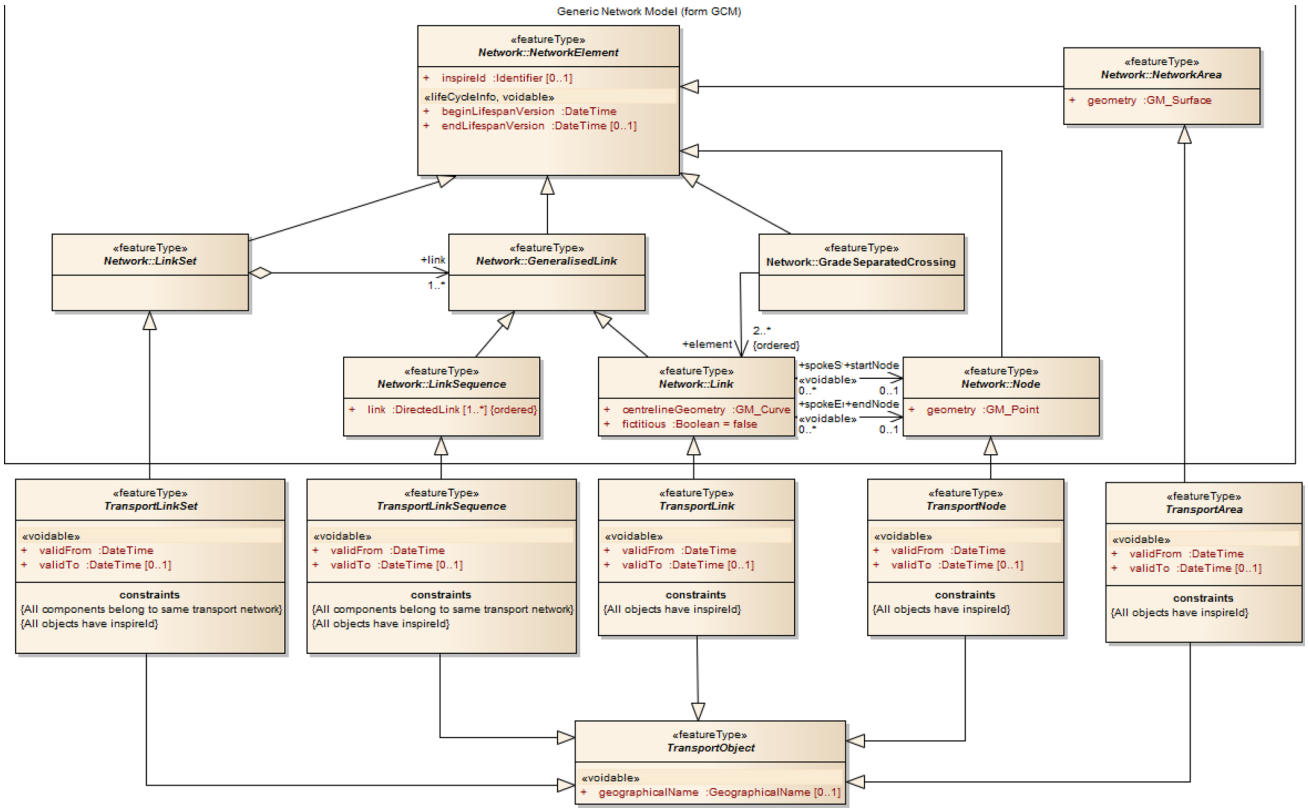

Fig. 16 INSPIRE Network base model and common transport elements overview, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:1:9:6:7590

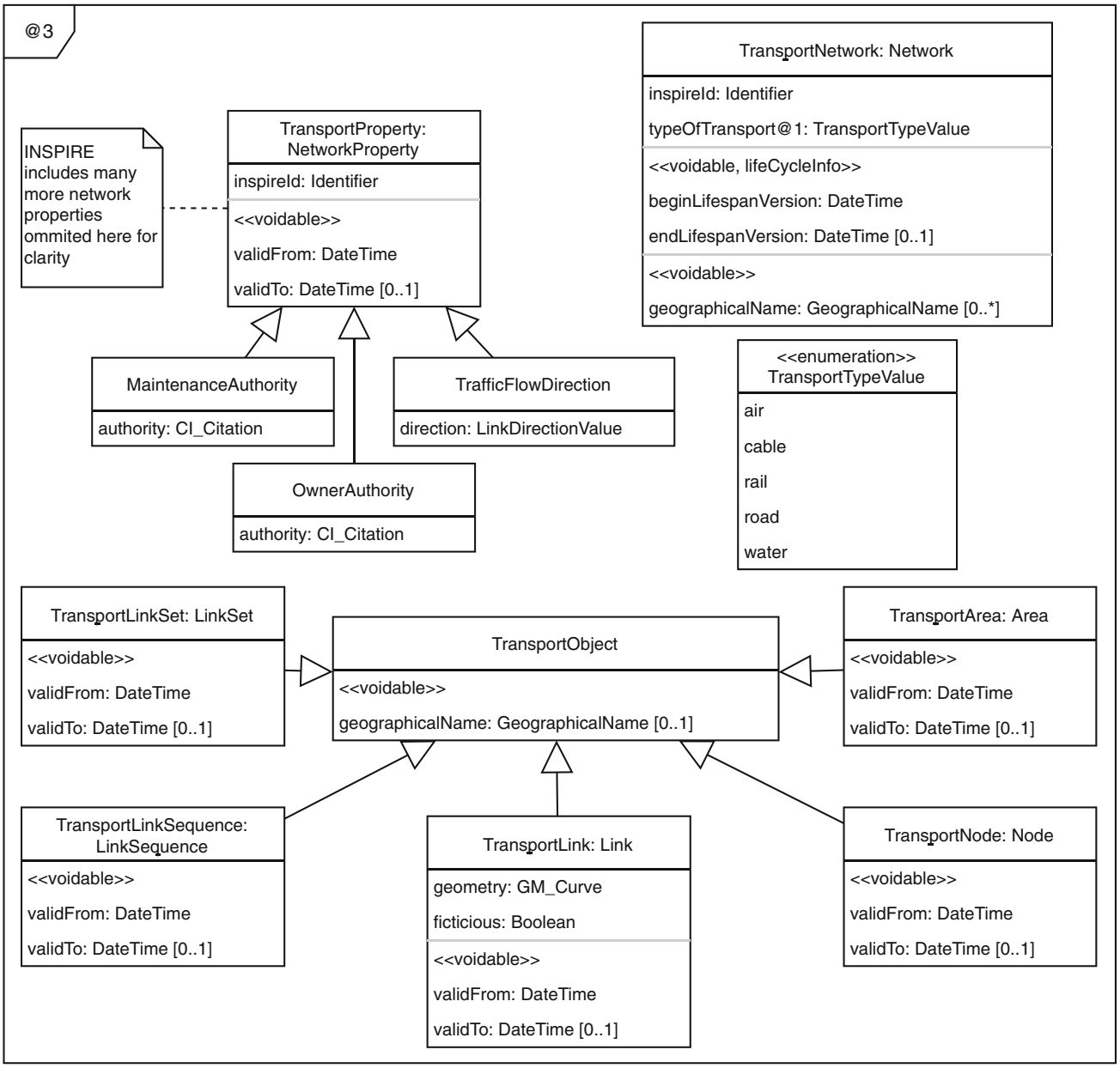

Fig. 17 Modeling multilevel spatial networks—meta-level @3 for transportation networks

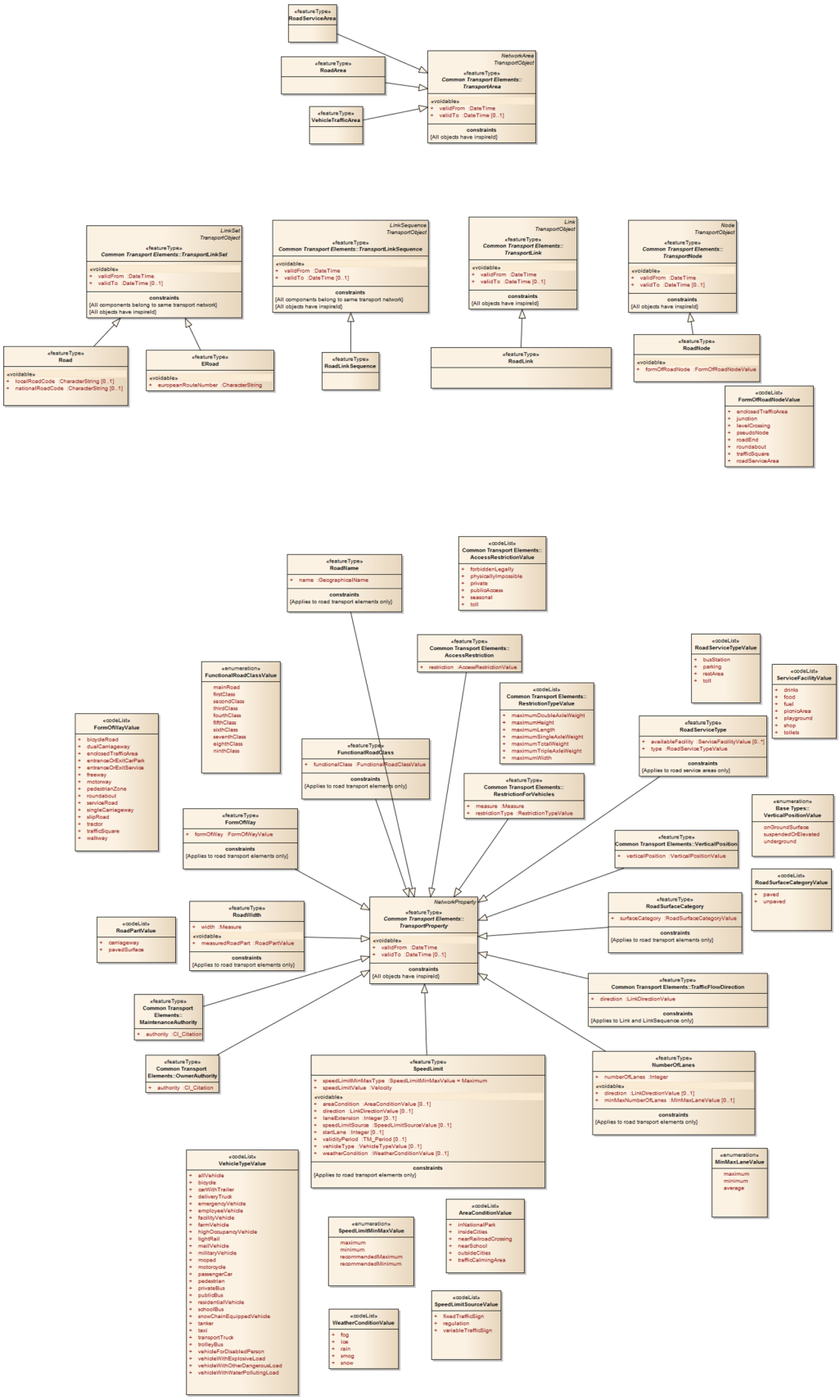

Fig. 18 INSPIRE Road transport network, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:1:9:7:7627

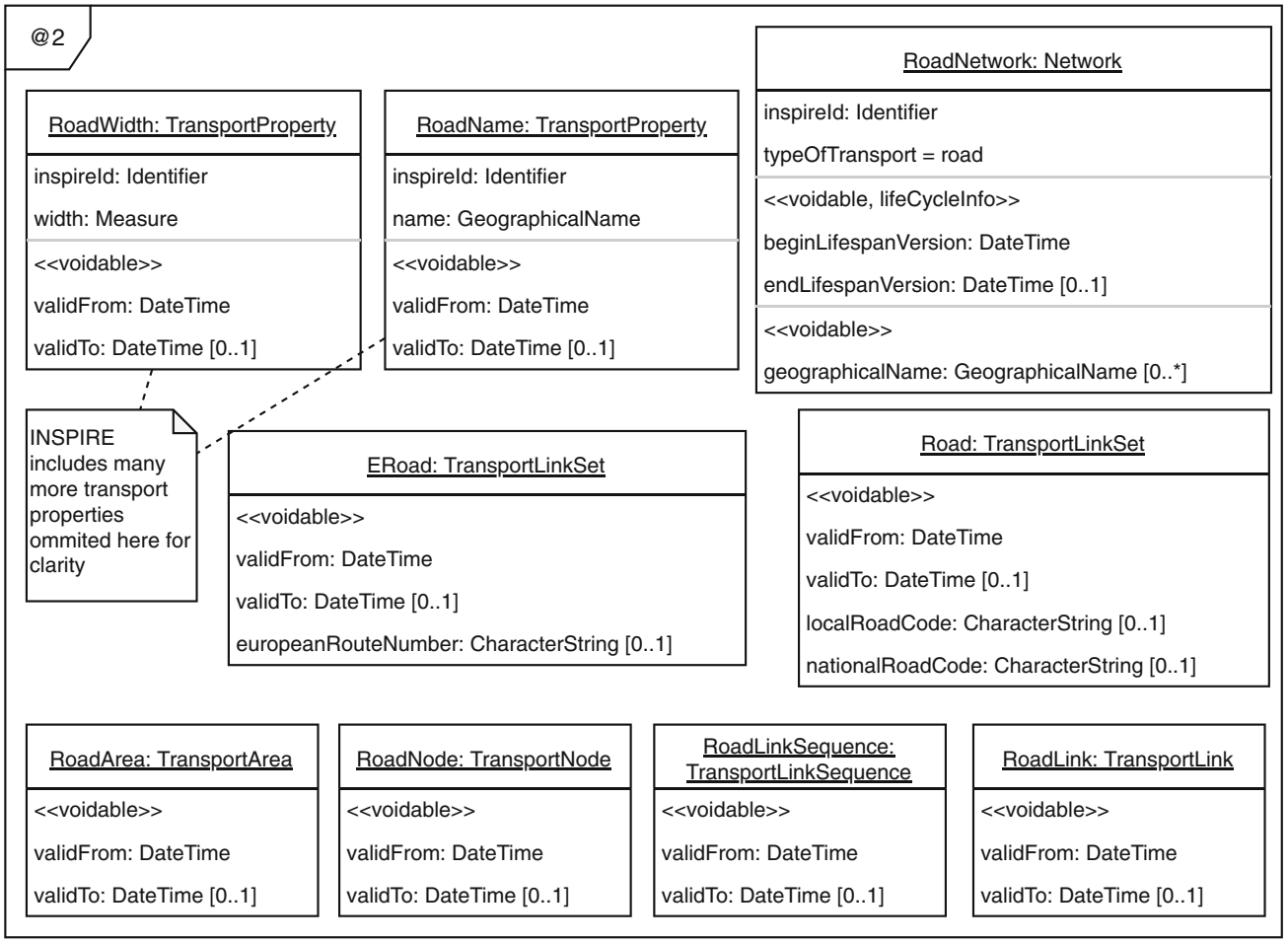

Fig. 19 Modeling multilevel spatial networks—meta-level @2 for roads networks

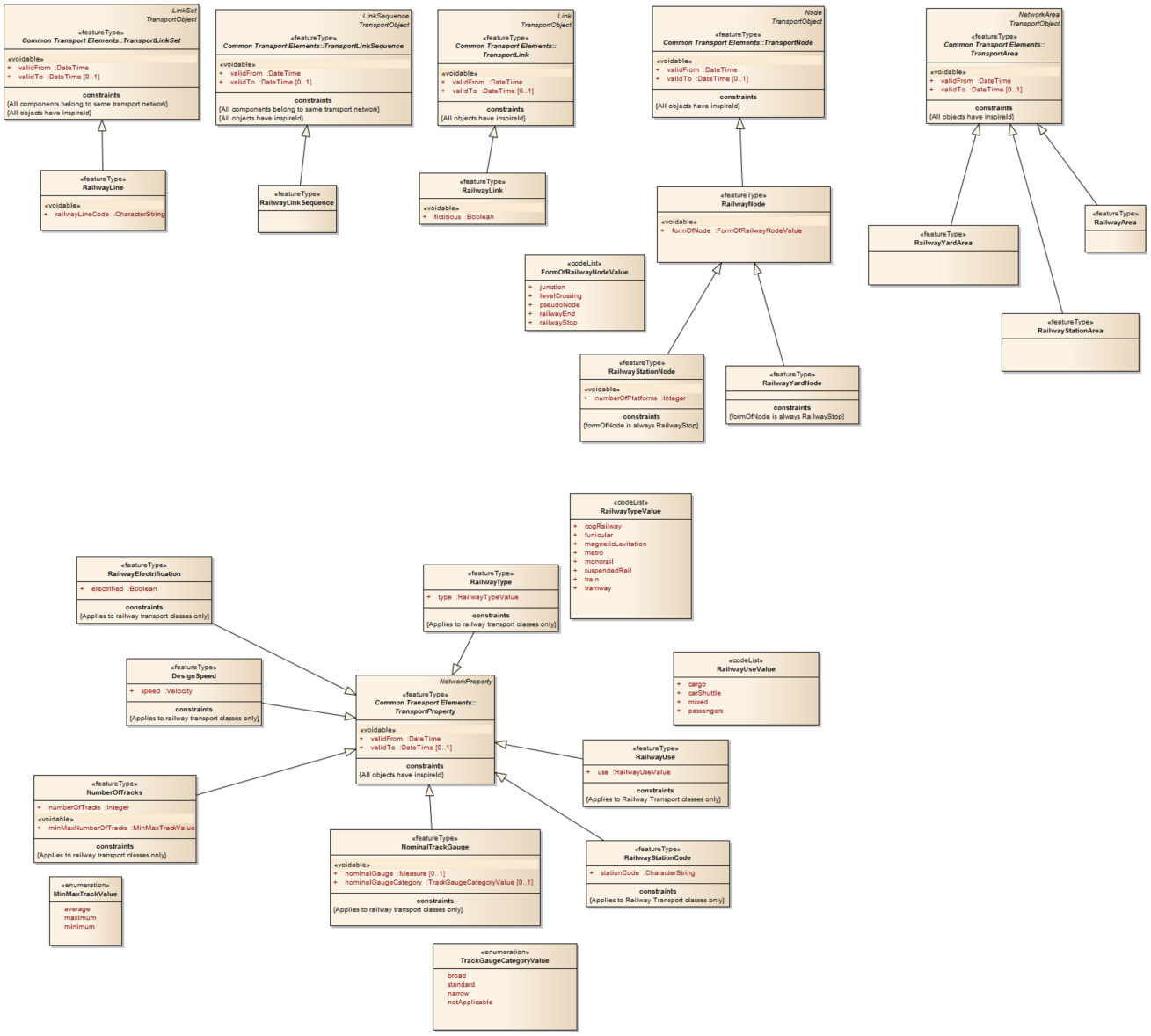

Fig. 20 INSPIRE Railway transport network, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:1:9:4:7508

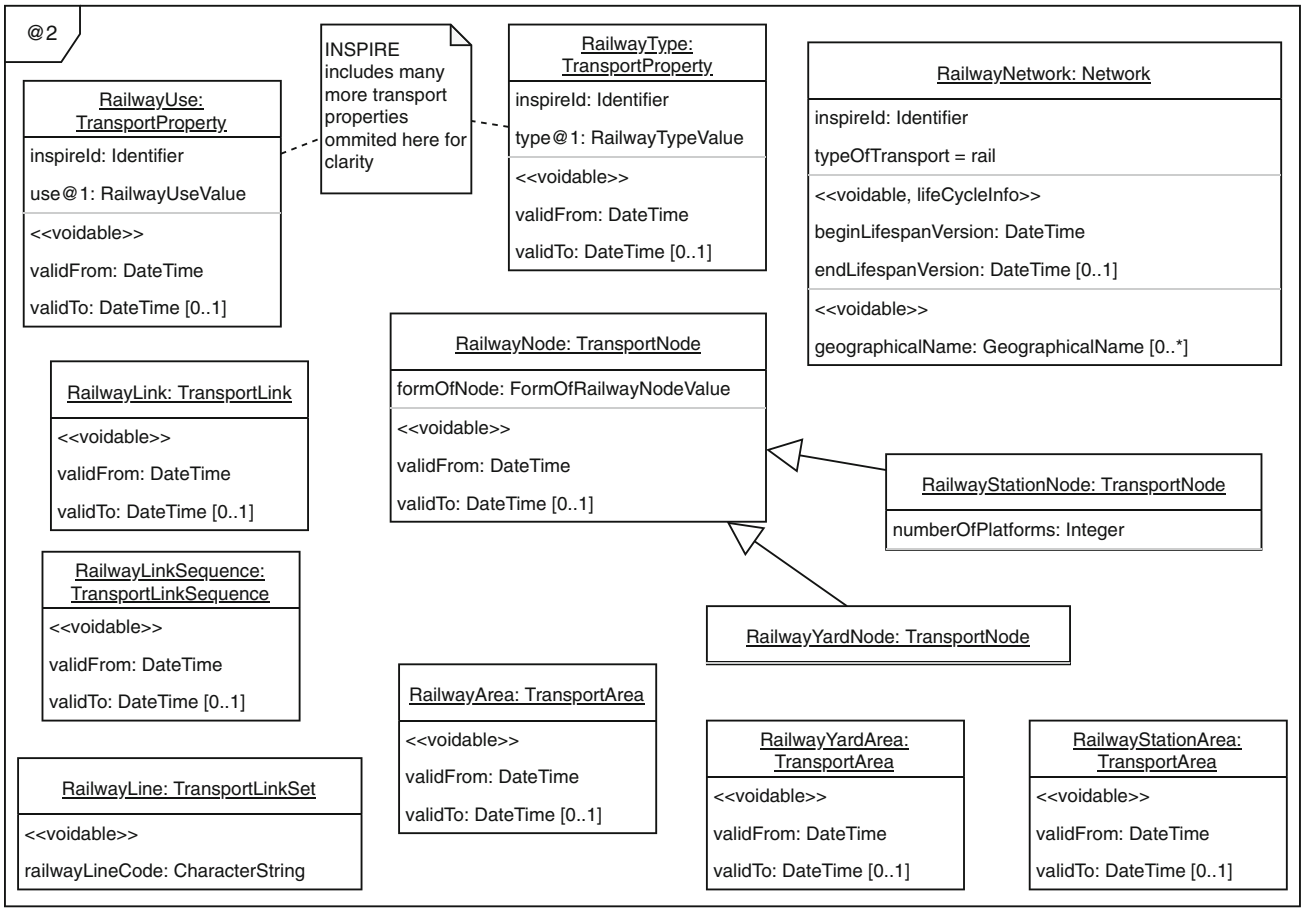

Fig. 21 Modeling multilevel spatial networks—meta-level @2 for railway networks

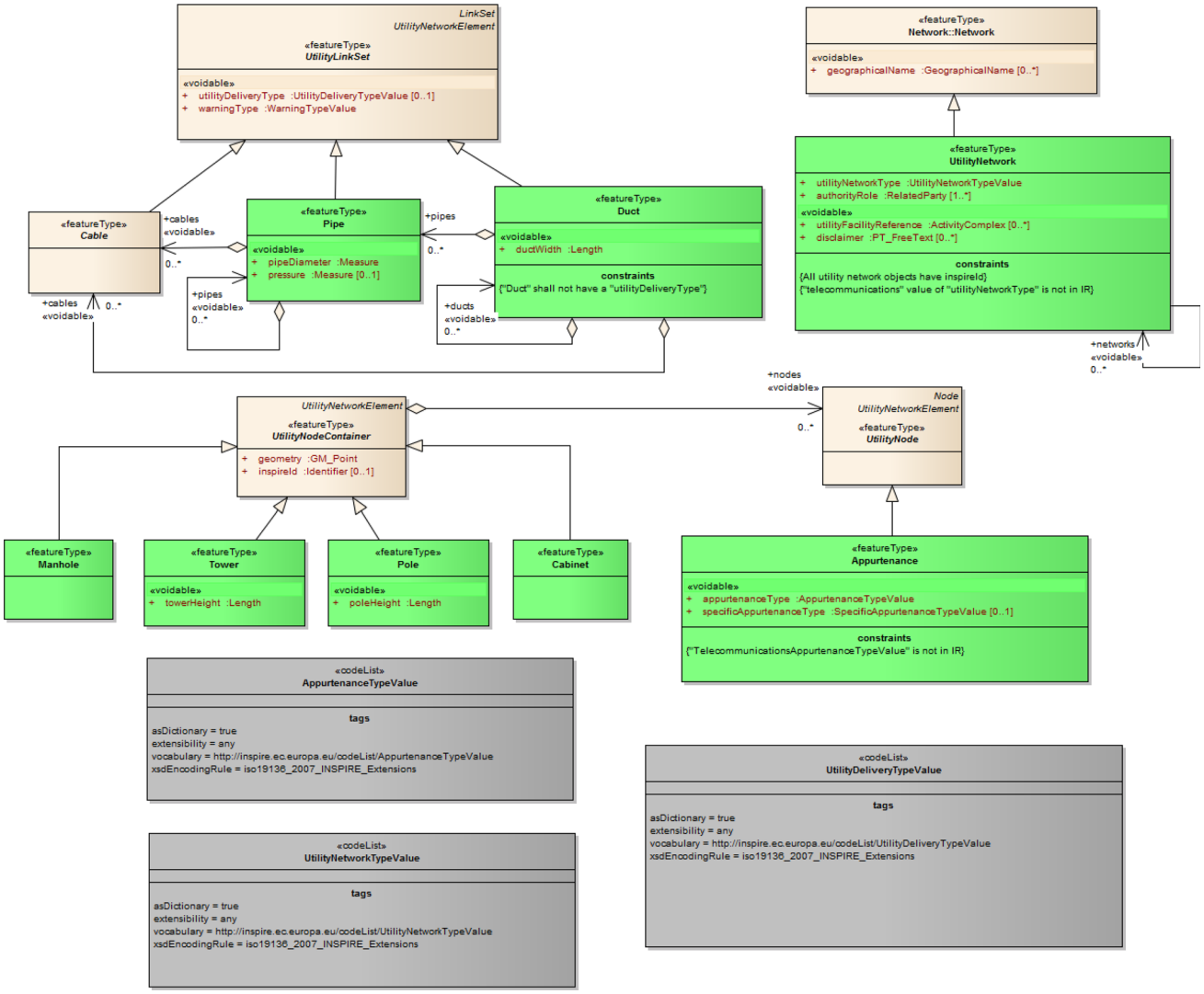

Fig. 22 INSPIRE Common utility network elements, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:3:20:3:1:8887

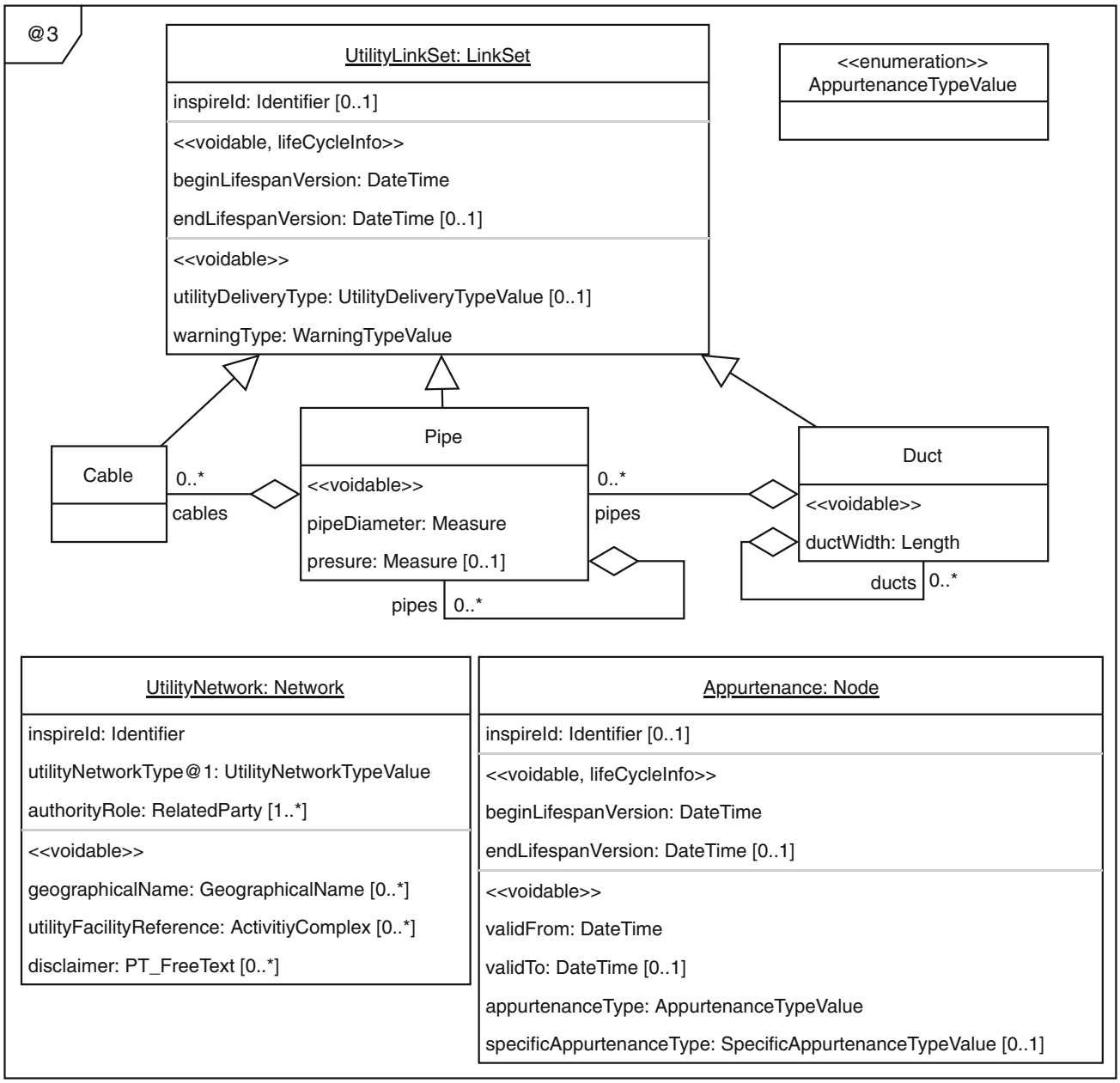

Fig. 23 Modeling multilevel spatial networks—meta-level @3 for utility networks

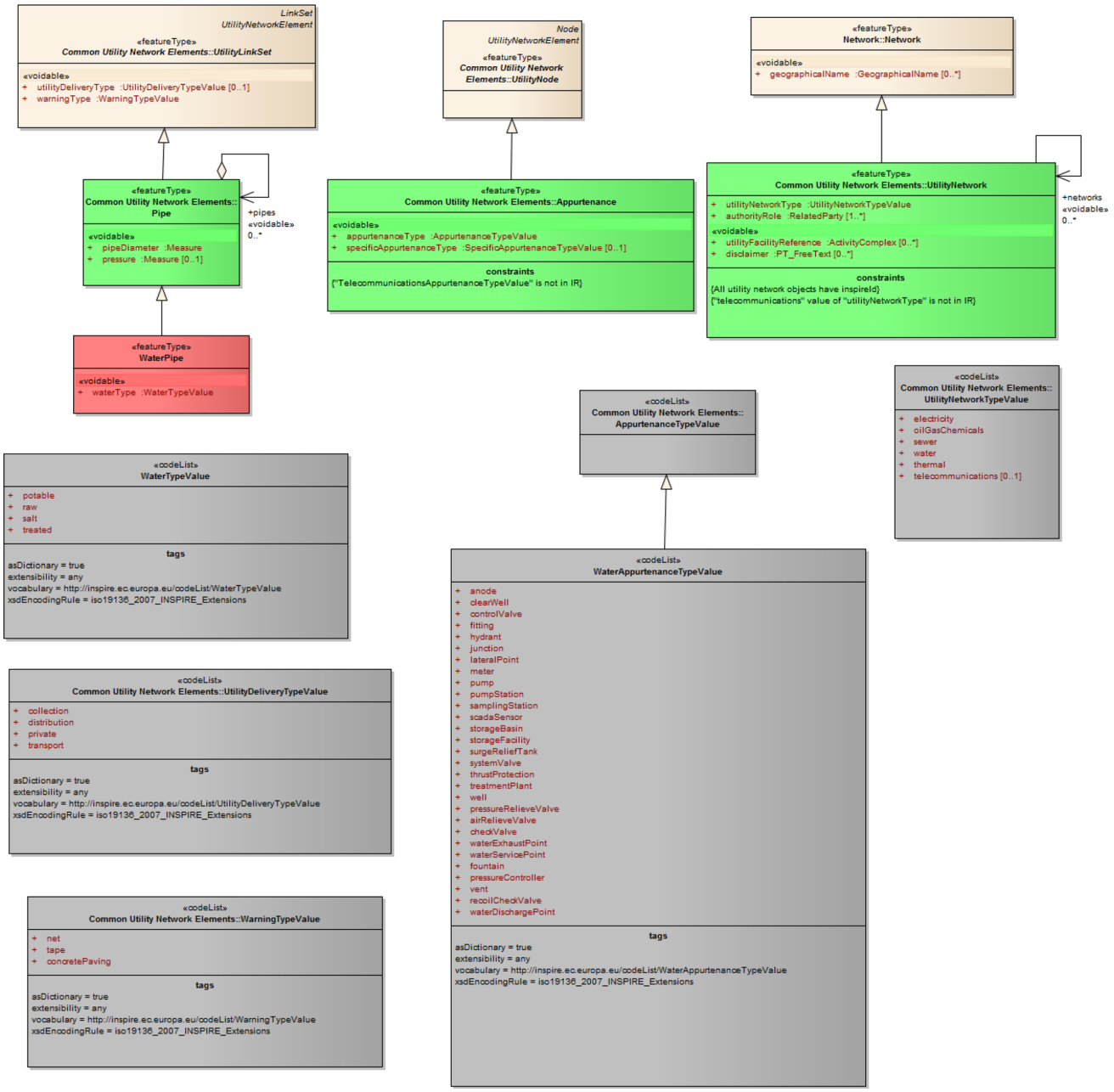

Fig. 24 INSPIRE Water network, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:3:20:3:7:8933

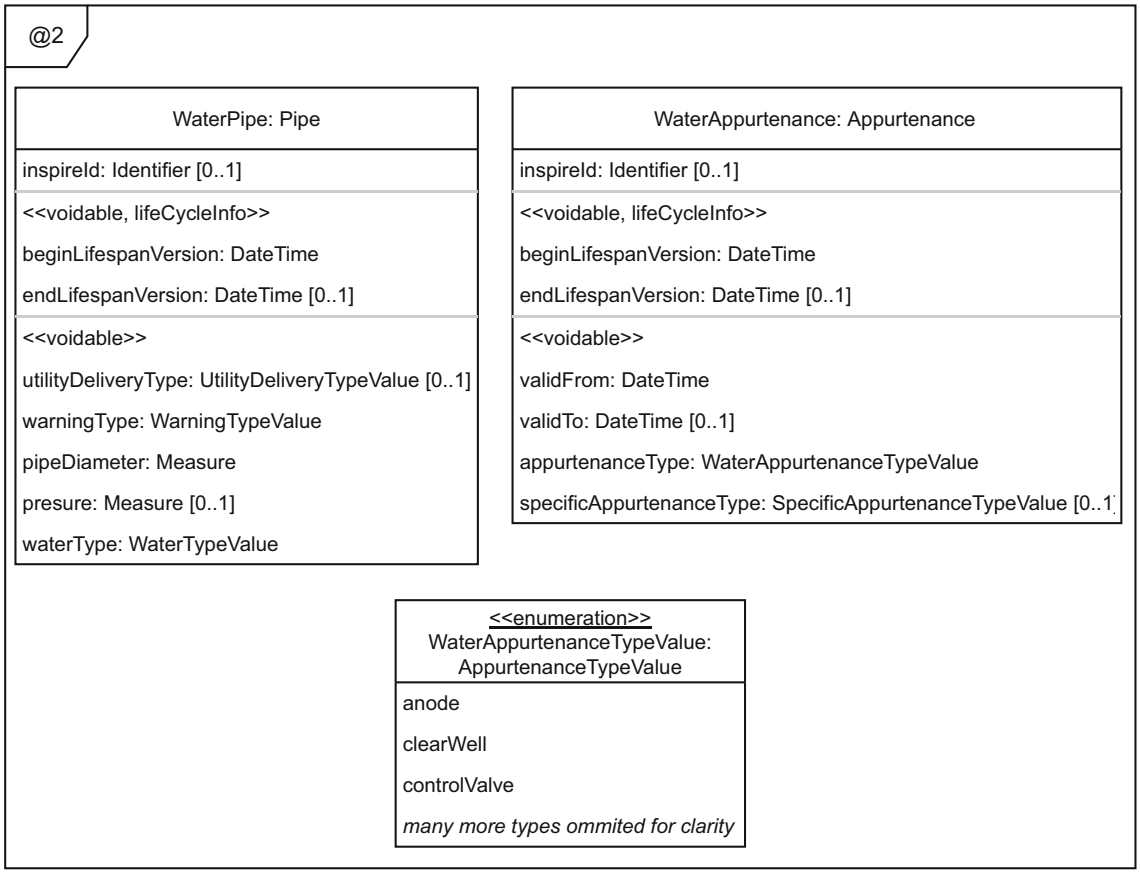

Fig. 25 Modeling multilevel spatial networks—meta-level @2 for water pipes networks

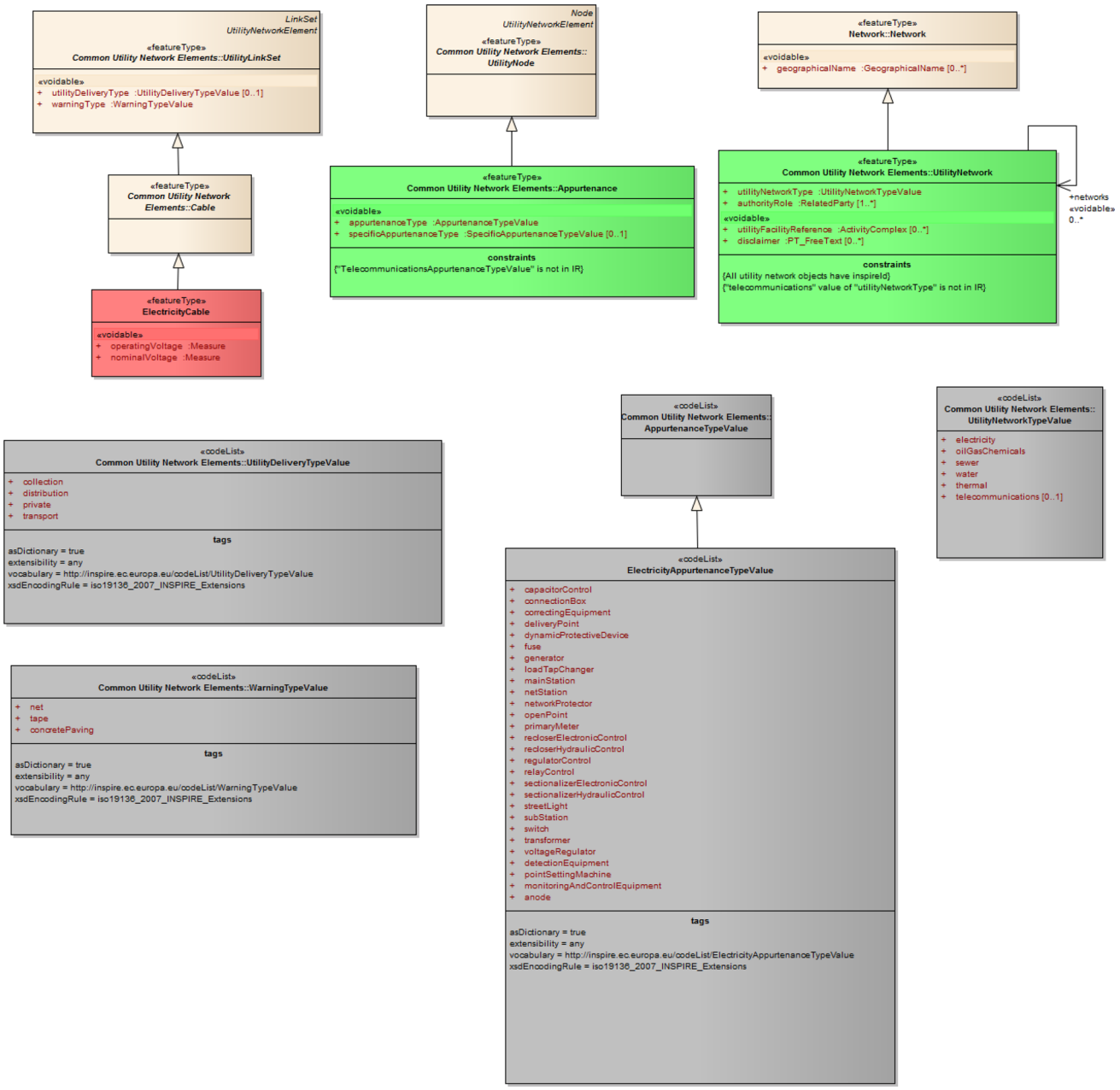

Fig. 26 INSPIRE Electricity network, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:3:20:3:2:8910

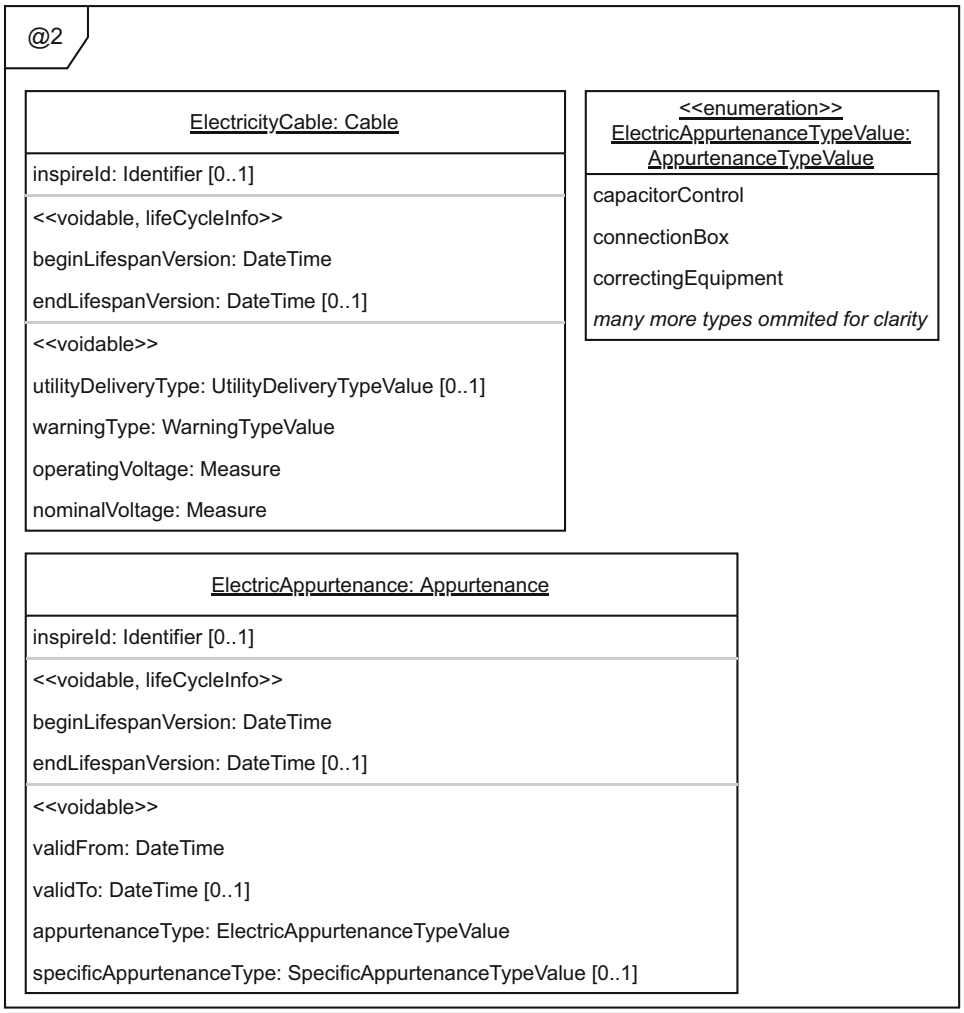

Fig. 27 Modeling multilevel spatial networks—meta-level @2 for electricity networks

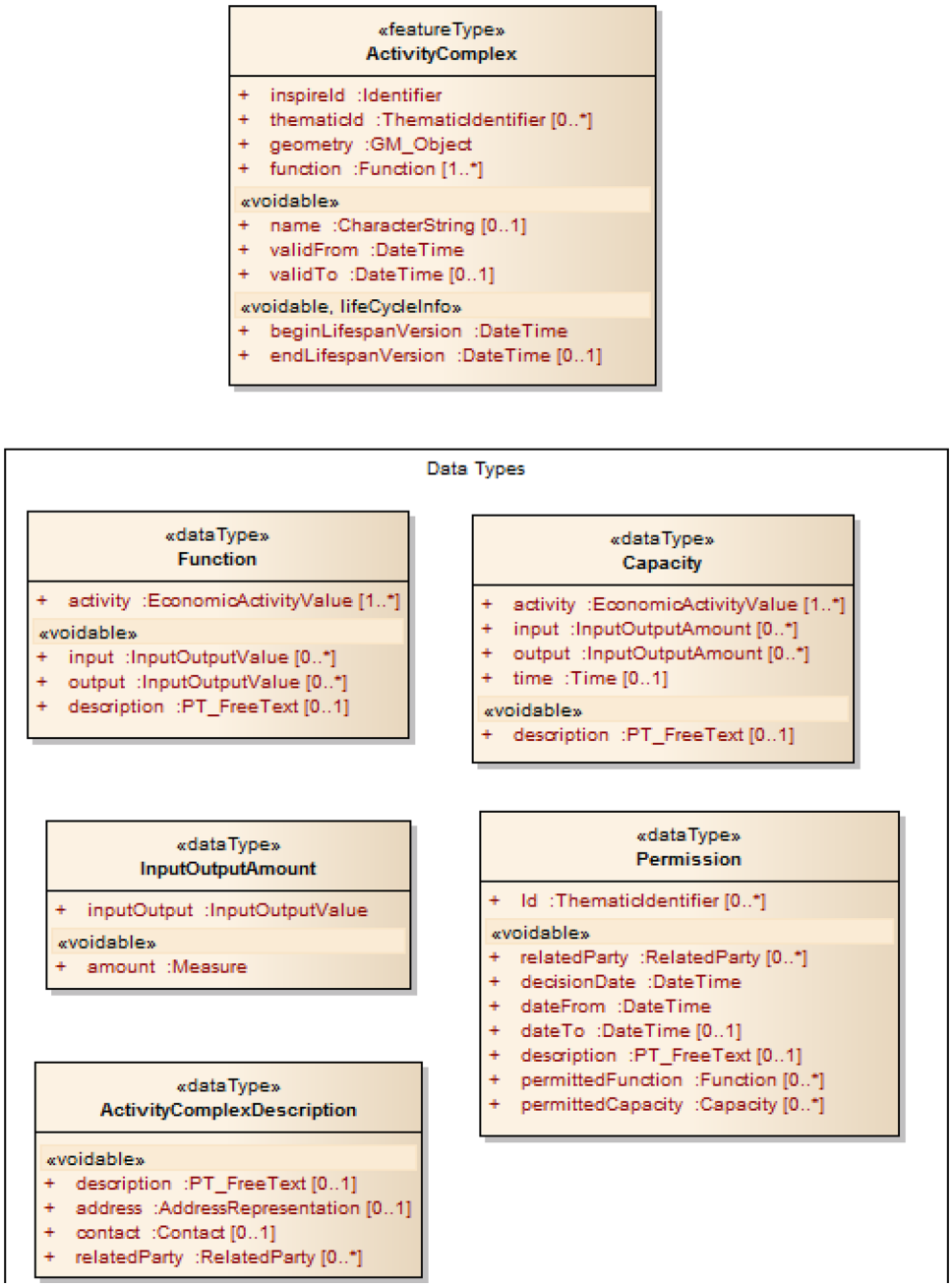

Fig. 28 INSPIRE Activity complex base model, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=3:1:4:1:8990

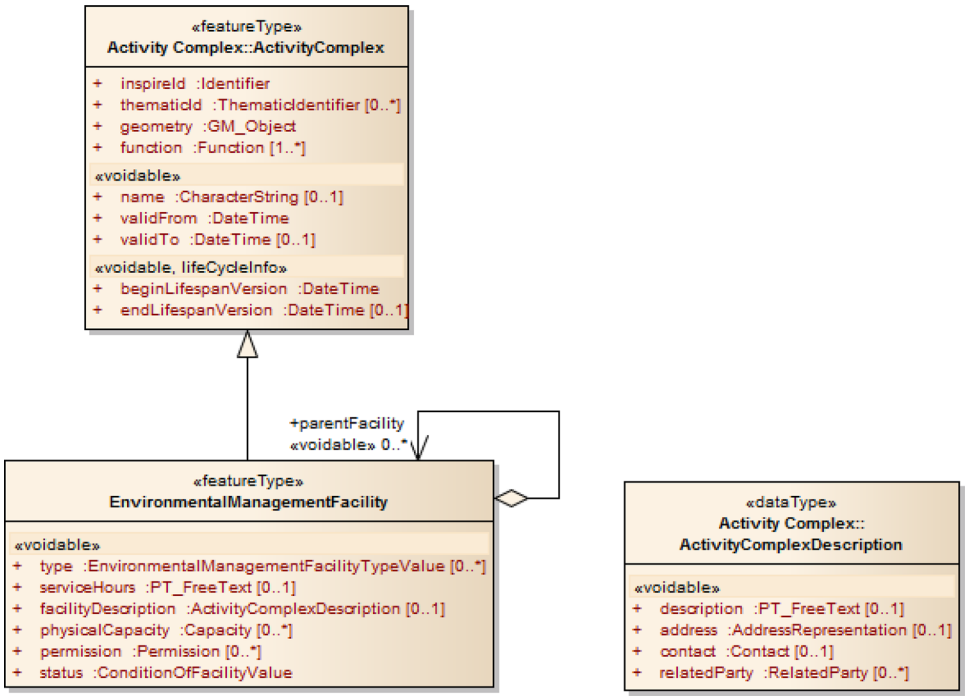

Fig. 29 INSPIRE Environmental management facilities, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:3:20:2:8857

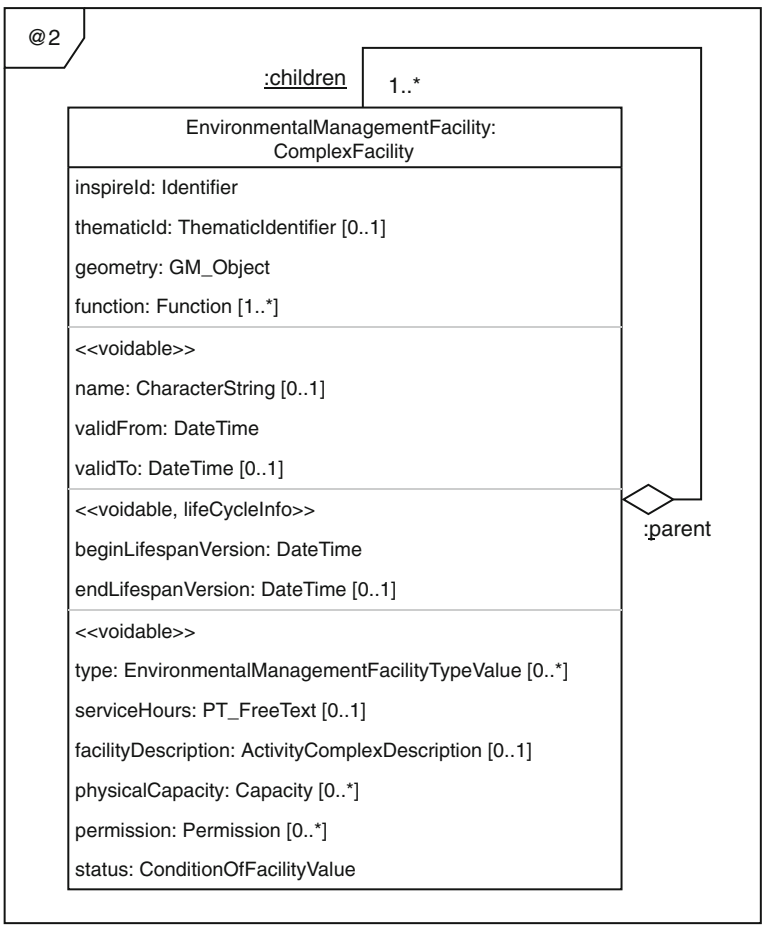

Fig. 30 Modeling multilevel facilities management—meta-level @2 for environmental management facilities

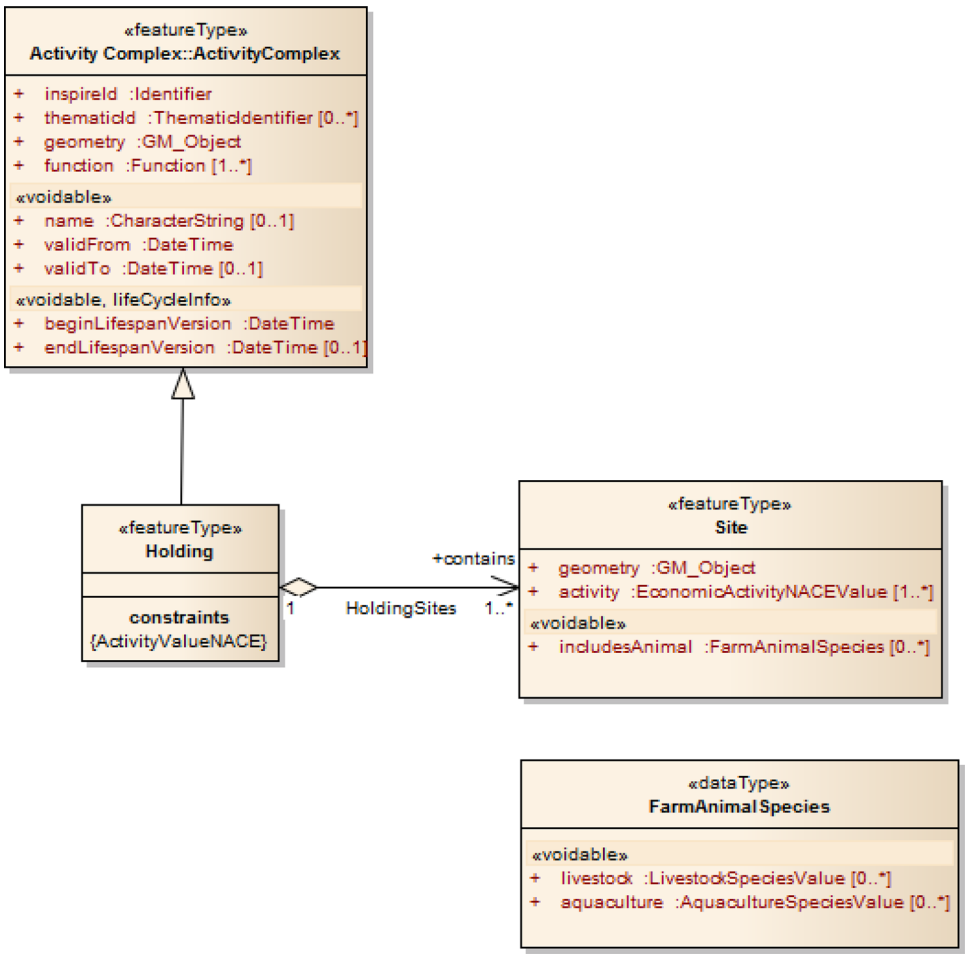

Fig. 31 INSPIRE Agricultural and aquaculture facilities, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:3:3:1:7925

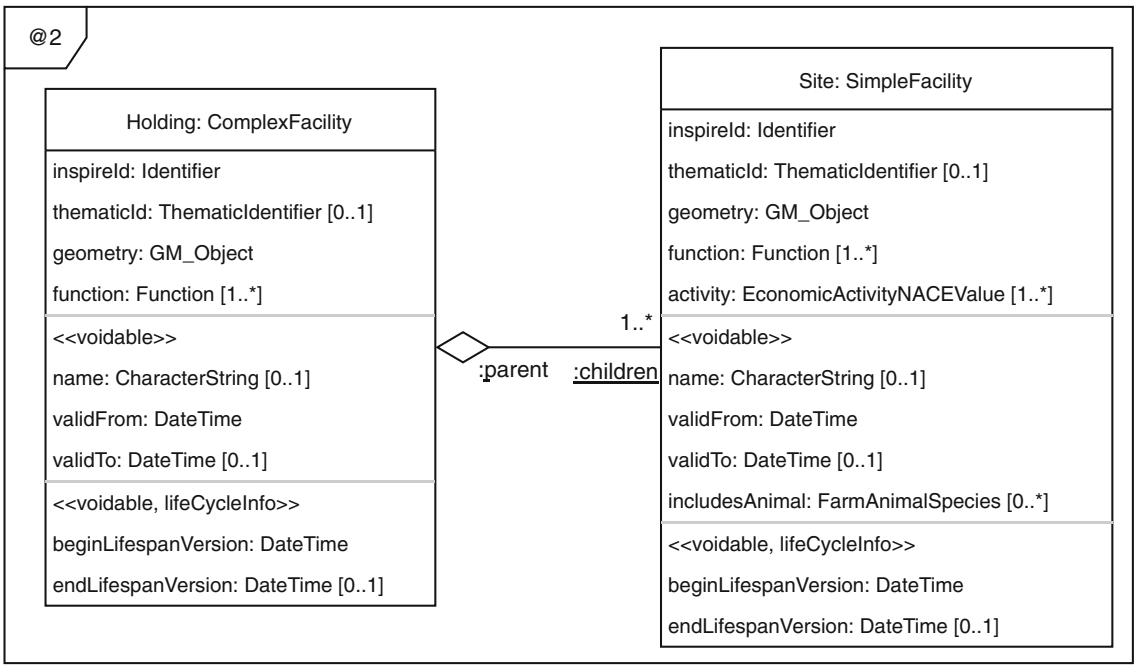

Fig. 32 Modeling multilevel facilities management—meta-level @2 for agricultural facilities

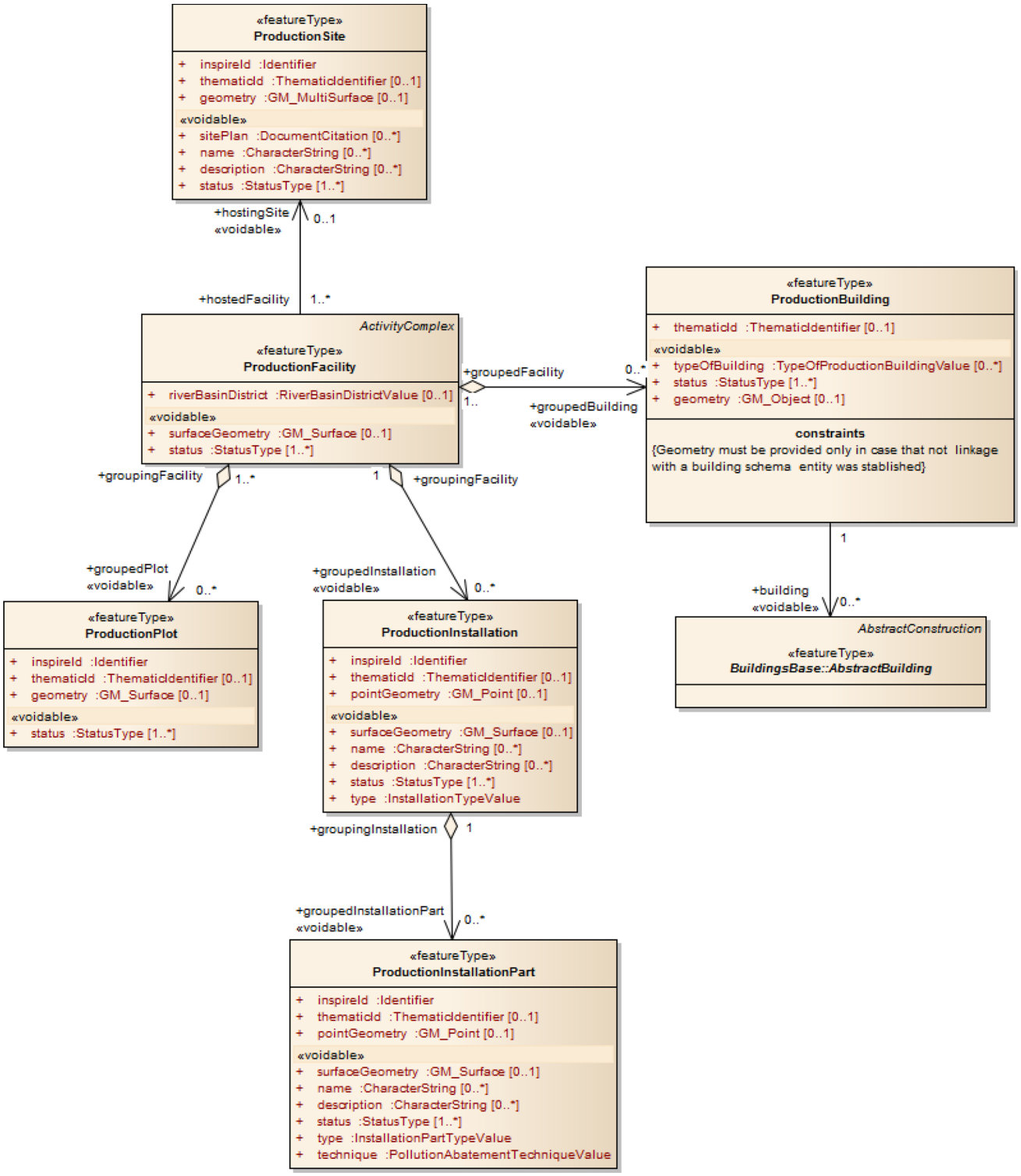

Fig. 33 INSPIRE Production and industrial facilities, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:3:15:1:8641

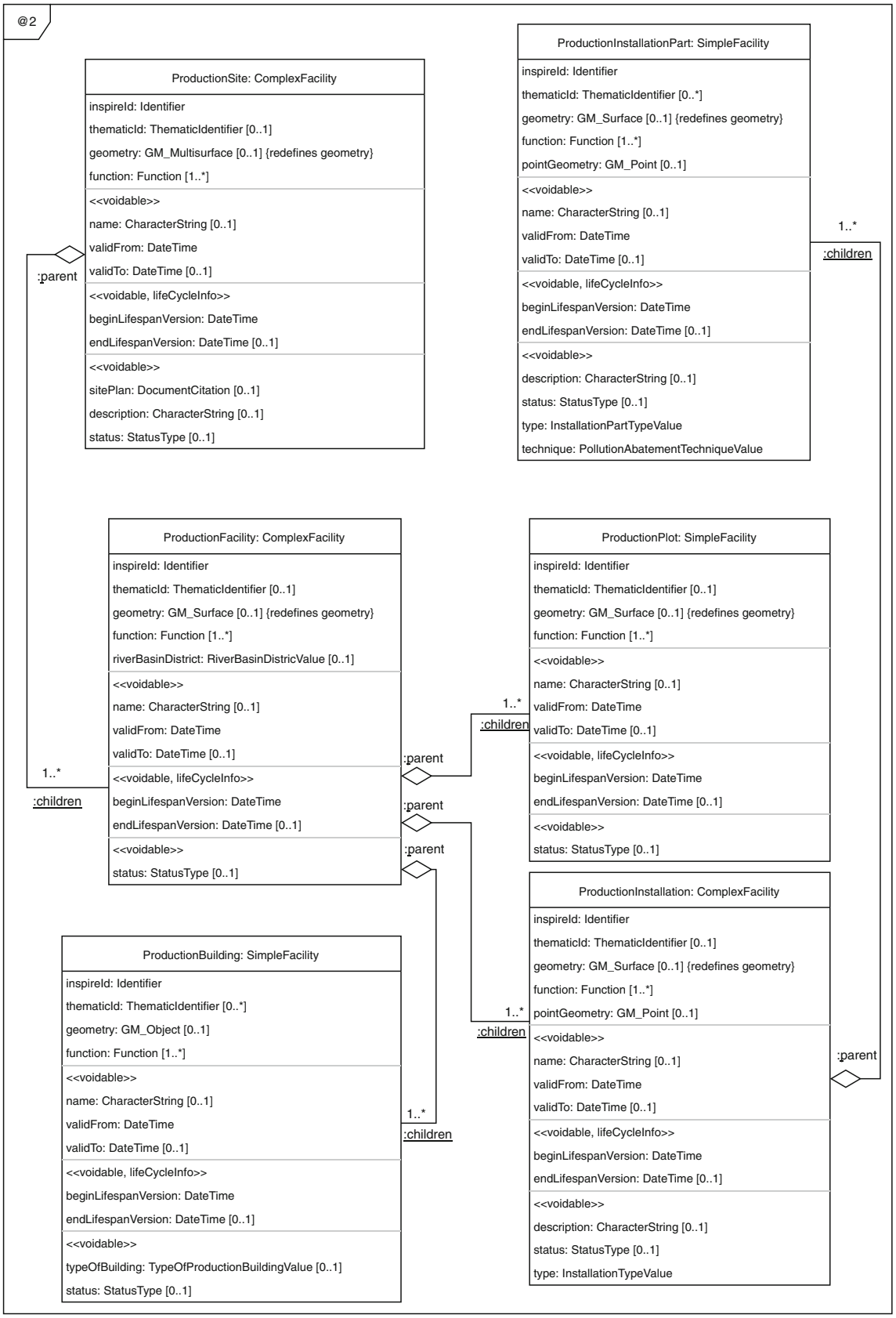

Fig. 34 Modeling multilevel facilities management—meta-level @2 for production and industrial facilities

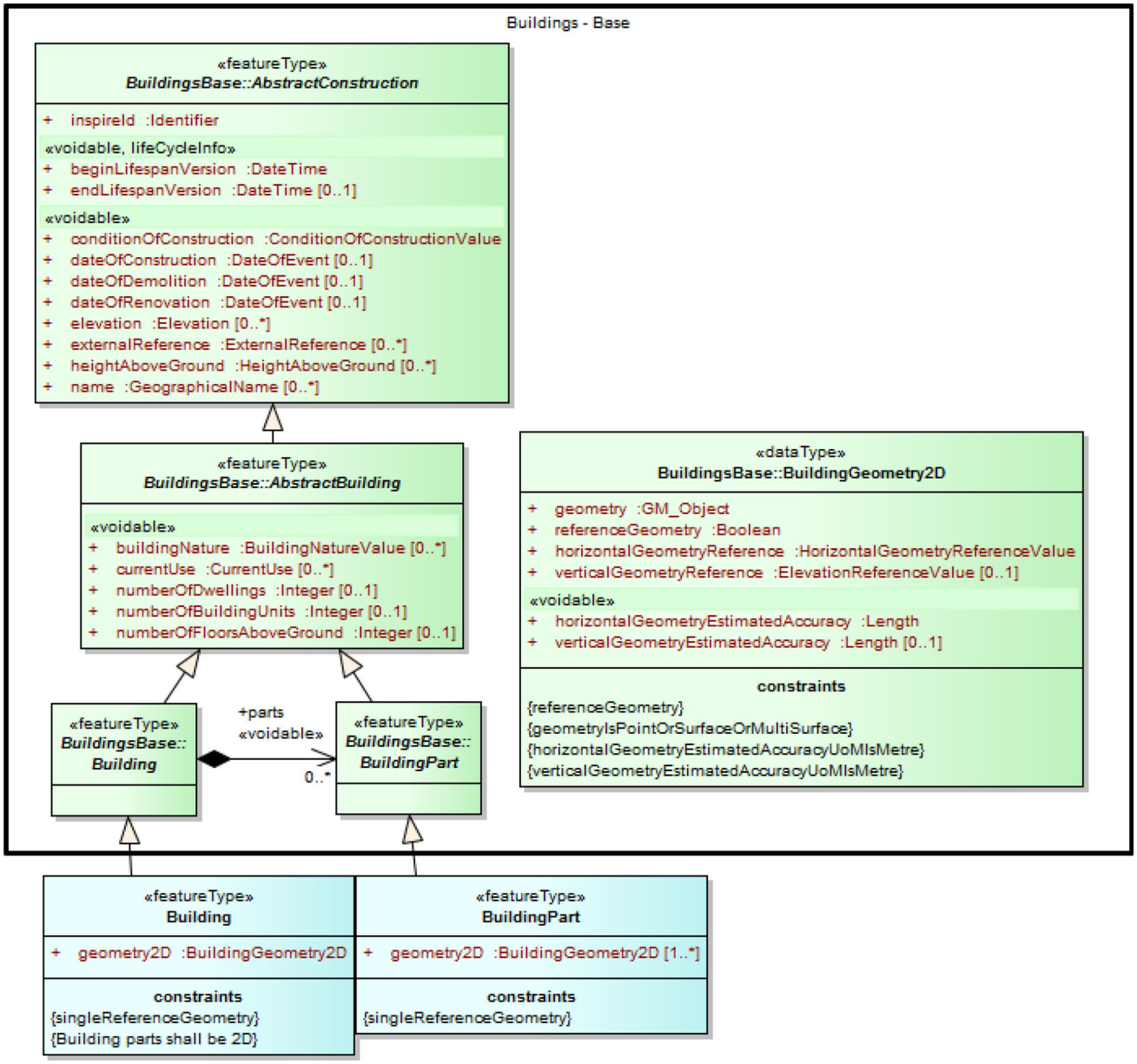

Fig. 35 INSPIRE Buildings base and core 2D, from https://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/index.htm?goto=2:3:2:2:7911

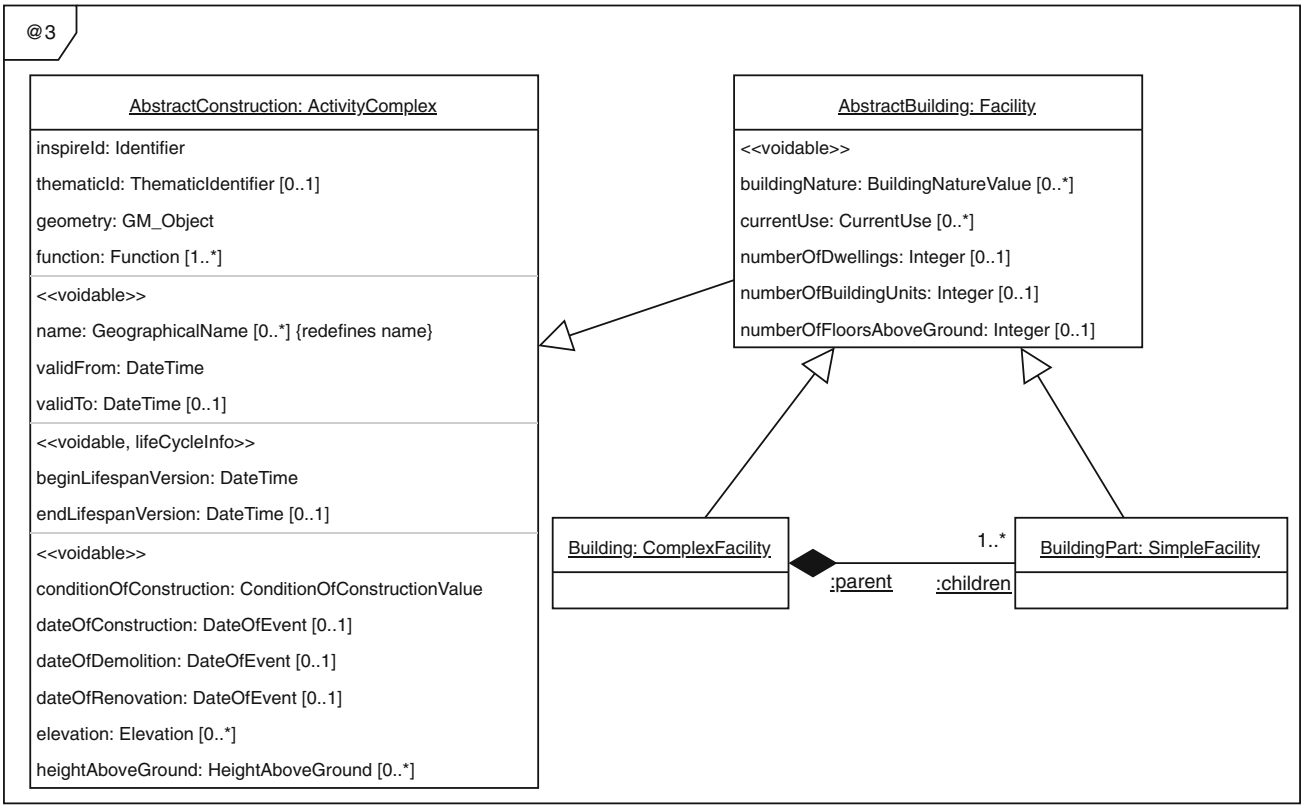

Fig. 36 Modeling multilevel facilities management—meta-level @3 for building facilities

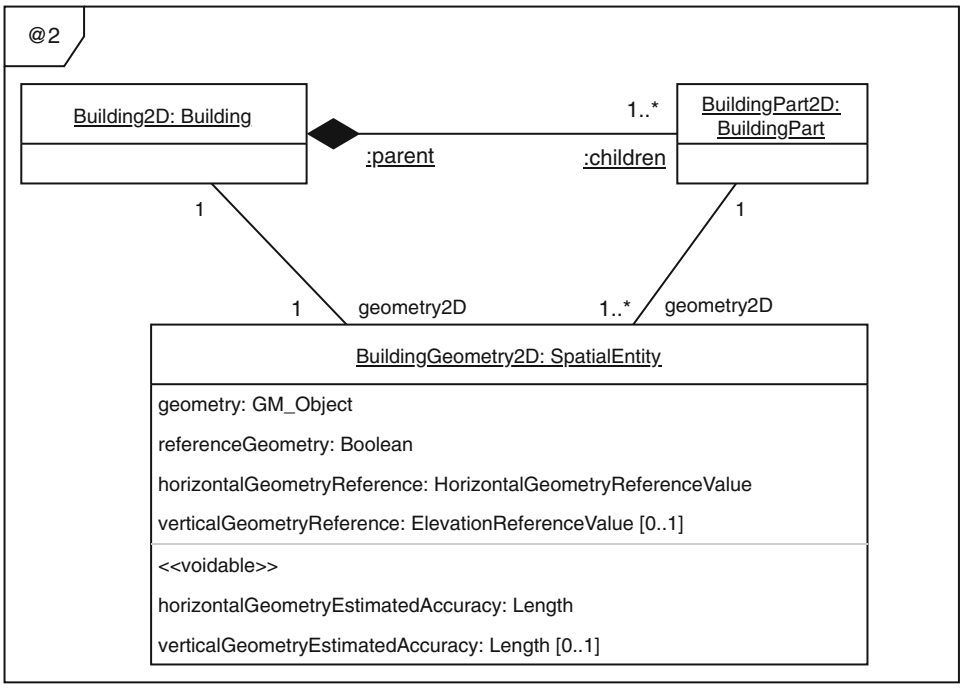

Fig. 37 Modeling multilevel facilities management—meta-level @2 for building facilities

References

1. Worboys, M.F., Duckham, M.: GIS: A Computing Perspective. CRC Press, Boca Raton (2004)
2. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice, 2nd edn. Morgan & Claypool Publishers, California (2017)
3. Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer, Berlin (2007)
4. Cortiñas, A., Luaces, M.R., Pedreira, O., Places, Á.S., Pérez, J.: Web-based geographic information systems SPLE: domain analysis and experience report. In: Proceedings of the 21st International Systems and Software Product Line Conference, (SPLC'2017), pp. 190–194 (2017)
5. Cortiñas, A., Luaces, M.R., Pedreira, O., Places, Á.S.: Scaffolding and in-browser generation of web-based GIS applications in a SPL tool. In: Proceedings of the 21st International Systems and Software Product Line Conference (SPLC'2017), pp. 46–49 (2017)
6. Alvarado, S.H., Cortiñas, A., Luaces, M.R., Pedreira, O., Places, A.S.: Developing web-based geographic information systems with a dsl: proposal and case study. J. Web Eng. **19**, 167–194 (2020)
7. de Lara, J., Guerra, E., Cuadrado, J.S.: When and how to use multilevel modelling. ACM Trans. Softw. Eng. Methodol. **24**(2), 12:1–12:46 (2014)
8. Atkinson, C., Kühne, T.: The essence of multilevel metamodeling. In: International Conference on the Unified Modeling Language, pp. 19–33. Springer (2001). https://doi.org/10.1007/3-540-45441-1_3
9. Atkinson, C.: Meta-modelling for distributed object environments. In: Proceedings First International Enterprise Distributed Object Computing Workshop, pp. 90–101 (1997)
10. Frank, U.: Designing models and systems to support it management: a case for multilevel modeling. In: Proceedings of $2^n d$ International Workshop on Muli-Level modelling (MULTI'16) - MODELS Workshops (2016)
11. Tomlinson, R.: A geographic information system for regional planning. J. Geogr. **78**(1), 45–48 (1969)
12. Atkinson, C., Kuhne, T.: Model-driven development: a metamodeling foundation. IEEE Softw. **20**(5), 36–41 (2003)
13. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. Softw. Syst. Model. **7**(3), 345–359 (2008)
14. Frank, U.: Toward a unified conception of multi-level modelling: advanced requirements. In: Proceedings of the 5th International Workshop on Multi-level Modelling (MULTI'2018), pp. 718–727 (2018)
15. Al-Hilank, S., Jung, M., Kips, D., Husemann, D., Philippsen, M.: Using multi level-modeling techniques for managing mapping information. In: Proceedings of International Workshop on Muli-Level modelling (MULTI'14) - MODELS Workshops (2014)
16. Benner, B.: A multi-level approach for model-based user interface development. In: Proceedings of 4^{th} International Workshop on Muli-Level modelling (MULTI'17) - MODELS Workshops (2017)
17. Nesic, D., Nyberg, M.: Applying multi-level modeling to data integration in product line engineering. In: Proceedings of 4^{th} International Workshop on Muli-Level modelling (MULTI'17) - MODELS Workshops (2017)
18. Rodríguez, A., Rutle, A., Durán, F., Kristensen, L.M., Macías, F.: Multilevel modelling of coloured petri nets. In: Proceedings of 5^{th} International Workshop on Muli-Level modelling (MULTI'18) - MODELS Workshops (2018)
19. Rossi, M.T., De Sanctis, M., Iovino, L., Rutle, A.: A multilevel modelling approach for tourism flows detection. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 103–112. IEEE Press (2019)
20. Lisboa-Filho, J., Sampaio, G.B., Nalon, F.R., de V. Borges, K.A.: A uml profile for conceptual modeling in gis domain. In: Proceedings of DE Workshop at International Conference on Advanced Information Systems Engineering (CAISE 2010), pp. 18–31 (2010)
21. Sampaio, G.B., Nalon, F.R., Filho, J.L.: Geoprofile - UML profile for conceptual modeling of geographic databases. In: Proceedings of the 12^{th} International Conference on Enterprise Information Systems (ICEIS 2010), pp. 409–412 (2010)
22. Filho, J.L., Nalon, F.R., Peixoto, D.A., Sampaio, G.B., de Vasconcelos Borges, K.A.: Domain and model driven geographic database design. In: Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J. (eds.) Domain Engineering, Product Lines, Languages, and Conceptual Models, pp. 375–399. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-36654-3_15
23. Kutzner, T.: Geospatial data modelling and model-driven transformation of geospatial data based on uml profiles. Ph.D. thesis, Technical University of Munich (2016)
24. Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: a literature review. Inf. Syst. **35**(6), 615–636 (2010)
25. de Lara, J., Guerra, E.: Deep meta-modelling with metadepth. In: Vitek, J. (ed.) Objects, Models, Components, Patterns, pp. 1–20. Springer, Berlin (2010)
26. Atkinson, C., Kühne, T.: Rearchitecting the uml infrastructure. ACM Trans. Model. Comput. Simul. **12**(4), 290–321 (2002). https://doi.org/10.1145/643120.643123
27. International Organization for Standardization: Iso 19107:2003 - geographic information: Spatial schema. https://www.iso.org/standard/26012.html. Visited on 2020-06-21
28. The Open Geospatial Consortium: OpenGIS Simple Feature Access - Part 1: Common Architecture. http://www.opengeospatial.org/standards/sfa. Visited on 2020-06-21
29. International Organization for Standardization: Iso 19125:2004 - geographic information "simple feature access" part 1: Common architecture. https://www.iso.org/standard/40114.html. Visited on 2020-06-21
30. Güting, R.H.: Graphdb: Modeling and querying graphs in databases. In: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, pp. 297–308. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1994)

Suilen H. Alvarado is a Ph.D. student in Computer at the University of A Coruña, Spain since 2018. She received an M.Sc. degree in Computer Science from the University of Concepción, Chile in 2018, and a Computer Engineering degree from the University of Matanzas, Cuba in 2012. Her research topics of interest include the application of automated software development techniques in Geographic Information Systems.

Alejandro Cortiñas is an Assistant Professor at the Database Lab of the Universidade da Coruña (Spain). He received a PhD in Computer Science from the same university in 2017 for his thesis, entitled "Software Product Line for web-based Geographic Information Systems". His research topics of interest include software product lines, generative programming, geographic information systems, and spatial big data.

Miguel R. Luaces received his M.S. degree in Computer Science from the University of A Coruña (Spain) in 1998 and an European Ph.D in Computer Science from the University of A Coruña (Spain) in 2004. He undertook research in the area of spatial, temporal and spatio-temporal databases at the FernUniversität Hagen (Germany) under the ChoroChronos project funded by the European Union. Today, he is an Associate Professor at the University of A Coruña, and he is currently a member of the Databases Laboratory of the University of A Coruña where he has been involved successfully in a number of research and development projects. His research interests include Geographic Information Systems, Spatial and Spatio-temporal Databases, Software Engineering, and Web-based Information Systems.

Oscar Pedreira has M.Sc. and Ph.D. degrees in Computer Science from University of A Coruña, Spain. He is an Associate Professor since 2008 at the same institution. He is a researcher of the Database Laboratory. His research interests include topics in databases (algorithms for similarity search, data structures and algorithms for graph databases, geographic information systems), and in software engineering (process improvement, testing, MDE, and SPL). He has co-authored many articles published in journals and conferences relevant for the research areas mentioned. He has continuously participated in research projects and technology and knowledge transfer projects with different companies.

Angeles S. Places is currently an Associate Professor at the Computer Science Department of the University of A Coruña. She received her PhD in Computer Science in 2003 from the same university. Her research interests are in the areas of Digital Humanities, Web Information Systems, Geographic Information Systems and Software Engineering. For further details about her CV see: http://lbd.udc.es/ShowResearcherInformation.do?id=5