Applying Multilevel Modeling to the Development of Geographic Information Systems

Suilen H. Alvarado, Alejandro Cortiñas, Miguel R. Luaces, Oscar Pedreira and Angeles S. Places
Universidade da Coruña, Centro de Investigación CITIC, Database Laboratory
Facultade de Informática, Campus de Elviña s/n, 15071, A Coruña, Spain
Email: {s.hernandez,alejandro.cortinas,luaces,opedreira,asplaces}@udc.es

***Abstract*—Multilevel modeling is an approach to model-driven engineering (MDE) in which the number of metamodel levels is not fixed. In this paper, we present and discuss the application of multilevel software modeling to the development of geographic information systems (GIS), and its potential benefits. Different GIS applications may provide different features and functions, but they all share a set of common concepts (regarding spatial data types, operations, services, etc.), common architecture, and a common set of technologies. Although we do not present a complete set of models, we present representative parts of that set (spatial networks, territory decomposition, and trajectories) that, when compared with previous work on MDE applied to GIS, support our proposal that multilevel modeling can provide more benefits to GIS development than just applying a more traditional two-level MDE approach.**

***Index Terms*—Model-driven engineering, multilevel software modeling, geographic information systems**

I. Introduction

Model-driven engineering (MDE) is an approach to software development in which models play a central and active role, far beyond just describing the system. In MDE, models describe the software system and 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 [1], [2]. A common approach to MDE is based on the OMG's[1] *model-driven architecture* (MDA)[2], which defines four layers: 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.

The traditional approach to MDE considers two-levels. In the most abstract one, a metamodel defines the main concepts of the domain. In 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* [3]–[5]. In contrast to a more "traditional" approach, multilevel modeling does not constrain the number of levels, so the designer could 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 several levels. Multilevel modeling solves some drawbacks and restrictions that can occur in the traditional two-level modeling [6]. As explained in [7], 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.

De Lara et al. present in [6] a research work focused on when and how to use multilevel modeling in software development. 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.

In this paper, we present and discuss the idea of applying multilevel modeling to the development of geographic information systems, and we present a preliminary proposal of a set of models that show, based on comparison with previous works, the advantages that this approach may bring when compared with a two-level approach.

A *geographic information system* (GIS) is an information system in which some or many of the entities it manages have a spatial component that defines them in the space and plays a central role in their processing. GIS are present in many application domains. Without the intention of being exhaustive, the most typical applications include maps and road network applications, transportation and logistics, or territory administration, for example. Although different GIS applications have a different purpose and implement different functionalities, they all share a common set of concepts, such as geometries, coordinate systems, layers, maps, and spatial operations, and they also share a common architecture, and a set of typical structures, as we will see later in this article.

In previous works [8], [9], we have addressed the development of web-based GIS applications applying a combination

Acknowledgements: Partially funded by: Xunta de Galicia/FEDER-UE CSI: ED431G/01 (Centros singulares de investigación Galicia); Xunta de Galicia/FEDER-UE CSI: ED431C 2017/58 (Grupo de Referencia Competitiva); Xunta de Galicia / FEDER-UE, ConectaPeme, GEMA: IN852A 2018/14; MINECO-AEI/FEDER-UE Datos 4.0 (TIN2016-78011-c4-1-R); MINECO-AEI/FEDER-UE ETOME-RDF3D3 (TIN2015-69951-R); MINECO-AEI/FEDER Flatcity (TIN2016-77158-C4-3-R); MICINN-AEI/FEDER-UE BIZDEVOPS: (RTI2018-098309-B-C32).

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

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

of the software product line engineering (SPLE) and MDE approaches. From the perspective of MDE, we followed a two-level approach, that is, we defined a very generic metamodel with the most common elements in a GIS, which allowed us to create a model that was finally transformed into the code of a working system. That previous experience has allowed us to reformulate the development of GIS systems with MDE following a multilevel approach.

The rest of the article is structured as follows: In Section II we present background and related work, including a brief description of the main elements of a GIS, and a summary of previous work on applying a two-level MDE approach for their development. In Section III we present our proposal for developing GIS under a multilevel modeling approach. Although we do not present a complete set of metamodels and models of the domain, we present the most abstract levels and three examples of lower levels that show the benefits that can be obtained with this approach. Finally, Section IV presents the conclusions of the paper, the work we are currently undertaking, and lines for future work.

II. Related Work

A. A brief description GIS development

Geographic information systems is a field focused on information systems with geospatial characteristics and capabilities that allow us to represent and manipulate knowledge regarding geographical information [10]. GIS were traditionally used by some organizations, such as public institutions, for territorial administration, but since the major advances happened in communication technologies, more and more companies and organizations in many fields and domains are adopting GIS solutions to improve their workflows. Some fields in which using a GIS solution seems mandatory nowadays are, for example, warehouse logistics, in order to plan the routes in the most efficient way; public transportation, to know which lines are overused or underused, and to decide how to change them accordingly; or even social networks, 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 afterwards is used by ad services.

Regardless of the area of application or the purpose of each GIS, these systems share most of the features among them, such as digitizing geographic data, viewing geolocated data in map viewers, the common tools related to maps (from panning and zooming to measuring dimensions or objects within the map, or sorting the different layers), route calculation, etc. Even more, GIS have been around for some time [11] and during the first decades each GIS development was totally independent of the rest, and even when the functional features provided by the systems were quite similar, the concepts behind GIS, such as the definition of what is a *polygon*, were different among several systems. But nowadays this situation has changed thanks to two organizations, ISO/TC 211[3] and OGC[4], which have defined a set of evolving standards related to all levels of GIS. Therefore, nowadays GIS not only share the functional features but also the models, procedures, services, and even the architecture.

For example, the standard *ISO 19107: Geographic Information - Spatial Schema* [12] defines all the geographic data types, such as *Point*, *Line*, *Polygon*, or *Sphere*, and all the GIS related operations, such as the predicates *intersects*, *overlaps*, or *within a specified distance*. OGC also defines a set of web services that are widely used and supported by map servers and map viewers. Some of the most important ones are *web map service* [13] or *web feature service* [14]. Regarding architecture, the standard *ISO 19119: Geographic Information - Services* [15] identifies architecture patterns for service interfaces and the relationship between them, proposes a geographic services architecture, and provides some guidelines for the selection and specification of geographic services.

These standards are followed by most GIS software assets and therefore most GIS applications are quite similar. In the case of web-based GIS applications, the similarities affect even to the specific software assets used, since most of them are widely popular such as GeoServer, a map server that provides most of the standard services, or OpenLayers and Leaflet, two map viewer libraries.

Even with the use of common software assets to build similar applications regarding features, GIS applications are usually developed "from scratch" with the help of these tools and libraries, which means low productivity, long time-to-market, and high costs, specially 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.

B. An MDE+SPLE approach for GIS

In [8], [9] 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.

Software product lines engineering is a field that pursues the industrialization in software development by applying the same processes that were carried out in the factories industrialization, such as reusing components and assembling specific products from a selection of features from the whole set supported. This whole set of features supported by a product family is represented by feature models [16]. The first step in order to apply SPLE techniques to GIS is to analyze different GIS products and extract the set of possible features of these products, classifying these features into common features or variability. Each one of these features has to be implemented with a software asset or component. In fact, most of the features related to processing geographic information, such as *importing a shapefile*[5] or *geocoding postal addresses*, are implemented in encapsulated components that provide

[3]International Organization for Standardization committee on Geographic information/Geomatics: https://committee.iso.org/home/tc211

[4]The Open Geospatial Consortium: http://www.opengeospatial.org/

[5]Shapefile reference: https://doc.arcgis.com/en/arcgis-online/reference/shapefiles.htm

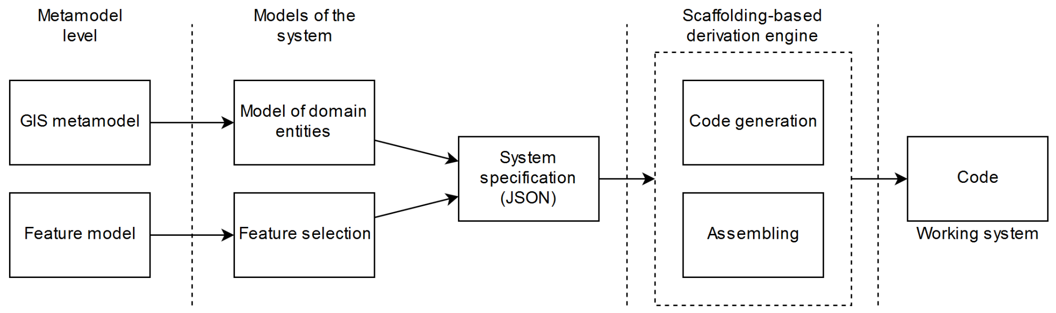

Fig. 1. Tool architecture

interfaces and that can be used "as they are" independently of the domain or the particular product, maybe with small variations such as *supported file formats* or *using Google Maps geocoding*[6] *or OpenStreetMaps Nominatim*[7]. Therefore, from a specification of the features to be included in the system, we can build the application combining and configuring the components related to those features.

Although the features and components are common to the family of GIS products, there is one caveat: the data model depends on the particular domain and, even when there are some models that appear frequently, it is required that each product is defined by means of the data it should support with total flexibility. The only difference between the data model in GIS and other applications is that in GIS we need geographic data types.

Generating code from models was addressed with model-driven engineering, so we defined a metamodel to specify how the data model of the products of our family is described.

Figure 1 shows the architecture of our tool for the automatic generation of GIS using SPLE and MDE. In the metamodel level, our design combines two metamodels: the first one defines the entities of the GIS domain, allowing to model entities with a geographic component and the relations between them; the second one is a feature model [16] which contains the features that we can select for a specific product and the set of constraints between them. The metamodel considers the definition of entities with their attributes and relationships, with the particularity that spatial data types can be used. That is, the MDE part of the platform presented in [8], [9] allows the designer to create models with entities using spatial data types. In order to generate a product, these metamodels are instantiated into models, defining both the data model and the selection of features of a particular system. These two models combined are the system specification, which in our case is represented by a JSON document. This specification is processed by the derivation and code generation engine that finally generates the source code of a working system.

Our tool, and the metamodels it handles are very flexible and complete, and they allow us to define GIS applications for any domain. For example, if we need to develop a product handling public resources, we can define entities representing hospitals and other kinds of medical centers, education buildings such as universities or schools, the public bus transportation network, the sections of a water supply network, etc. However, this design following the two-level modeling approach has some caveats. First, it would force us to define all possible elements of a GIS in a single metamodel. This would not allow us to reflect in the model or set of models of the system common situations in this domain, such as defining entities that refine other entities at higher levels of abstraction. Second, since GISs 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 allow us to take any advantage of this scenario.

As we will explain in Section III, these disadvantages can be addressed by applying a multilevel approach. 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 proposal shows that applying a multilevel approach allows us to metamodel, at an intermediate level, the most common GIS structures, so we can use these structures to make simpler models in the lower level.

III. Developing GIS with a Multilevel Modelling Approach

In this section, we present a proposal for the development of GIS under a multilevel modeling approach. We do not present a complete set of metamodels of this domain, but three representative scenarios within the domain of GIS-based applications that illustrate the benefits that could be obtained from this solution in real developments: spatial networks, territory administration, and trajectories.

De Lara et al., in [6], have identified a series of metamodeling patterns that benefit when multilevel metamodeling is used. Some of these patterns appear in the examples we present next, such as the *Type-Object Pattern* [17], which explicitly models both types and instances to allow adding dynamically new types; the *Relation-Configurator Pattern*, which allows

[6]Google Maps geocoding: https://developers.google.com/maps/documentation/geocoding/intro

[7]OSM Nominatim: https://nominatim.openstreetmap.org/

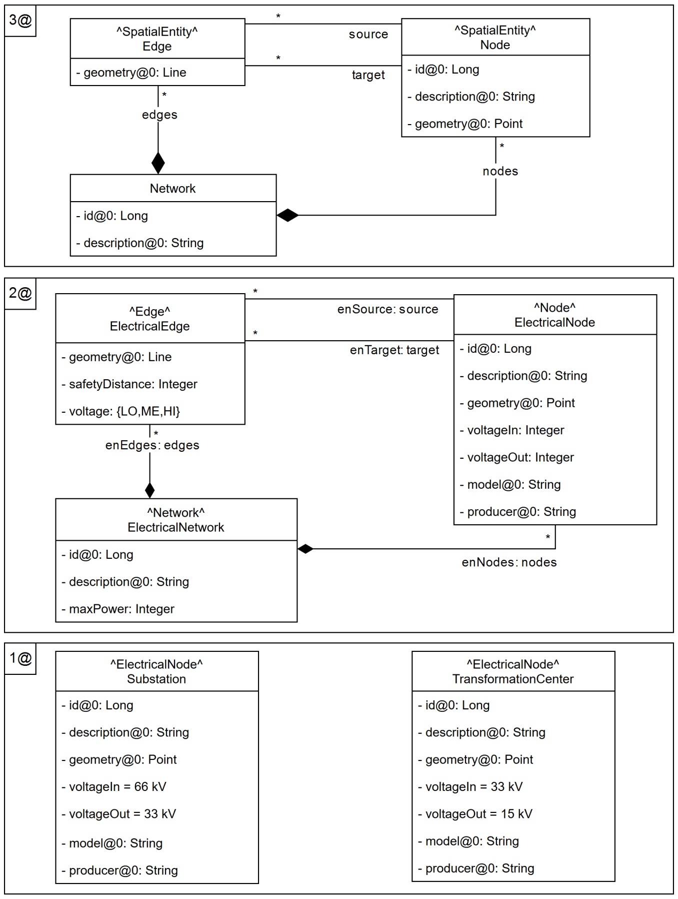

Fig. 2. Modeling multilevel Networks.

defining types of relationships that can be instantiated at lower levels of the model according to the configuration required; or the *Dynamic features* pattern [18], which allows dynamically adding characteristics to an instance of a certain type.

A. Spatial Networks

Spatial networks are a fundamental structure in GIS. Most domains require modeling and processing different types of networks. The most common are transport networks, such as those representing roads, railways, or flight *highways*, and resource distribution networks, such as electrical supply, telecommunications, or water supply networks.

From the most abstract point of view, a network is composed of nodes and edges. Each node has at least an identifier and a location, which is represented using a `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` geometry (a line is not just a straight line, but a sequence of connected segments).

Although both nodes and edges can have different attributes for networks of a different nature (for example, roads vs. electrical supply), the structure of the networks is always the same. In the two-level MDE solution for GIS presented in [8], [9], the designer could define both road and electricity networks by defining the classes composing those network structures and their attributes, but we would have to repeat the network structure for each of the networks we would need.

Figure 2 shows a multilevel solution for modeling networks, with four levels. For reasons of space, level @4 is not shown in the figure, as it only contains a metaclass *SpatialEntity* with an attribute *geometry* to be instantiated in level @0. The meta-level @3 is used to model the basic structure of any network: `Nodes` are spatial entities for which the geometry is a point. In addition, all nodes must have an identifier and a description. `Edges` are spatial entities too, but their geometry is a line. In addition, edges have two associated nodes (source and target). All these attributes are instantiated in level @0. These classes define a generic structure for a network, independently of the nature of either the nodes or edges.

The meta-level @2 shows the model of an specific type of network, an *Electric 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 with 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 level @0.

B. Trajectories

A growing trend in many GIS applications is the representation and analysis of trajectory data. The trajectories are geometric places through which a moving body transits. There are several scenarios in the real world in which it is required to represent different types of trajectories. Some examples are hurricane trajectories, maritime trends, or animal migration monitoring.

In Figure 3, we present the set of models to represent trajectory types. At the meta-level @2, we model a trajectory as a meta-class that is composed of a set of sections that use a `Geometry` specific type, a `Line`. In the next level, we show the models of two possible situations where a trajectory needs to be store.

C. Territory Administration

One of the scenarios of GIS-based applications is the management of data on the territorial and urban order. In this context, one of the representations is the geographic delimitation of spaces and territories, both rural and urban.

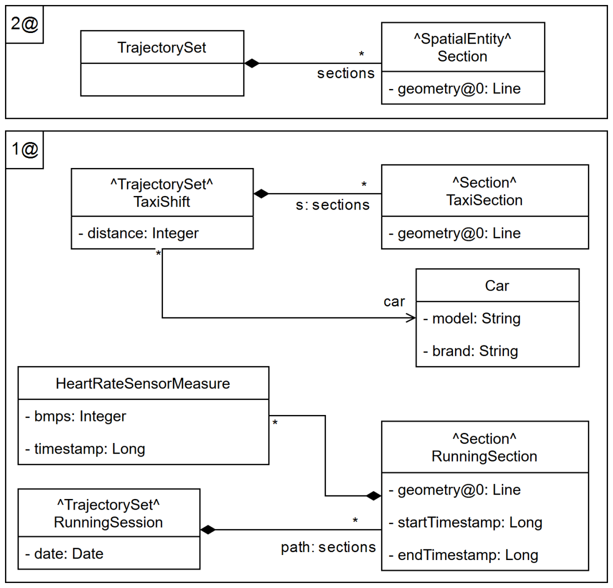

Fig. 3. Modeling multilevel Trajectory.

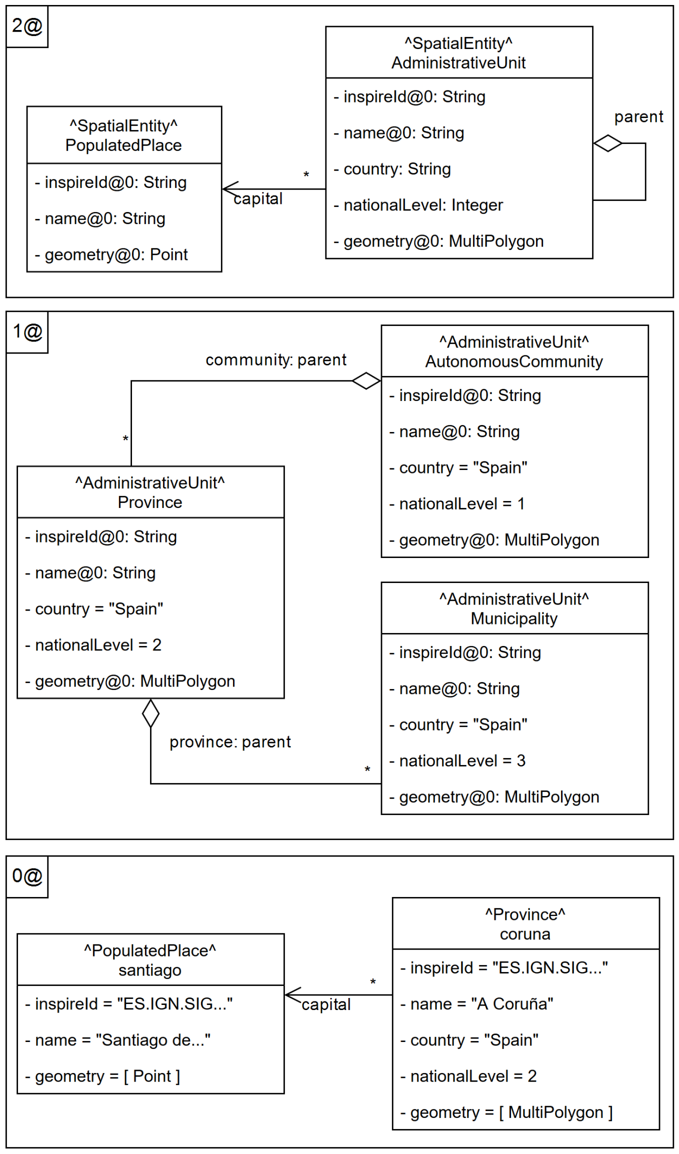

Fig. 4. Modeling multilevel Administrative Unit.

Typically, the administration of the territory structures the space into administrative units, that can be composed of other administrative units they contain, and so on. For example, a country is usually divided into states or autonomous regions, which can be further divided into counties or provinces, which can be divided into municipalities, which can be divided into even smaller territory units. Depending on the domain needs, different territory divisions may be necessary. For example, municipal territory plans in Spain structure the space into regions with different building permissions, which are further divided into properties. As it happened in the case of the spatial networks, all territory decompositions have the same structure, but in different applications, or even within the same application, different territory decomposition schemes may be needed.

In Figure 4 we present the models on this domain. In level @2, we define the meta-classes for a general decomposition of territory. An *AdministrativeUnit* has attributes *id*, *name*, *country*, *nationalLevel*, and *geometry*, and can aggregate other administrative units. A *PopulatedPlace* represents a city or village (with attributes *id*, *name*, and *geometry*). Each administrative unit will have its capital in a populated place. The meta-classes in level @1 define specific territory decomposition units for the particular case of Spain. As we can see in the model, the meta-classes instantiate some of the attributes of *AdministrativeUnit*, such as *country* and *nationalLevel*. At level @0, we show some of the instances of the models defined.

IV. Conclusions and Future Work

In this paper, we have addressed the modeling of Geographic Information Systems following a multi-level approach. The set of models we presented does not pretend to be complete, but focuses on three scenarios which we think show cases in which the modeling of GIS can benefit from multi-modeling. In the three examples we explained how typical structures present in many GIS applications can be moved to more abstract levels to later be refined and instantiated in lower levels. In previous works we had already considered the application of MDE to the domain of GIS, but following a two-level approach. In [8], [9] we presented a platform that allows the designer to define a GIS from its data model, by specifying its entities, their properties, allowing the use of standard spatial data types, and the relationships between the entities. Based on that previous experience, and although the set of models presented in this paper is just a part of all the potential elements included in a GIS, we consider that it shows that the application of multi-modeling in this domain can led to simpler, more flexible, and more expressive solutions when compared with a two-level approach.

A promising line for future work is given by the INSPIRE directive. The European Parliament and of 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). One of the results of INSPIRE are technical guidelines (called Data Specifications[8]) that specify common data models to achieve interoperability of spatial data sets and services across Europe. As an example, INSPIRE defines a generic application schema for networks[9] that provides basic types that are supposed to be extended in other data specifications. Particularly, the data specification for transport networks[10], which covers road networks, rail networks, water networks, air transport, and cableways networks, is defined using the basic types described in the generic application schema for networks. Therefore, this is a clear example of a problem where multilevel modeling is well-suited. Considering that INSPIRE defines data specifications for 34 themes, the usefulness of multilevel modeling is clear.

We are currently completing the set of meta-models/models we presented in this article to come up with a more complete solution. The models presented in this paper were not created following a formal modeling language since their purpose was discussing the potential benefits of multilevel modeling in this domain.

References

[1] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice: Second Edition*. Morgan & Claypool Publishers, Mar. 2017.

[2] O. Pastor and J. C. Molina, *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*. Springer, 2007.

[3] C. Atkinson and T. Kühne, "The Essence of Multilevel Metamodeling," in *International Conference on the Unified Modeling Language*. Springer, 2001, pp. 19–33.

[4] C. Atkinson and T. Kuhne, "Model-driven development: a metamodeling foundation," *IEEE Software*, vol. 20, no. 5, Sep. 2003.

[5] C. Atkinson and T. Kühne, "Reducing accidental complexity in domain models," *Software & Systems Modeling*, vol. 7, no. 3, Jul. 2008.

[6] J. de Lara, E. Guerra, and J. S. Cuadrado, "When and How to Use Multilevel Modelling," *ACM Trans. Softw. Eng. Methodol.*, vol. 24, no. 2, pp. 12:1–12:46, Dec. 2014.

[7] U. Frank, "Toward a unified conception of multi-level modelling: advanced requirements," in *Procs. of the 5th International Workshop on Multi-level Modelling (MULTI'2018)*, 2018, pp. 718–727.

[8] A. Cortiñas, M. R. Luaces, O. Pedreira, Á. S. Places, and J. Pérez, "Web-based geographic information systems SPLE: domain analysis and experience report," in *Procs. of the 21st International Systems and Software Product Line Conference, (SPLC'2017)*, 2017, pp. 190–194.

[9] A. Cortiñas, M. R. Luaces, O. Pedreira, and Á. S. Places, "Scaffolding and in-browser generation of web-based GIS applications in a SPL tool," in *Procs. of the 21st International Systems and Software Product Line Conference (SPLC'2017)*, 2017, pp. 46–49.

[10] M. F. Worboys and M. Duckham, *GIS: a computing perspective*. CRC press, 2004.

[11] R. Tomlinson, "A geographic information system for regional planning," *Journal of Geography*, vol. 78, no. 1, pp. 45–48, 1969.

[12] International Organization for Standardization, "Iso 19107:2003 - geographic information: Spatial schema," https://www.iso.org/standard/26012.html, visited on 2019-07-02.

[13] The Open Geospatial Consortium, "OpenGIS Web Map Server Implementation Specification," http://www.opengeospatial.org/standards/wms, visited on 2019-07-02.

[14] ——, "OpenGIS Web Feature Service 2.0 Interface Standard," http://www.opengeospatial.org/standards/wfs, visited on 2019-07-02.

[15] International Organization for Standardization, "Iso 19119:2016 - geographic information: Services," https://www.iso.org/standard/59221.html, visited on 2019-07-02.

[16] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, and Y. Bontemps, "Generic semantics of feature diagrams," *Computer Networks*, vol. 51, no. 2, pp. 456–479, 2007.

[17] R. C. Martin, D. Riehle, and F. Buschmann, Eds., *Pattern Languages of Program Design 3*. Boston, MA, USA: Addison-Wesley, 1997.

[18] D. Riehle, M. Tilman, and R. Johnson, *Pattern Languages of Program Design 5*. Addison-Wesley, 2005, ch. Dynamic object model.

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

[9]https://inspire.ec.europa.eu/documents/inspire-data-specifications-%E2%80%93-base-models-%E2%80%93-generic-network-model

[10]https://inspire.ec.europa.eu/Themes/115/2892