KJENSTAD, K., 2006, On the integration of object-based models and field-based models in GIS.

This is an electronic version of an article published in the *International Journal of Geographical Information Science*, Vol. 20, No. 5, May 2006, 491–509, available online at: http://www.informaworld.com/smpp/content~content=a747921911

On the integration of object-based models and field-based models in GIS

Abstract

This paper proposes a common base-model for the classical object-based and field-based conceptual models in GIS. The model which is called the PGOModel or "Parameterised Geographic Object Model" is given formal definition by using the UML modeling language. Within the scope of the paper it has been shown that the PGOModel encompasses the classical object-based and field-based models. Two extensive examples demonstrate the application of the PGO model. The PGOModel seems ontologically well founded except for the so-called PGOAtom concept.

1 Introduction

Over the last 10-15 years the *object-based* and *field-based* conceptual modeling approach has gradually generalized the old time vector and raster data structure modeling approach in GIS. This change of approach has had an important impact on the development of Geographic Information Science from an exotic tool to be used by map producers and geographers into an integrated and specialized part of Information Science. However, for a long time there has been an increasing demand for further integration of the object-based and field-based models (Bian 2000, Cova & Goodchild 2002, Egenhofer et al. 1999, Peuquet 1988, Peuquet et al. 1999, Winter 1998). These early works have focused on describing the *co-existence* and *interactions* of the two models. The present paper discusses *integration* of the two models in terms of searching for a common *base-model*.

The *object* and *field* concepts as such predate their explicit use in a GIS context (Angel & Hyman 1976, Sachs 1973, Tobler 1978). They were introduced in GIS in the early nineties (Burrough 1996, Couclelis 1992, Frank 1992, 1996, Goodchild 1989, 1992, Kemp 1997) and subsequently included in textbooks (Burrough & McDonnell 1998, Worboys 1995). This process was triggered in part by the introduction of the concept of *object-orientation* into GIS (Worboys 1994, Egenhofer & Frank 1987, 1992, Frank & Egenhofer 1992, Oosterom & Vanderbos 1989, Worboys et al. 1990).

The *object-based model* has been used as a means of conceptual structuring of geographic information, in particular in the modeling of real world objects (or entities) with a precise and "crisp" spatial location and extent. However, with the introduction of object-oriented methods and models (Rumbaugh et al. 1991), object-based modeling has been given a more prominent formal foundation, for instance in modeling tools and languages like UML (*Unified Modeling Language*) (Rumbaugh et al. 1998).

The *field-based model* concept originates from classical physics and has been used for modeling physical properties (e.g. gravity) whose magnitude is dependent on its spatial location. In Geographic Information Science, the *field* concept has been extended to include any (physical or non-physical) properties whose magnitude (*value*) is dependent on its spatial location. This definition has turned the field into a useful generalization of "field-like" spatial data structures such as TIN models and raster models.

The present paper is investigating the possibility for the object-based and field-based model being both derived from a common model using a UML modeling approach. This type of generalized source model for derived models is called a *base-model*. The criterion for a model being a common base-model for two given models is that both of them are proven to be special cases of the base-model. We are *not* necessarily searching for *the* base-model, but rather *a* base-model, as there may in theory be several possible base-models. The motivation for the search for a common base-model is the potential of revealing additional characteristics of the relationship between the object-based and field-based models. Our conceptualization of the geographic space will benefit from a more integrated view of the two models, and every aspect of this integration will contribute. As an additional bonus, a common base-model has the potential of being useful by its own.

A *model* is a simplified description of the structural and behavioral patterns for a subset of the reality. In GIS the actual subset of the reality is restricted to entities and properties having well-defined location in the geographic space. Moreover, models can be *refined* according to our actual needs. This refinement process defines models at different modeling *levels*, each with well-defined common characteristics. The *conceptual model* level, the *data model* level, the *implementation model* level, and the *storage model* level represent examples of such model levels. This paper is primarily restricted to models on the *conceptual* level. In other words, we will focus on modeling as a tool for describing and understanding our real world, and at this stage we are less concerned about the realization of the models in different types of information systems. The latter may potentially be the topic for further research.

Object-based models on the conceptual level are sometimes called *entity-based* models because they are focusing on modeling real world entities. However, in this paper is focusing on the representation on such models in a language (i.e. UML) where the term *object* is crucial. Hence, the term *object* is preferred over *entity* when characterizing the type of model as well as its representation (i.e. *object-based*). However, the class representing the generic *entity* in the object-based model will be called *Entity* in order to avoid too much confusion with the concept of *object* as a UML modeling element. Furthermore, the UML modeling language distinguishes between the concepts of *object class* (as the model element) and *object instance* (as its physical realization). We will subsequently use the terms *class* and *instance* for these concepts.

A specific modeling process is restricted to concepts from a well-defined application domain. This paper however, describes conceptual models in general, acting as hosts for application domain models. Such models are called *pattern-models* and they acts as a common base-model for a certain type of application domain models. For instance, the pattern model for the classical object-based model in GIS acts as a base-model for GIS object-based models in general, and the pattern model for the classical field-based model in GIS acts as a base-model for GIS field-based models in general.

The model to base-model relationship can be studied formally in the framework of a *meta-model*. A meta-model is formally defined as a *model of a model*. Modeling element such as classes and relations of any model are instances of meta-model classes with well-defined relations. In this framework the model to base-model relationship is established and proven formally by deriving all modeling elements and concepts of the former from meta-model classes which inherits meta-model classes deriving all modeling elements and concepts of the latter. This approach is complex and hard to understand because it involves a mixture of modeling elements at several levels of abstractions.

A simpler and more intuitive approach is to compare every part of the two models directly and investigate if *all* subsets of a model (for instance the subset consisting of a class including all its relations to other classes) are specializations of a well-defined subset of the base-model. Proving this "specialization" relationship includes comparing the properties of the class including the type and cardinality of its immediate relations. This approach is highly simplified by using the UML modeling approach as a graphical tool, because UML depicts the model as a visible graphical pattern.

The model to base-model relationship between each of the two classical pattern-models and their common base-model could easily use this approach because all three models involved may be explicitly formulated in UML. This is the approach used in the subsequent sections of this paper. However, the model to base-model relationship between any GIS application domain model and one of the two classical pattern-models is more difficult to justify because we are not able to demonstrate this relationship for all possible application domain models. This means that this particular relationship cannot be proven formally, but must be justified by professional experience and motivating examples. This means that the conclusions of this paper relay on the acceptance of the "correctness" of the pattern-models. This type of axiomatically based logical approach is still valid as a scientific method.

2 The search for a candidate base-model

The search for a candidate base-model requires a formal description of the object-based and field-based pattern models. These descriptions will be formalized using standard UML. They will contribute to our search for similarities and differences between the two models. We will begin by extending each of the pattern-models and subsequently attempt to associate similar concepts in the two models. A candidate base-model can then be suggested and described using UML formalism. The proof of this candidate model being a common base-model can subsequently be based on the comparison of the different UML-models involved.

Class names will be capitalized and typed in italic typeface in the subsequent description of UML models. Furthermore, the UML term *relation* will be used when describing formal class relationships and their name will also be capitalized and typed in italic typeface.

2.1 Formal description of the object-based model

Shaded classes and bold line relations in fig. 1 depicts a UML diagram of a pattern-model for a standard object-based model. The diagram tells us that the class called *Entity* is an

aggregation of the class called *Attribute*. Real life object-based models include other concepts such as the concept of *relation* or *methods* of entities, but we do not need to include them at the current stage.

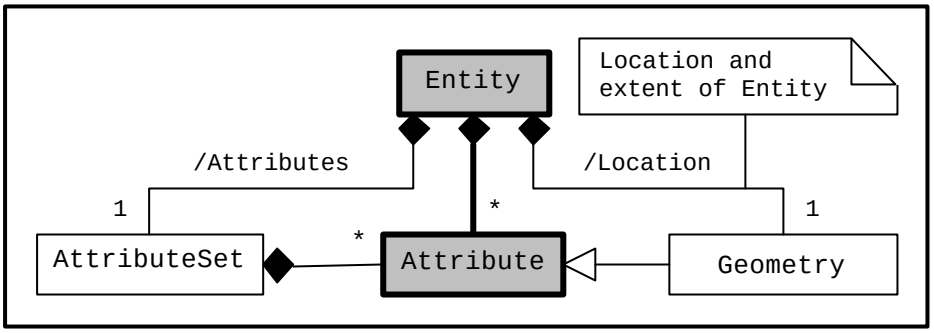

Fig. 1: Conceptual UML class diagram of a standard object-based model. Shaded classes and bold line relations represent the core object-based model, and the remainder represents extensions made in this paper.

2.2 Extension of the core object-based model

Furthermore, fig. 1 describes a proposed extension of the UML model. Two trivial and redundant helper classes have been added and the basic aggregation relation has been supplemented by two *derived* aggregation relations (represented formally by a "/"-prefix in UML). The class called *AttributeSet* represents the aggregation of all instances of class *Attribute* related to one instance of class *Entity*. Consequently, *Entity* aggregates exactly one *AttributeSet*. The class called *Geometry* is modeled as a sub-class of class *Attribute* because entities in GIS have a well-defined location formally described by an attribute of class *Geometry*, or more precisely, by one of its sub-classes. It must be emphasized that these helper classes are solely introduced in order to facilitate model pattern matching in the subsequent sections.

2.3 Formal description of the field-based model

Field models in GIS define a field as a *value* varying over a subset of the geographical space. A *geometry* bounds this subset. Hence, a field in GIS can be defined as a value varying over a geometry. In mathematical terms a field is a *function* from the *geographical space* (defined as the *domain* of the function), to a *value space* (defined as the *range* of the function). The concepts of *domain* and *range* will be defined mathematically in subsequent sections.

Let us formulate the field definition in an object-oriented framework using UML. This approach is also proposed by Bian (2000). Shaded classes and bold line relations in fig. 2 depicts a UML diagram of a pattern-model for a standard field-based model. The class called *Field* aggregates exactly one instance of a class called *Geometry* defining its *domain*, and exactly one instance of a class called *Values* defining its *range*. The *Geometry* class is furthermore defined as an aggregation of its atomic parts represented by a class called *Location*. The term *location* is preferred over *point* because the term *Point* is traditionally used when representing the 0-dimensionality *Geometry* sub-type. This aggregation relation is

of infinite cardinality because geometries normally consist of an infinite number of locations. Similarly, the *Values* class is an aggregation of objects of a class called *Value*. The relation between the *Location* and *Value* classes describes the basic property of a field, i.e. the many-to-one association of a unique value to each location. Hence, the relation between the *Geometry* and *Values* classes is a *derived* relation representing the previously mentioned *FieldFunction*. This model leads to the initial basic definition of a *Field*: *"A field is defined as an UML class representing a function from a Euclidian space to a value space"*.

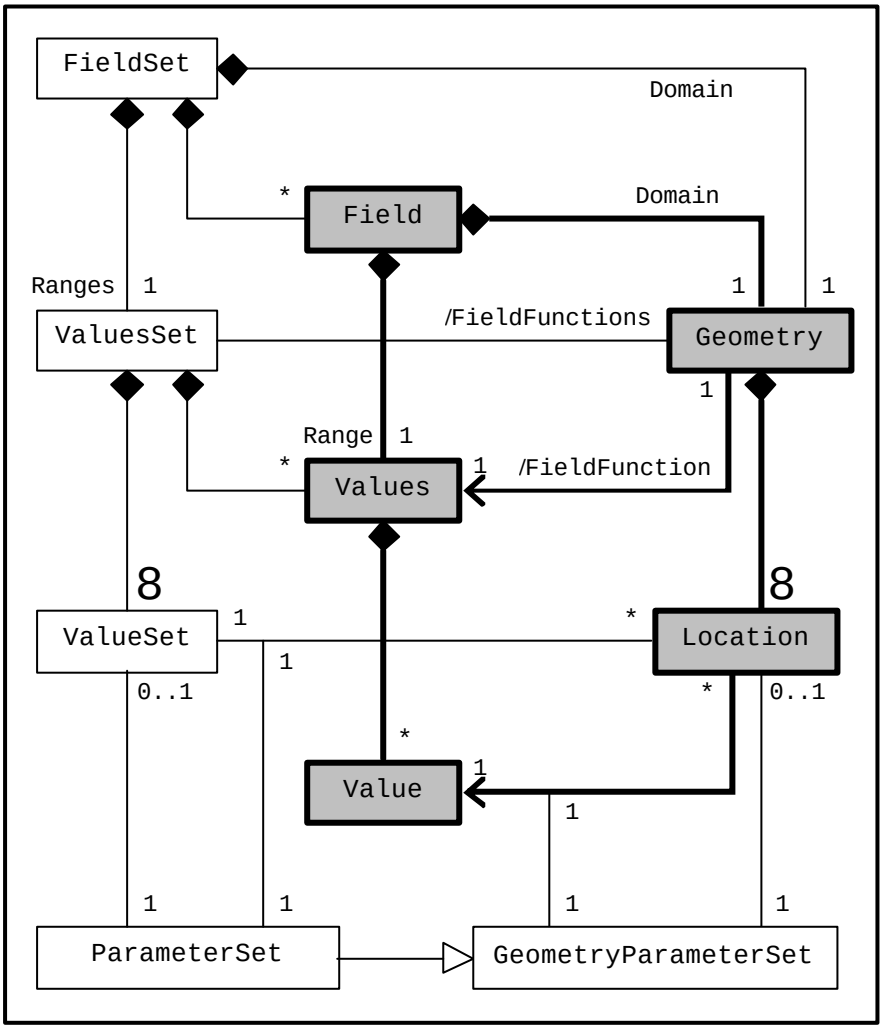

Fig. 2: Conceptual UML class diagram of a standard field-based model. Shaded classes and bold line relations represent the core field-based model, and the remainder represents extensions made in this paper.

2.4 First extension of the core field-based model

Fig. 2 describes several proposed extensions of the basic UML model. A one-to-one association between class *Location* and a class called *GeometryParameterSet* is introduced motivated by the fact that geometries may be described mathematically as parameter

functions. A parametric representation of a *Geometry* class is defined as a function from a *parameter space* to the *Euclidian space.* Therefore each combination of parameters (i.e. each instance of *GeometryParameterSet*) is associated with at most one *Location*. Furthermore, the dimensionality of the parameter space (i.e. the number of parameter attributes of the *GeometryParameterSet*) is equal to the dimensionality of the geometry (i.e. point geometries being 0-dimensional, curve geometries being 1-dimensional, etc.), and being equal or inferior to the dimensionality of the Euclidian space. There are an infinite number of possible functions representing the same geometry. Hence, an *GeometryParameterSet* instance contains all parameters associated with one particular *Location* on a *Geometry* instance The fact that one *Location* instance is associated with exactly one *GeometryParameterSet* instance and exactly one *Value* instance, enables the derived association of the *GeometryParameterSet* class with the relation between the *Value* and *Location* classes. Such types of relations are valid in UML and may often be useful. This first extension of the field model leads to the following alternative formal definition of a field: *"A field is defined as an UML class representing a function from an n-dimensional parameter space to a value space"*.

2.5 Second extension of the core field-based model

So far we have avoided discussing the dimensionality of the value space. Normally it is 1, but 2 and 3 are also common (e.g. a gravity field has a 3-dimensional value space). This observation suggests the possibility of defining the geometry itself as a new field over the same geometry (i.e. a trivial one-to-one field function), and therefore, over the same parameter space. Thus, fields defined by this approach belong to a family of fields over the same parameter space representing any phenomena. Thus, the field definition can be generalized to an *m*-dimensional value space. Let us introduce the concept (and class) *FieldSet* to represent this family of fields over the same geometry.

This second extension of the field definition is represented in UML by the classes and relations on the left hand side of fig. 2. The classes called *ValuesSet* and *ValueSet* are defined accordingly as aggregates of the *Values* and *Value* classes respectively, and with similar relations to the other classes in the diagram. Similarly, each instance of *Values* associated to an instance of *ValuesSet* have their own *FieldFunction*, and the collection of all *FieldFunction* relations is represented by the derived relation between the *Geometry* and *ValuesSet* classes called *FieldFunctions*. The "many-cardinality" of the *Values* to *Value* relation has been extended to an infinity-cardinality of the *ValuesSet* to *ValueSet* relation motivated by the previously-mentioned fact that the *Geometry* itself can be modeled as a field.

2.6 Third extension of the core field-based model

The concept (and class) called *ParameterSet* introduces one final extension of the UML diagram of fig. 2 motivated by the fact that we do not need to restrict the parameters to only those required for defining the *Geometry*. Hence, the *ParameterSet* is a specialization of the *GeometryParameterSet* including possible additional parameters. The second and third extensions lead to the following general definition of the *FieldSet* (as an extension of the

definition of the *Field*): *"A FieldSet is defined as an UML class representing a function from an n-dimensional parameter space to an m-dimensional value space"*.

2.7 A preliminary candidate base model

A candidate base-model is steadily appearing in terms of a model generalizing the object-based and field-based models. We must now focus on describing the candidate common base-model and trying to prove that the two models are specializations of the candidate base-model. It must be emphasized that we are not using the term *specialization* and *generalization* to describe class inheritance, but rather to describe model specialization in terms of restrictions to class relation type and cardinality.

It seems clear that the *Entity* class from fig. 1 resembles a conceptual specialization of the *FieldSet* class from fig. 2, provided that the *AttributeSet* class is considered as a conceptual specialization of the *ValuesSet* class. This is true if an *AttributeSet* class is considered as a *ValuesSet* class that is independent of parameters. This is equivalent to associating a *ValueSet* class with the special case *ParameterSet* class having 0 parameters. This association is valid and equivalent to the infinity-cardinality relation between the *ValuesSet* and the *ValueSet* degenerating to the special case 0-cardinality relation, i.e. no relation at all.

It is evident that the *Field* class from fig. 2 represents a conceptual specialization of a *FieldSet* class in the special case of an instance of *FieldSet* aggregating exactly one instance of *Field*. This is true because the associated instance of *ValuesSet* also aggregates exactly one instance of *Values*. This means that the *Field* and *FieldSet* classes will be merged because they have both exactly the same set of one-to-one aggregation relations to other classes in the model.

Hence, the *FieldSet* class represents a key class in a candidate base model because it has the potential of representing both the *Entity* and the *Field* classes in a generalized model. In addition, all four classes on the left hand side of fig. 2 must be included into this candidate base model because they are all included in the chain of arguments used for proving model specialization.

In order to neutralize the concept, the names of the classes will be changed. The name *ParameterizedGeographicObject* (abbreviated *PGObject*) will be introduced in order to replace *FieldSet*. All classes belonging to the common base-model will be given names prefixed by the abbreviation *PGO*, and the model itself will be called the *PGOModel*. As an exception to this rule, the name *PGObject* will be used instead of *PGOObject*. The terms *field* and *entity* will be completely omitted from the *PGOModel* in order to avoid mixing its concepts with those of the classical models.

3 ParameterizedGeographicObject or PGO

3.1 Definitions of the PGOModel concepts

In fig. 2, it can be observed that the *FieldSet* and the *ValuesSet* classes may be merged into one class because of the one-to-one cardinality. In the *PGOModel* framework, represented by UML in fig. 3, this merged class is called *PGObject*. For similar reasons, the *ValueSet* and the *ParameterSet* classes have been merged into a class called *PGOAtom*.

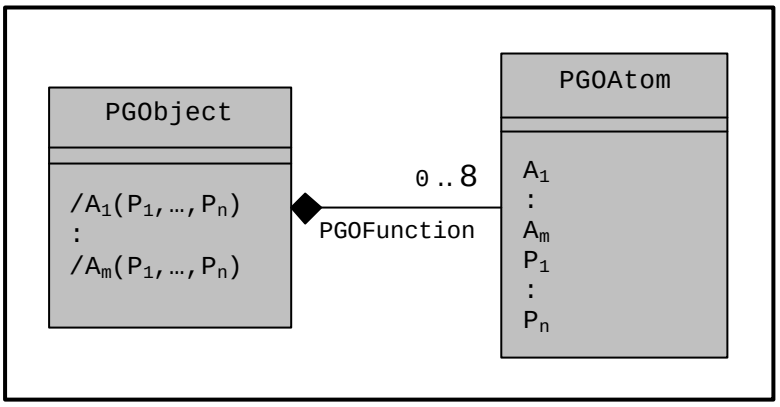

Fig. 3: Conceptual UML class diagram of the *PGOModel*.

With this merging, the member attributes A_1 to A_m of the *ValueSet* and the member attributes P_1 to P_n of the *ParameterSet* classes from fig. 2 have become ordinary member attributes of the *PGOAtom* class. On the other hand, the *ValuesSet* class from fig. 2 has no ordinary attributes of its own, only derived attributes from its relation to the *ValueSet* and *ParameterSet* classes. These derived attributes $/A_1$ to $/A_m$ are functionally dependent on the attributes A_1 to A_m of the associated *ValueSet* class and the attributes P_1 to P_n of the associated *ParameterSet* class. Hence, the merged *PGObject* class must include derived attributes $/A_1$ to $/A_m$ in order to maintain its position as a candidate base model.

The *FieldFunction* and *FieldFunctions* concepts of fig. 2 are generalized into the functional relationship from the attributes A_1 to A_m and P_1 to P_n of *ValueSet* and *ParameterSet* to the derived attributes $/A_1$ to $/A_m$ of the *ValuesSet* in the extension of the *Field* model. Consequently, this functional relationship is directly linked to the relation between the *PGObject* and *PGOAtom* classes in the generalized *PGOModel*, and the relation is called *PGOFunction*. This relation is defined as a *strong aggregation* in UML, which means that the *PGObject* incorporate all associated *PGOAtom* objects.

3.2 Mathematical description of the PGOFunction

The *PGOFunction* can alternatively be considered a mathematical function from an n-dimensional parameter space to an m-dimensional value space and is represented informally and graphically in fig. 4. The entire value space is often called the *codomain* (Weisstein et al. 2004: Topic "Codomain") of a function, and the subset of the codomain representing legal

value space values is called the *range* (Weisstein et al. 2004: Topic "Range") of the *PGOFunction*. The subset of the parameter space including parameters with a defined mapping in the attribute-space represents the *domain* (Weisstein et al. 2004: Topic "Domain") of the *PGOFunction*. The range of the *PGOFunction* will normally represent a subset of the codomain. Therefore, the *PGOFunction* is normally *non-surjective* (Weisstein et al. 2004: Topic "Surjection"). In order for the *PGOFunction* to be mathematically *injective* (Weisstein et al. 2004: Topic "Injection"), it must represent a one-to-one relationship from the domain to the range. This property reflects our choice of *PGOFunction*, but it is easy to find examples of *non-injective PGOFunction*s. For this reason, the *PGOFunction* is usually *non-bijective* (Weisstein et al. 2004: Topic "Bijection"). The dimensionality *n* of the parameter space represents and may be defined as the *degree of freedom* (Weisstein et al. 2004: Topic "Degree of Freedom") for the associated *PGObject*.

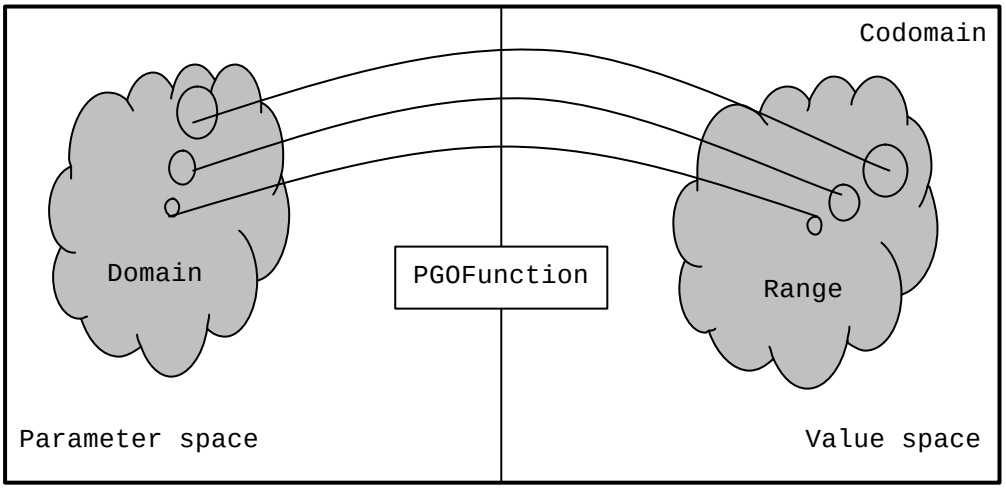

Fig. 4: Informal drawing of the *PGOFunction* as an injective function from a domain in the parameter space to a range in the codomain of the value space.

3.3 Formal definitions of the main elements of the PGOModel

The main elements of the core *PGOModel* of fig. 3 are defined as follows:

*A **PGOModel** is a UML class model conceptualizing a geographic object with attributes dependent on a set of n parameters varying over an n-dimensional domain where n=0.*

*A **PGObject** is a UML class conceptualizing a geographic object in a PGOModel. The PGObject has m derived attributes, each being dependent on all or a subset of the n parameters defined in the PGOModel. The value n is defined as the "degree of freedom" of the PGObject.*

*A **PGOAtom** is an UML class conceptualizing the set of values of a PGObject class linked to one specific combination of values of the n parameters defining the PGObject. Hence, the n parameter values of the PGObject may also be considered attributes of the PGOAtom class.*

*A **PGOFunction** is an UML class relation conceptualizing the strong aggregation of a possible infinite number of PGOAtom objects into a PGObject. A PGOFunction is defined mathematically as a function from a domain in an n-dimensional parameter space to a range in an m-dimensional value space.*

4 Extensions of the PGOModel

The core *PGOModel* from fig. 3 must be elaborated in order to resolve our initial challenge. Some of the extensions are not necessary in the development of a conceptual model but are included in order to make the *PGOModel* more easily applicable and understandable. In a few instances we may exceed the limits of the pure conceptual modeling level.

4.1 PGObject tessellation

The first extension of the *PGOModel* is the result of a possible practical (but not conceptual) problem concerning the one-to-infinity cardinality of the *PGOFunction* relation. The solution is to split the *PGObject* into a set of smaller sub-*PGObjects* in such a way that each instance of *PGOAtom* class of fig. 3 is associated with "one and only one" sub-*PGObject*. This solution is analogous to and generalizes the concept of tessellation of geometry objects as described by Egenhofer and Herring (1991) and restated by Bian (2000). This sub-*PGObject* will subsequently be called *PGOPatch*. Similarly, the complete tessellation relation will be called a *PGOTessellation*, and the relation between the *PGOPatch* and *PGOAtom* classes will be called a *PGOPatchFunction*. The introduction of the *PGOPatch* class modifies the UML model of fig. 3 to an extended UML model presented in fig. 5. The challenge of the tessellation process is normally to make each *PGOPatch* small enough to be able to formulate an explicit *PGOPatchFunction* for each *PGOPatch*.

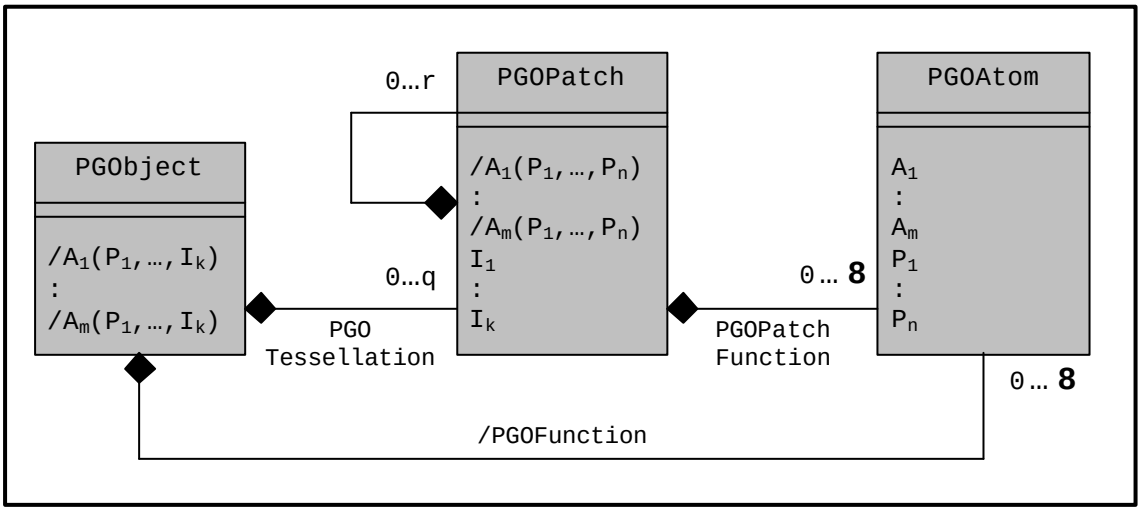

Fig. 5: Conceptual UML class diagram of the alternative tessellated *PGOModel*.

The following comments are connected to elements of fig. 5:

- The *PGOFunction* relation of fig. 3 is replaced in fig. 5 by the combination of the *PGOTessellation* relation of *finite* cardinality and the *PGOPatchFunction* relation of *infinite* cardinality. Hence, the *PGOFunction* relation of fig. 3 has turned into a UML *derived relation* in fig. 5.

- Practical problems connected to the infinite upper cardinality of the *PGOPatchFunction* relation is solved by defining it as an explicit mathematical function. This means that the *PGOPatchFunction* relation (and consequently the derived *PGOFunction* relation) becomes an *implicit relation*.
- The tessellation process produces a *PGOTessellation* relation between a *PGObject* and a finite number of *q PGOPatch* objects, and each of them can be assigned a unique set of index values (I_1, ..., I_k). The indexes of the *PGOPatch* class can be considered an extension of the parameter concept by additional discrete parameters representing the indexes.
- The *PGOTessellation* relation may in some cases also be defined as an implicit relation by an explicit mathematical function. This is the case of a regular grid type of *PGOTessellation*.
- The "self-relation" of the *PGOPatch* class represents a possible tessellation of *PGOPatch* instances enabling hierarchic structures of *PGOPatch* instances. This option may be utilized to mix explicit and implicit *PGOTessellations*.

4.2 PGOPatch generic subclasses

There are no restrictions on how to define *PGOPatch* objects, except for the requirement that their attributes (i.e. *PGOPatchFunction*) may be formulated explicitly as a function of the parameters. It is useful to search for generic (i.e. a standardized mathematical type) sub-classes to the *PGOPatch* class to be reused in different *PGOModel* objects. There are three obvious candidates chosen on the basis of simplicity, and their class inheritance is presented by a UML class diagram in fig. 6:

- A *PGOPatchConstant* represents a *PGOPatch* of constant value. Thus, the *PGOPatchConstant* is independent of parameters.

- A *PGOPatchHyperCube* is defined over a domain in the parameter space of *hyper-cube* and axis-parallel geometric shape. This is a generalization into a parameter space of arbitrary dimension of the *rectangular* shaped geometry in a parameter domain of dimension $n=2$. $2n$ hypercube corner points with known values fix each instance in the parameter space. Values linked to parameters in the hyper-cube interior are computed by a simple low-order polynomial interpolation function, for example a bilinear polynomial interpolation function in the $n=2$ case as shown in fig. 7.

- A *PGOPatchSimplex* is defined over a domain in the parameter space of *simplex* geometric shape (Frank & Egenhofer 1992). This is a generalization into a parameter space of arbitrary dimension of the *triangular* shaped geometry in a parameter domain of dimension $n=2$. $n+1$ simplex corner points with known values fix each instance in the

parameter space. Values linked to parameters in the simplex geometry interior are computed by a simple linear interpolation function as shown in fig. 8.

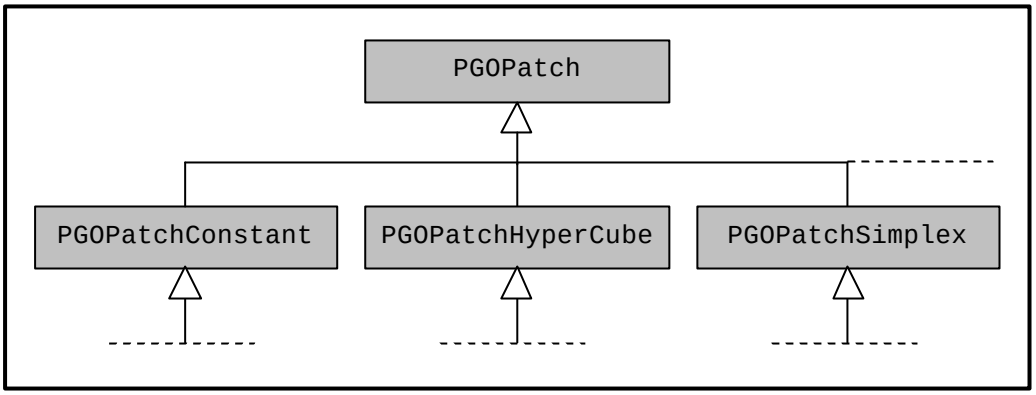

Fig. 6: Conceptual UML class diagram of *PGOPatch* subclasses.

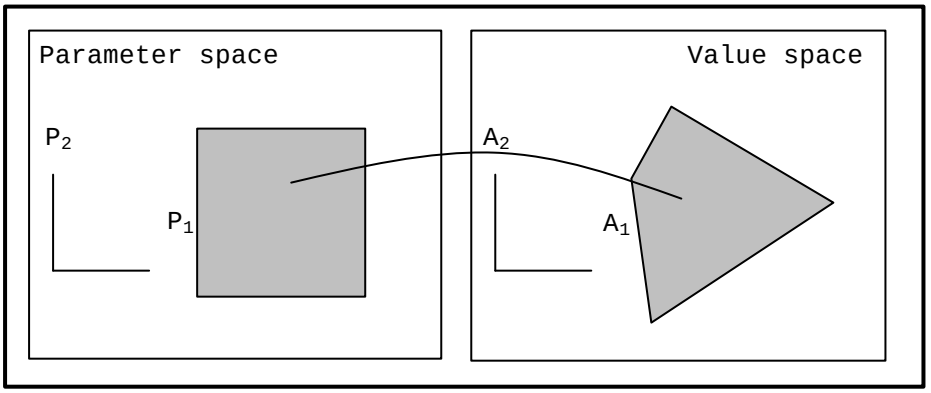

Fig. 7: Example of a 2-parameter *PGOPatch* object of type *PGOPatchHyperCube* with two attributes A1 and A2.

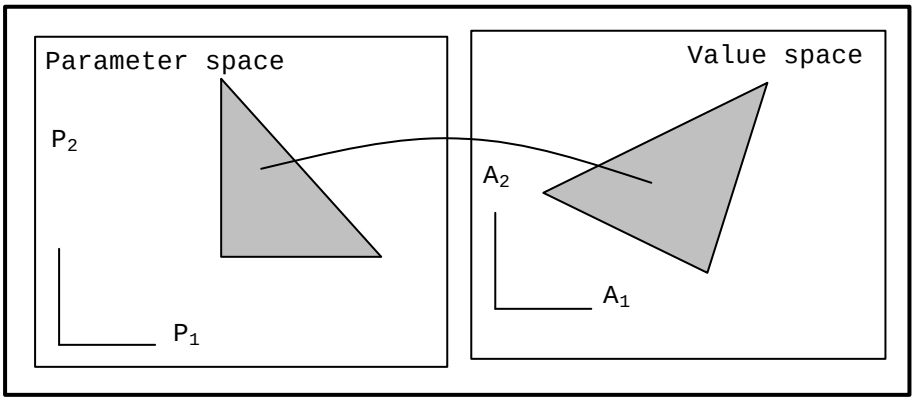

Fig. 8: Example of a 2-parameter *PGOPatch* object of type *PGOPatchSimplex* with two attributes A1 and A2.

4.3 PGObject derived-attributes

The *PGOModel* includes possible implicit definition of *PGObject* attributes as a function of other attributes of the same *PGObject* or attributes of other *PGObjects.* Such attributes are called *derived -attributes* in UML. The *derived-attribute* concept of *PGObjects* may generalize the *overlay* concept of GIS.

4.4 PGObject parameter projections

It is useful to consider the possibility of giving a subset of the *PGObject* parameters fixed values, while allowing a derived *PGObject* to be dependent on the remainder of the parameters. This option is useful in describing snapshots or subsets of an entire *PGObject*. Such derived *PGObjects* represent an analogy to *projections* in geometry, hence, can be called a *PGOProjection*.

4.5 PGObject partitioning

The *PGOModel* (fig. 5) requires that the *PGObject*, *PGOPatch* and *PGOAtom* objects all have the same full set of *m* attributes. This is impractical if the *PGOModel* prefers different *PGOTessellation* for different attributes. This option can be supported by splitting the attributes of the *PGObject* over a specific number of *PGOPart* instances as shown in fig. 9. The *PGObject* then becomes a strong aggregation (called *PGOPartition* relation) of *PGOPart*, each having their own tessellation called *PGOPartTessellation*.

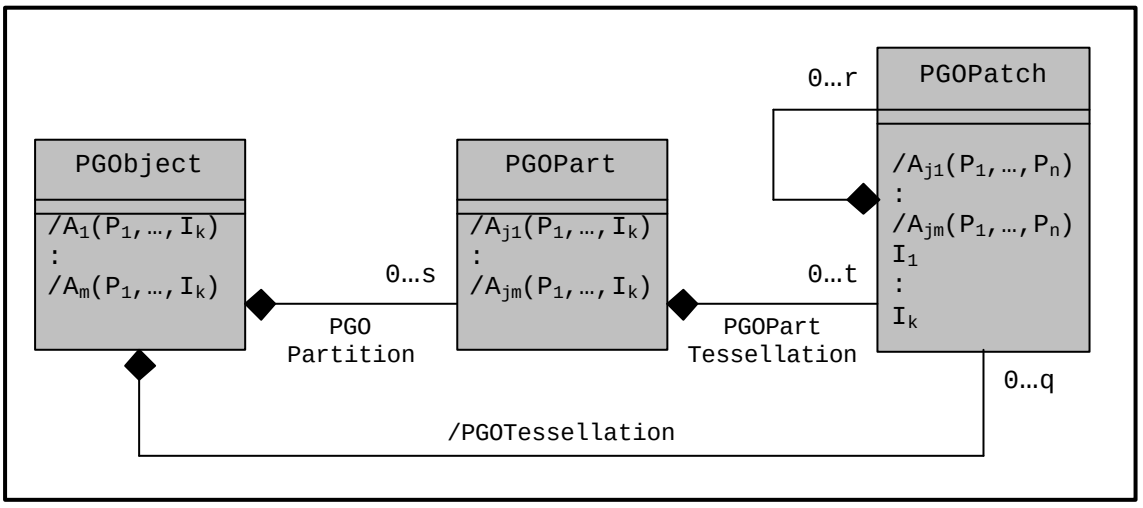

Fig. 9: Conceptual UML class diagram of *PGObject* partitioning.

5 Proof of the PGOModel being a common base-model

The proof of the *PGOModel* according to the initial requirements is its ability to encompass the classic object-based and field-based model of GIS. Several underlying arguments for this proof have already been noted in the previous development in this paper of the *PGOModel*. The letters *n*, *m*, etc. in this section refer to fig. 3, fig. 5 and fig. 9.

The classic object-based model is described by the *PGOModel* by setting $n=0$. The infinity-cardinality of the *PGOFunction* is consequently degenerated into a 0-cardinality. Thus, there is nothing to tessellate and partition, and therefore $k=q=r=s=t=0$. In other words, the *PGOPart*, *PGOPatch* and *PGOAtom* classes are not involved in this special case. The case of an object-based model attribute of type geometry may represent a possible exception, because we have to set $n>0$ if we define the geometry attributes mathematically using a parametric representation and a standard geometry tessellation. Furthermore, we have to define $k>0$, $q>0$ and $r=s=t=0$.

The classic field-based model is described by a *PGOModel* where $m=1$. Thus, there is no *PGOPartition* involved, and hence, $s=t=0$. The field function is furthermore represented by the *PGOFunction*, and the number of parameters of the *PGOAtom* (i.e. the *degree of freedom* of the *PGObject*) is set to n where $0<n<4$, depending on the dimensionality of the geometry defining the domain of the field. If the geometry is defined by a standard geometry tessellation, then $k>0$, $q>0$ and $r=0$.

6 Discussions

6.1 PGObject attributes of type object-reference

The model presented by Cova & Goodchild (2002) defines the concept of *"object field"* as a method for integrating the two classical models. Their approach proposes a solution to the model integration problem by defining one of the classical models in the context of the other. The *"object field"* approach basically links object instances to any field location, and hence represents a model with both field-like and object-like properties. The *"object field"* is formally established by augmenting the field concept with attributes of type *object-reference*. This augmentation is similarly possible in the *PGOModel* framework by allowing *PGObject* attributes of type *object-reference*. For both models this augmentation is restricted to tessellations having patches of constant value because there is in general no valid algebra defined for object-references. This means in *PGOModel* terms that an *"object field"* is restricted to *PGOPatch* objects of subtype *PGOPatchConstant*. All four cardinality types in the object/field relation of the *"object field"* model (one-to-one, etc.) are possible in both models by allowing an additional augmentation of field or *PGObject* attributes respectively with the type *"container-of-object-references"*. Hence, the *"object field"* model intuitively maps to the *PGOModel.*

The opposite is not true because the *object* concept of the *"object field"* model is not explicitly augmented with the general possibility to allow attributes being dependent on parameters. Hence, the *"object field"* model is a special case of the *PGOModel.*

6.2 PGOModel considered as a hybrid model

The model presented by Winter 1998 proposes a *hybrid representation model* for the object-based and field-based concepts. There is basically no significant difference between the concept of *common base-model* and *hybrid representation model*. This means that the model of Winter 1998 and the PGOModel are both developed for the purpose of trying to bridge the two classical models of GIS even if the *hybrid representation model* is much more focused on developing an implementable model. However, the two models are developed using different tools, and the two models have significant differences.

The model of Winter 1998 presents a *hybrid representation model* for the traditional raster and vector representation models to be used as a common representation model for the entity-based and field-based conceptual models in GIS. The model is developed using strict mathematical formalism. The atomic part of the *hybrid representation model* consists of the topological 0-, 1- and 2- cells forming a discrete regular and axis-parallel topological pattern with a finite user-defined resolution. The model is basically limited to 2D, but could easily be generalized to higher dimensions. The model has no parameter concept. One particular 2-cell with its associated 0- and 1- cells maps nicely to the *PGOPatch* subclass *PGOPatchHypeCube* of dimension $n=2$. Seen from this point of view the *hybrid representation model* could be regarded as a special case of the PGOModel because it does not include other types of *PGOPatch* objects and it does not explicitly include any concept mapping the *PGOAtom* concept. However, the *hybrid representation model* is enriched with the necessary formal topological concepts missing in the current version of the PGOModel. These issues are classified as a topic for further work in the conclusion section of this paper. The methodology of Winter 1998 is a promising approach for such a work.

6.3 Ontology of the PGOModel

The formulation of the *PGOModel* requires reflection on its ontological foundations and potential consequences.

The ontological foundation of the classical field-based model is still subject to debate (Bian 2000, Peuquet el. al. 1999, Smith & Mark 1998). The controversy is primarily linked to the nature and origin of a field, and especially to a field which is possibly the modeling result of so-called *fiat-objects* (Smith & Mark 1998). *Fiat-objects* are defined as the result of "human reasoning and language", and hence, exist beyond the physical world. *Fiat-objects* are more likely to be "field-like" than "object-like" according to Peuquet et. al. (op.cit.).

The alternatives, which represent features in the physical world, are defined as so-called *bona fide objects*. The previous argument on *fiat-objects* leads to a possible inverse statement of *bona fide objects* being more likely "object-like" than "field-like". Hence, the classical object-based model, generally representing *bona fide objects*, poses no major ontological problem

because the objects of this model generally represent physical features located in geographic space, if we exclude the problem of representing *bona fide objects* with fuzzy extent.

However, even *fiat-objects* represented in an object-oriented framework as the core field-based model of fig. 2 should not pose any major ontological problem. This is because object-oriented modeling as such can easily transform abstract (i.e. non-physical) objects into quasi-physical objects on the modeling level. Furthermore, there is no ontological conflict related to our extensions of the *Field* object into a *FieldSet* object introduced in fig. 2 (and hence the *PGObject* of fig. 3) because the extension simply introduces the option of defining a *FieldSet* object as an "aggregation of *Field* objects over the same geometry".

However, our extension of the field model connected to the *ParameterSet* object introduced in fig. 2 (and hence the *PGOAtom* object of fig. 3) is ontologically more challenging. Mathematically the *PGOAtom* represents the smallest element aggregating the entire *PGObject*. However, the fact that many *PGOFunction*s can construct the same *PGObject* is ontologically problematic. This means that the relationship between a *PGObject* and a *PGOAtom* is ontologically more complex than the normal relationship between physical objects and physical atoms. We conclude that the *PGOAtom* object clearly cannot be reasonably ontologically founded beyond the pure abstract mathematical level.

7 Examples

7.1 Glacier example

A *glacier* is a well-defined physical object subject to GIS modeling. It has a clear extent, and its attributes could be defined either as independent (*size* and *glacier type*) or as dependent (*velocity* and *thickness*) on the specific location on the glacier.

A glacier is easily modeled by the object-based model provided that we restrict attributes to those being independent of location, such as the *size* and *glacier type*.

The field-based model can be used in the modeling of the individual attributes of a glacier, in particular those being dependent on location, such as the *velocity* and *thickness*. Glaciologists are also using the concept of *velocity field* when describing the dynamic behavior of a glacier.

The modeling of the glacier as a *PGObject* enables the integration of the velocity and thickness attributes into an object-based model. A FEM (*Finite Element Method*) based on the glacier flow-line structure, represents a candidate *PGOTessellation* often used by glaciologists. Each *PGOPatch* is of the *PGOPatchHyperCube* subtype if we model continuously varying attributes or the *PGOPatchConstant* subtype if we model discretely varying attributes. In the *velocity* attribute case, known velocity values are associated with *PGOPatchHyperCube* object corner points or entire *PGOPatchConstant* objects. The dimension *n* of the *PGOPatch* object is 3 in a static model or 4 in a model describing glacier variations over time. The first parameter P_1 may represent the distance along the center flow-line of the glacier. The second parameter P_2 may represent the signed depth perpendicular to the glacier center flow-line. The third parameter P_3 may represent the signed horizontal

distance from and perpendicular to the center flow-line. The alternative fourth parameter P_4 may represent the distance in time from a defined time origin. The word *distance* does not necessarily refer to a Euclidian distance, but rather a parameter with mathematical *metric* properties (Weisstein et al. 2004: Topic "Metric"). Neutralized parameter values varying in the range from 0 (or -1) to 1 over the entire glacier may be a better choice. This *PGOTessellation* has a topologically regular "hyper-raster" type of structure and each *PGOPatch* object can be indexed accordingly. Consequently, $k = n$ and the number of *PGOPatches* in all k directions can be set constant. The flow-line tessellation model enables ignoring the topology of merging lateral glaciers.

The *PGOAtom* object represents a physical ice-molecule at a particular point in time and space with its set of constant value attributes and unique set of constant value parameters. The *PGOModel* enables the modeling of the location in time and space of each *PGOAtom*, and hence each *PGOPatch* and *PGObject*, by including *x*, *y*, *z* and *time* in their list of attributes.

The *thickness* attribute is not directly dependent on a "3D" tessellation of the glacier and can be modeled by aggregating all *PGOPatches* which are "vertically stabled on top of each other" into a new *PGOPatch* object using the *PGOPatch* class "self-relation". The word *vertically* in this FEM type of tessellation model does not imply vertical in a strict sense, but instead perpendicular to the surface of the glacier, i.e. *PGOPatch* objects with the same I_2 index value. Hence, the velocity and thickness attributes will have different *PGOTessellations*, and it is necessary to introduce a *PGOPart* object into the model in order to model them as sister attributes. An alternative to modeling the thickness attribute as an explicit attribute is to model it as a derived-attribute computed as the distance between the top and bottom surfaces of corresponding parts of a glacier. The top and bottom surface attributes both have the same *PGOTessellation* as the thickness attribute. These attributes can alternatively be modeled as two *PGOProjection* objects of the glacier *PGObject* where P_2 is equal to 0 and 1 respectively.

7.2 Railway network example

Utility networks such as railway networks, have traditionally been modeled using an object-based model. A network is first decomposed into its smallest elements, each characterized by a uniform set of attributes, called *track sections*, and then each of them is modeled as a class. The complete network topology can subsequently be modeled by introducing a set of topological relations between track section objects, by introducing an imaginary *track node* object. A sequence of connected track sections between two neighboring track *junction* nodes is called a *track connection*. Discretely varying railway network attribute information, such as track *quality*, is modeled as attributes of the track section object. However, the same approach is not possible for continuously varying attribute information, such as possible *top-speed*. Attribute information connected to points on the network, such as a *security installation*, can be related to track node objects. The latter may require introducing artificial track nodes into the network.

The *quality* and *security installation* attributes can in theory be subject to a field-based modeling approach by considering a *track section* and *track node* based constant value tessellation of the field. This approach is however never applied because it is equivalent to the

object-based model of the network. In contrast, the *top-speed* attribute is better suited to a field-based approach because it can take advantage of a non-constant value tessellation. This is a field over a curve geometry domain usually represented parametrically as curvilinear coordinates by railway authorities. Curvilinear coordinates are represented by a continuous measure (for instance distance) along the track from a well-defined starting-point.

The latter approach is a natural starting-point for introducing the *PGOModel* to the railway network example. One *PGObject* may represent the entire network and it may be subject to a *PGOTessellation* of *PGOPatch* objects analogous to the splitting of the object-based model of the network. A *track section* is represented by a *PGOPatch* object and *track connections* are represented by an aggregated *PGOPatch* object. In the *quality* attribute case, known track quality values are associated with *PGOPatchConstant* objects, while in the *top-speed* attribute case, known speed values are associated with *PGOPatchHyperCube* object corner points. The dimension *n* of the *PGOPatch* object is *1* in a static model, *2* if we wish to describe variations over time and *3* if we also wish to describe scale-dependent variations. The first parameter P_1 may represent the curvilinear coordinate along the track from a well-defined reference point. The alternative second parameter P_2 may represent the distance in time from a defined time origin. The alternative third parameter P_3 may represent scale. The indexing scheme for the *PGOPatch* objects is necessarily complex due to the topology of the network. Moreover, the *PGOModel* presented in this paper has not yet defined any mechanisms for explicit modeling of the *PGOPatch* topology. This means that the current *PGOModel* is restricted to implicit (i.e. "spaghetti") modeling of *PGOPatch* topology. An alternative approach is to model each *track connection* as a separate *PGObject* and treat network topology as topological relations between *PGObject*s. The latter case requires an explicit *PGObject* representing the *track node*.

The *PGOAtom* represents a physical location of the network at a particular point in time, space and resolution with its set of constant value attributes and unique set of constant value parameters. The *PGOModel* enables the modeling of the location in time and space of each *PGOAtom*, and hence each *PGOPatch* and *PGObject*, by including *x*, *y*, *z*, *time* and *scale* in their list of attributes. This definition allows two *PGOAtom* objects with the same parameter P_1 and a different parameter P_2 representing two different locations in space in the case of a track relocation project. If we let *x*, *y* and *z* depend on the P_3 parameter, then we are able to represent a scale-dependent network layout which vary from a precise network layout to an overall schematic layout. This example also allows a continuous derived-attribute representation of *travel time* from the start of the *track connection* as the multiplication of the *top-speed* and the parameter P_1. *Security installation* may be modeled as separate *PGObject*s which are related to the network *PGObject(s)* using the *object reference PGObject* attribute type.

8 Conclusions

This paper presents a candidate base-model called *PGOModel* for the classical object-based and field-based conceptual models in GIS. The *PGOModel*, the object-based model and field-based model are formally defined as pattern-models by using the UML modeling language.

Within the scope of this paper, the proof of validity of the *PGOModel* is linked to its demonstrated ability to *encompass* the classical object-based and field-based models.

Except for the *ontological* problem of the *PGOAtom* concept, we have seen that the *PGOModel* does not represent additional ontological problems compared to the classical object-based and field-based models in GIS. The ontological problem of the *PGOAtom* is related to the way *PGOAtom* parameters are considered. However, this does not represent any major ontological problem if we maintain that the parameters represent pure mathematical tools used in the construction of a *PGObject* from *PGOAtoms*.

There are several issues connected to the *PGOModel* which are beyond the scope of this paper. These issues provide appropriate topics for further research. Several have already been mentioned, such as *PGObject relations* and *topology*, others include the important concept of *methods* in object-orientated modeling. Furthermore, the question of application of the *PGOModel* to non-conceptual modeling levels has not been challenged. In particular the applicability of the *PGOModel* on the *implementation* level is of special interest.

Regarding the potential *usefulness* of the *PGOModel*, it is to be hoped that the model offers greater insights into the nature of geographic information. It may help us understand the relationship between the two classical models while revealing their strengths, weaknesses and limitations. It is, however, important to keep in mind that both the classical object-based and field-based models are highly adequate for solving most current GIS conceptual modeling issues.

It is believed that the major contribution of the PGOModel beyond playing the role as a common base-model for the two classical models is linked to its alternative formulation and definition of the field concept. The traditional view of a field as a mathematical function from a geometry space domain to a value space could be supplemented by the alternative and generalized view of a field defined as a set of mathematical functions from a parameter space domain to a multidimensional value space. Another contribution of the PGOModel may be the demonstrated usefulness of including attributes whose value is dependent on parameters into the classical object-based model.

9 Acknowledgments

The author is particularly appreciative for the helpful comments of Dr. Tor Lønnestad, and our discussions connected to several draft versions, and also to the anonymous referees for their valuable comments on earlier versions of this paper. I am grateful also to Mrs. Judith McGuiness Torvik for assistance with improving the English language.

10 Literature

Angel, S., and Hyman, G. M., 1976, *Urban Fields: a geometry of movement for regional science* (London: Pion Limited).

Bian, Ling 2000, Object-oriented representation for modeling mobile objects in an aquatic environment. *International Journal of Geographical Information Science, vol. 14, no. 7, 603-623*

Burrough, P. A., 1996, Natural objects with indeterminate boundaries. In *Geographic Objects with Indeterminate Boundaries,* edited by P. A. Burrough and A. U. Frank (London: Taylor and Francis), pp. 2–28.

Burrough, P. A., and McDonnell, R. A., 1998, *Principles of Geographical Information Systems* (Oxford: Oxford University Press).

Couclelis, H., 1992, People manipulate objects (but cultivate Fields): beyond the raster-vector debate in GIS. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space,* Lecture Notes in Computer Science 639, edited by A. U. Frank and I. Campari (Berlin: Springer-Verlag), pp. 65–77.

Cova, T. J. and Goodchild, M. F. 2002, Extending geographical representation to include "Fields of spatial objects. *International Journal of Geographical Information Science, vol. 16, no. 6, 509-532*

Egenhofer, M. J., and Frank, A. U., 1987, Object-oriented databases: database requirements for GIS. In *Proceedings of the International GIS Symposium: The Research Agenda, Volume 1* (Washington, DC, US Government Printing Office), pp. 189–211.

Egenhofer, M., and Frank, A., 1992, Object-oriented modeling for GIS. *Journal of the Urban and Regional Information Systems Association,* 4, 3-19.

Egenhofer, M. J., and Herring, J. R., 1991, High-level spatial data structures for GIS. In *Geographical Information Systems: Principles and Applications,* edited by D. J. Maguire, M. F. Goodchild and D. W. Rhind (London: Longman), pp. 227–237.

Egenhofer, M. J., Glasgow, J., Gunther, O., Herring, J. R., Peuquet, D. J., 1999, Progress in computational methods for representing geographical concepts. *International Journal of Geographical Information Science, vol. 13, no. 8, 775-796*

Frank, A. U., 1992, Spatial concepts, geometric data models, and geometric data structures. *Computers and Geosciences,* 18, 409–417.

Frank, A. U., 1996, The prevalence of objects with sharp boundaries in GIS. In *Geographic Objects with Indeterminate Boundaries,* edited by P. A. Burrough and A. U. Frank (London: Taylor and Francis), pp. 29–40.

Frank, A. U., and Egenhofer, M., 1992, Computer cartography for GIS: an object-oriented view on the display transformation. *Computers & Geosciences,* 18, 975–987.

Goodchild, M. F., 1989, Modeling error in objects and Fields. In *Accuracy of spatial databases,* edited by M. F. Goodchild, and S. Gopal (London: Taylor & Francis).

Goodchild, M. F., 1992, Geographical data modeling. *Computers & Geosciences*, 18, 401–408.

Kemp, K. K., 1997, Fields as a framework for integrating GIS and environmental process models. part 1: representing spatial continuity. *Transactions in GIS*, 1, 219–234.

Oosterom, P.v., and Vanderbos, J., 1989, An object-oriented approach to the design of geographic information systems. *Computers and Graphics*, 13, 409–418.

Peuquet, D. 1988, Representations of Geographic Space: Toward a Conceptual Synthesis, *Annals of the Association of American Geographers* 98 (3): 375.

Peuquet, D., Smith, B., and Brogaard, B., 1999, *The ontology of "Fields: report of a specialist meeting held under the auspices of the Varenius Project* (Santa Barbara: National Center for Geographic Information and Analysis).

Rumbaugh, J., Blaha, W., and Premerlani, F., 1991, *Object-Oriented Modeling and Design* (Englewood Cliffs, New Jersey: Prentice Hall) .

Rumbaugh, J., Jacobson, I., Booch, G., 1998, *The unified modeling language reference manual.* (Reading, Mass.: Addison-Wesley).

Sachs, M., 1973, *The Field concept in contemporary science* (SpringField: Charles C. Thomas).

Smith, B., and Mark, D., 1998, Ontology and geographic kinds. In *Proceedings of International Symposium on Spatial Data Handling (SDH '98): Vancouver, B.C.*, pp. 308–318.

Tobler, W. R., 1978, Migration fields. In Clark, W., and Moore, E., eds., *Population Mobility and Residential Change*, Studies in Geography No. 25 (Evanston: Department of Geography), pp. 215–232.

Weisstein, E. W. et al., 2004, *MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com*

Winter, S., 1998, Bridging Vector and Raster Representation in GIS. In *Proceedings of the 6th International Symposium on Advances in Geographic Information Systems (ACM-GIS '98): Washington, D.C., pp. 57-62.*

Worboys, M. F., 1994, Object-oriented approaches to georeferenced information. *International Journal of Geographical Information Systems*, 3, 385–399.

Worboys, M. F., 1995, *GIS: a computing perspective* (London: Taylor & Francis).

Worboys, M., Hearnshaw, H., and Maguire, D., 1990, Object-oriented data modeling for spatial databases. *International Journal of Geographical Information Systems*, 4, 369-383.