Open Geospatial Consortium

35 Main Street, Suite 5
Wayland, MA 01778
Telephone: +1-508-655-5858
Facsimile: +1-508-655-2237

Editor:
Telephone: +1-703-830-6516
Facsimile: +1-703-830-7096
editor@opengeospatial.org

The OpenGIS® Abstract Specification
Topic 6: The Coverage Type and its Subtypes

Version 4

OpenGIS® Project Document Number 00-106.doc

Revision History

Date	Description
31 March 1999	Carry forward 98-106r2 as 99-106; update for new document template and 1999 copyrights; refer to Section 1.3 for additional history of this document.

This page is intentionally left blank.

Table of Contents

1. Introduction

1.1. The Abstract Specification

The purpose of the Abstract Specification is to create and document a conceptual model sufficient enough to allow for the creation of Implementation Specifications. The Abstract Specification consists of two models derived from the Syntropy object analysis and design methodology [1].

The first and simpler model is called the Essential Model and its purpose is to establish the conceptual linkage of the software or system design to the real world. The Essential Model is a description of how the world works (or should work).

The second model, the meat of the Abstract Specification, is the Abstract Model that defines the eventual software system in an implementation neutral manner. The Abstract Model is a description of how software should work. The Abstract Model represents a compromise between the paradigms of the intended target implementation environments.

The Abstract Specification is organized into separate topic volumes in order to manage the complexity of the subject matter and to assist parallel development of work items by different Working Groups of the OGC Technical Committee. The topics are, in reality, dependent upon one another— each one begging to be written first. *Each topic must be read in the context of the entire Abstract Specification*.

The topic volumes are not all written at the same level of detail. Some are mature, and are the basis for Requests For Proposal (RFP). Others are immature, and require additional specification before RFPs can be issued. The level of maturity of a topic reflects the level of understanding and discussion occurring within the Technical Committee. Refer to the OGC Technical Committee Policies and Procedures [2] and Technology Development Process [3] documents for more information on the OGC OpenGIS™ standards development process.

Refer to Topic Volume 0: Abstract Specification Overview [4] for an introduction to all of the topic volumes comprising the Abstract Specification and for editorial guidance, rules and etiquette for authors (and readers) of OGC specifications.

1.2. Introduction to The Coverage Type and its Subtypes

1.2.1. Review of Topic 1: Feature Geometry

The reader is encouraged to read Topic 1 of the OpenGIS™ Abstract Specification [5] for background information.

1.2.2. The Structure of the Abstraction Specification

This Section 2 of this document provides an "essential model" of coverages, in the sense of [1]. Section 3 of this document provides the "abstract model" in the sense of [1].

1.2.3. Why Coverages are Important

GIS coverages (including the special case of Earth images) are two- (and sometimes higher-) dimensional metaphors for phenomena found on or near a portion of the Earth's surface.

Fundamentally, coverages (and images) provide humans with an n-dimensional (where n is usually 2, and occasionally 3 or higher) "view" of some (usually more complex) space of geographic features. In our setting, the "view" will be geospatially registered to the Earth.

It is often useful to think of the spatial domain of a coverage as a "viewport" on a video screen, and to imagine that rules apply to the vector values assigned by the ***C_Function*** to color the viewport so that one can *see* the underlying phenomena.

The power of coverages is their ability to model and make visible spatial relationships between, and the spatial distribution of, earth phenomena.

1.2.4. Coverage

A coverage is a special case of (or a subtype of) feature.

Figure 1-1 shows that features with geometry and coverages are two subtypes of the supertype: feature. Other feature subtypes may not be directly associated with any geometry at all.

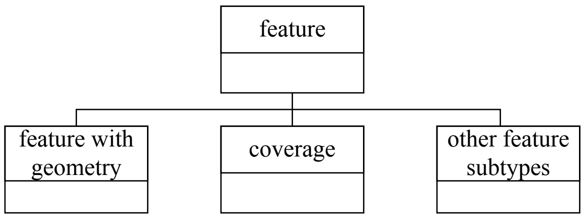


Figure 1-1. Feature subtypes

A feature with geometry has one or more properties taking values that are OpenGIS geometries (as defined in Topic 1, and in the Implementation Specifications.)

A coverage has a property usually named "***Coverage_function***" whose value is a ***C_Function*** (which stands for Coverage Function). The ***C_Function*** is a function that has some spatial domain as its domain, and its range can be any value set. Most generally, the range is a set of homogeneous tuples. For responses to OGC Request 5, the range may be simplified to be a collection of homogeneous vectors (that is, homogeneous tuples whose coordinates are numeric).

A coverage may have more than one property taking a ***C_Function*** as a value.

1.2.5. Coverage, a Generic Name

For simplicity, as well as by common usage, in this specification we will use the term "coverage" in a generic sense, to include "image," "map," "field" and so on.

Also, for simplicity in this version of the Essential Specification, we omit a detailed discussion of the temporal dimension in coverages. The temporal settings will be addressed in a future version of this document.

1.2.6. The Coverage as a Feature, and as a Feature Collection

A coverage can be designed to represent a single feature, or a set of features. Our discussion is often written to support the case of a coverage modeling many features. It is easily modified to the case where a coverage models a single feature. For example, a coverage may have a spatial domain containing a single parcel, or an entire platt. In the latter case, the coverage may be treated as modeling a single feature (the platt), or as a collection of features (the collection of parcels.)

This is not very different from the situation with features with Geometry. Consider for example the feature collection consisting of all the roads in Fairfax County. This collection can also be treated as a single feature: the road network administered by Fairfax County. At the essential level, we should be comfortable moving back and forth between any of the concepts in Figure 1-2 whenever it makes sense to do so.

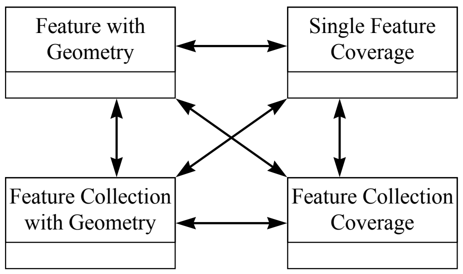


Figure 1-2. Alternate Views of Features and Feature Collections

1.2.7. Comparison of a Feature with Geometry and a Coverage

Figure 1-3 presents a summary of the relationship between a feature with geometry, and a coverage.

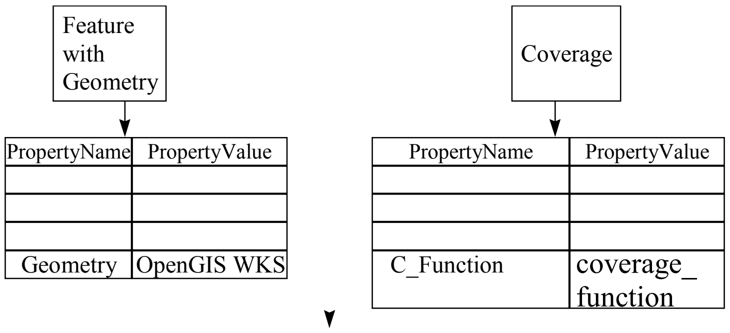


Figure 1-3. The Relationship between a Feature with Geometry and a Coverage

This topic sometimes assumes that the coverages and images are inherently two-dimensional. However, as we will see, it is easy to replace the two-dimensional setting with an n-dimensional one.

1.3. Status of The Coverage Type and its Subtypes

At the August 1997 meetings in Cambridge, England, the Technical Committee voted to issue RFP 5, Coverages, on January 5, 1998, with a 240 day interval from release to first submissions.

At the San Rafael meetings, the Technical Committee agreed to proceed with two staggered RFPs. The first would seek implementation specifications on three coverage subtypes, all of them of ***Grid*** type. A second RFP would add several additional subtypes.

The 98-106r2 version of Topic 6 reflects the work of an ad hoc coverage working group that has been restructuring the requirement into a more formal presentation. Many of the figures in this version of Topic 6 are taken from their contributions. It was approved at the Munich meetings.

The 98-106r2 version of Topic 6 does not extend the scope or content of the prior baseline document. This version is rewritten to address clarity, formal structure, and to place the models of the types of coverages in the same order (more or less) as they will be needed in foreseen RFPs.

The 98-106r2 version of Topic 6 of the OpenGIS™ Abstract Specification, is the principal reference of OGC's Request Number 5, an RFP: Access to Open GIS Coverages [6].

The 99-106r1 version of Topic 6 extends the prior versions by the concept of features in (grid) coverage, and topological operators for binary relations between features. These extensions were approved in the San Bernardino meeting 4/2000. Some editorial corrections are included also. The 2000 version of this document is 00-106.

1.4. References for Section 1

[1] Cook, Steve, and John Daniels, Designing Objects Systems: Object-Oriented Modeling with Syntropy, Prentice Hall, New York, 1994, xx + 389 pp.

[2] Open GIS Consortium, 1997. OGC Technical Committee Policies and Procedures, Wayland, Massachusetts. Available via the WWW as <http://www.opengis.org/techno/development.htm>.

[3] Open GIS Consortium, 1997. The OGC Technical Committee Technology Development Process, Wayland, Massachusetts. Available via the WWW as <http://www.opengis.org/techno/development.htm>.

[4] Open GIS Consortium, 1999. Topic 0, Abstract Specification Overview, Wayland, Massachusetts. Available via the WWW as <http://www.opengis.org/techno/specs.htm>.

[5] OpenGIS™ Abstract Specification, OpenGIS™ Project Documents 99-100 through 99-116, available through www as <http://www.opengis.org/techno/specs.htm>.

[6] Open GIS Consortium, 1998. OGC Request Number 5, Core Task Force, Coverage Working Group, A Request for Proposals: Access to OpenGIS Coverages, Wayland, Massachusetts.

2. The Essential Model for The Coverage Type and its Subtypes

2.1. A Preview of Coverage Subtypes

The coverage type itself has many important subtypes. We list a few in Figure 2.1.9, and will discuss some of them in more detail later.

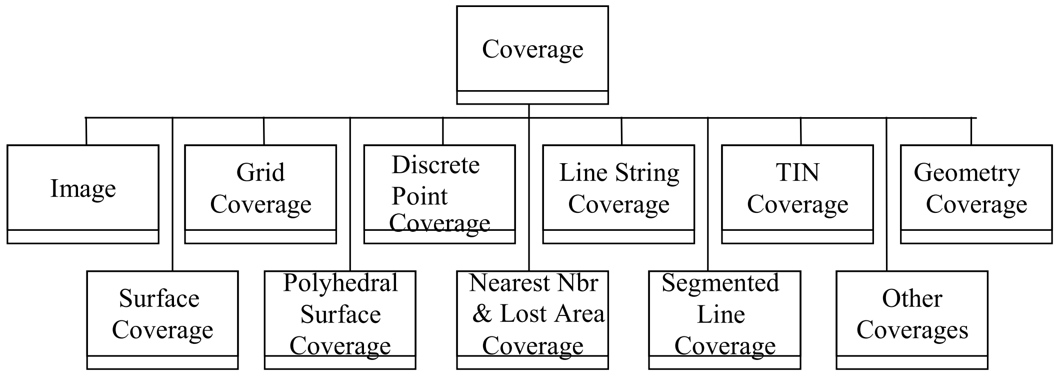


Figure 2.1.9. Coverage Subtypes

A detailed discussion of the Earth Image type is postponed to the next Topic of the Abstract Specification.

2.1.1. Spatial Domain

A spatial domain may be any geometry or collection of geometries. Usually, the geometry is accompanied with a spatial reference system, so that its points are associated with locations.

The most common spatial domain is a collection of points. It may be a finite collection of points, or may be the collection of all the points belonging to some specified geometry. A coverage with this kind of domain maps from points to values, or, more usually, to value vectors. Note that the term "vector" here means a tuple where each coordinate is numeric.

It is also allowed for the domain to consist of a collection of geometries. A coverage with this kind of domain maps from geometries to values, or to value vectors.

Commonly used spatial domains include collections of closed rectangles (or “pixels” or “tiles” in coordinate spaces, especially two-dimensional ones,) point sets, grids, triangles, and other collections of geometries.

The spatial domain of a coverage may be communicated by reference, or (if it is finitely representable) by values.

Hierarchies of coverage subtypes often correspond to hierarchies of their spatial domains.

2.1.2. Spatial Domains, a Closer Look

Spatial domains are often taken to be a collection of points or geometries inside a rectangle in Cartesian space whose sides are parallel to the axes, for convenience. The spatial domain is a metaphor for the “viewport” or window through which the “view” of the real world phenomenon is presented. We often mentally couple the spatial domain with points or geometries in a window on a computer display, although that coupling usually contains significant scaling, clipping, and offset technology.

Because today’s computer screens are rasters with pixel addresses, we often take the spatial domain to be the set of ordered pairs (i, j), where i and j are integers, and $A < i < N$, and $B < j < M$ for some integers A, B, N, and M. While the metaphor of the spatial domain as a set of points or geometries in a viewport is a useful one, there is no requirement that every coverage must exhibit this behavior.

2.1.3. The Range of a Coverage and the C_Function

The range of a ***C_Function*** is any value set. It is often the case that many associated functions sharing the same spatial domain need to be modeled. Therefore, the value set is usually represented as a collection of vectors.

C_Function: (geometry in spatial domain) $\rightarrow (v_1, v_2, v_3, \ldots, v_n)$

Note: For RFP5, treatment of vector valued ***C_Functions*** is mandatory. Treatment of ***C_Functions*** taking more general value types is optional.

In the setting of the previous paragraph, the ***C_Function*** models several (in fact, n) more elementary coverage functions, namely:

$f_1 : p \rightarrow v_1, \ldots, f_n : p \rightarrow v_n$ where p is a geometry in the spatial domain.

For example, the ***C_Function*** may assign to each point in a county the temperature, pressure, humidity, and wind velocity at noon, today, at that point. Every point in the county is mapped by the ***C_Function*** to a 4-dimensional vector.

For another example, a ***C_Function*** may have a domain consisting of seven county geometries. Each geometry might be a polygon (recall from Topic 1 that a polygon is a ring together with its interior). Each county geometry might map to the 5-dimensional vector: (area of county, perimeter of county, age of county manager, number of other counties sharing a boundary point with the county, population of county).

The range of the C_Function is the range of each component of the vector. The range of a ***C_Function*** must have a constant dimension over the entire spatial domain, and each coordinate of the vector must be of the same type over the entire spatial domain. That is, the range must be a homogeneous collection of vectors.

Each coordinate of value vectors usually will admit the value "not defined", or an equivalent code. Additional special values such as "overflow," and "out of range" are allowed at the discretion of specifiers at the implementation level.

2.2. The Basic Properties of Coverages

2.2.1. PointC_Function

A ***PointC_Function*** is the subtype of ***C_Function*** where the spatial domain consists of a collection of points. That is, geometries more complex than points do not occur in the spatial domain of a ***PointC_Function***.

2.2.2. PointValuePair

This section assumes that a ***C_Function***, f, is a ***PointC_Function***. The graph of a ***C_Function***, f, is the collection of ordered pairs (p, v) where

i. p is a point in the domain of f,

ii. v is a vector in the range of f,

iii. and f(p) = v.

The couples (p, v) in the prior paragraph are the instances of ***PointValuePairs*** associated with the ***C_Function***, f. That is, the ***PointValuePairs*** of a coverage are the entities in the graph of the coverage.

Many coverages are modeled by listing the finite collection of their ***PointValuePairs***.

2.2.3. GeometryC_Function

More general than ***PointC_Function***s, one can consider ***C_Function***s whose domain consists of a collection of unconstrained geometries. Such a ***C_Function*** would assign a value vector to each geometry in the collection. In this version of the Abstract Specification for Coverages and Images, we consider only a three types of geometries for the domain of a ***C_Function***: points, lines, and surfaces. ***PointC_Function***s have already been defined. ***LineStringC_Functions*** and ***SurfaceC_Function***s are next. Other types will be set aside for future versions of this document.

2.2.4. LineString C_Functions

A ***LineStringC_Function*** is ***C_Function*** whose domain consists of a collection of line strings. Usually the line strings are subsets of a coordinate plane, such roads, railroads, streams, and contours in the coordinates of a cartographic map.

One may, using appropriate coordinates or transformations, identify the map with a portion of the surface of the earth. In this setting, the domain of a ***LineStringC_Function*** is a set of one-dimensional regions on the earth (the road segments, for example) and the ***LineStringC_Function*** is a mapping of these regions of the earth to value vectors.

For example, one may consider the ***LineStringC_Function*** that (using an appropriate SRS,) maps road segments to the 4-dimensional vector: (address value at start of road segment, address value at end of road segment, county route number, length, identity of start node).

2.2.5. SurfaceC_Functions

A ***SurfaceC_Function*** is ***C_Function*** whose domain consists of a collection of surfaces. Usually the surfaces are subsets of a coordinate plane, such as county, state, and city polygons (including the interior of the polygons) in the coordinates of a cartographic map.

One may, using appropriate coordinates or transformations, identify the map with a portion of the surface of the earth. In this setting, a ***SurfaceC_Function*** is a mapping of regions of the earth to value vectors.

For example, one may consider the ***SurfaceC_Function*** that (using an appropriate SRS,) maps each state, county, and city polygon to the 4-dimensional vector: (perimeter, population, area, average temperature).

2.2.6. GeometryValuePair

One may generalize the notion of ***PointValuePair*** to that of ***GeometryValuePair***, as one generalizes from coverages whose spatial domains consisting of points to coverages with spatial domains having general geometries as elements. In this version of the Abstract Specification, we will consider only two extensions of ***PointValuePair***. These are the ***LineStringValuePair*** and the ***SurfaceValuePair***.

2.2.7. LineStringValuePair

As with ***PointValuePair***s, a ***LineStringValuePair*** is associated with the graph of a ***LineStringC_Function***. Given a ***LineStringC_Function***, f, an ordered pair (L,v) is a ***LineStringValuePair*** if

i. L is a line string in the domain of f,

ii. v is a vector in the range of f, and

iii. f(L) = v.

2.2.8. SurfaceValuePair

As with ***PointValuePair***s, a ***SurfaceValuePair*** is associated with the graph of a ***SurfaceC_Function***. Given a ***SurfaceC_Function***, f, an ordered pair (S,v) is a ***SurfaceValuePair*** if

i. S is a surface in the domain of f,

ii. v is a vector in the range of f,

iii. f(S) = v.

2.2.9. The Discrete Setting

The domains and ranges of ***C_Function***s up to this point have been unconstrained in cardinality. They could be infinite or finite sets. As an example of a ***C_Function*** with infinite domain and range, consider the ***C_Function***, f, that maps points in San Diego County to their temperature at noon today. Clearly, both the spatial domain and the range may take (essentially) infinitely many different values (up to the precision of the compute environment), assuming temperature is measured with a perfect instrument, and assuming both position and temperature are taken as type float.

"Discrete" is another word for "finite." A discrete set is one that can be referenced by listing its contents.

A discrete coverage is one whose ***C_Function*** has a finite domain (and small enough to be referenced by value.)

Note that all computers are finite state machines, so in a technical sense, all coverages modeled on computers are finite. However, it is not practical to list all possible states of today's computers, or even to list all possible values of a variable of type float, so we may take these settings as infinite environments.

Most coverages encountered in GIS are discrete, or are rooted in a discrete setting. As we will see, discrete coverages are often extended to infinite coverages using "evaluation" or interpolation techniques.

The opposite of "discrete" is either "infinite" or "continuous." By "continuous set," we mean a set that contains a subset that is a continuous, one-to-one image of the real numbers between 0 and 1. All line strings, polygons, and surfaces bounded by polygons are continuous in this sense.

2.2.10. Discrete Line Strings

A finite collection of line strings, such as streams, boundary segments, and utility lines, is a discrete set of line strings. It is a candidate spatial domain for a discrete coverage. Each line string is continuous, but here we are not concerned with the line strings as point sets. Rather, we are concerned with a finite set whose elements are line strings.

2.2.11. Discrete Surfaces

A finite collection of surfaces (a surface is a simple closed polygon together with its interior, see Topic 1 for more details), such as counties, states and federal forests, is a discrete set of surfaces. It is a candidate spatial domain for a discrete coverage. Note that each surface may itself be continuous, but there are only finitely many surfaces in the set. Here, the set contains surfaces, not points.

2.2.12. Discrete Geometries

The notion of discrete surfaces can be extended to more complex geometries, but this version of the Abstract Specification does not do so. More complex geometries will be added later as they are needed.

2.2.13. DiscreteC_Function

A DiscreteC_Function is a ***C_Function*** whose spatial domain and whose range are finite. A "finite" set in this setting means that an exhaustive listing of the elements of the set is feasible.

2.2.14. DiscretePointC_Functions

A ***DiscretePointC_Function*** is a subtype of DiscreteC_Function , and a subtype of ***PointC_Function***.

A ***DiscretePointC_Function*** is characterized by a finite domain consisting of points. A ***DiscretePointC_Function*** may be represented by a complete list of its ***PointValuePair***s.

Most coverage types are rooted in a ***DiscretePointC_Function***.

2.2.15. DiscretePointCoverage

A coverage is a DiscretePointCoverage if its C_Function is a DiscretePointC_Function.

2.2.16. Earth Images, a Subtype of DiscretePointC_Function

A special kind of (or subtype of) ***DiscretePointC_Function*** is the digital earth image. An everyday color digital earth image may be regarded as a mapping from a grid of points (representing pixel centers) to a space of (red, green, blue) vectors. Here, "red", "green" and "blue" might be one-byte integers representing the brightness (or radiance), of the scene in appropriate spectral bands at the pixel location. In this section, we do not discuss the Spatial Reference Systems that relate image pixel positions to earth locations (except for orthorectified images, where the relationship is obvious and trivial: the identity function). This relationship is to be found in Topic 7, Earth Images.

Note that for multispectral images, the dimension of the range vectors equals the number of spectral bands in the image.

2.2.17. DiscreteLineStringC_Function

A C_Function is a DiscreteLineStringC_Function if it is both a LineStringC_Function and its domain is discrete.

The set of line strings representing European passenger railroad segments (junction to junction) is an example of a discrete domain of line strings.

2.2.18. DiscreteLineStringCoverage

A coverage is a DiscreteLineStringCoverage if its C_Function is a DiscreteLineStringC_Function.

2.2.19. DiscreteSurfaceC_Function

A ***C_Function*** is a ***DiscreteSurfaceC_Function*** if it is both a ***SurfaceC_Function*** and its domain is discrete.

The set of surfaces representing the US counties, states, and federal forests is an example of a discrete domain of surfaces.

2.2.20. DiscreteSurfaceCoverage

A coverage is a DiscreteSurfaceCoverage if its C_Function is a DiscreteSurfaceC_Function.

2.2.21. Features in a Coverage

A coverage whose *C_Function* has a finite domain can have an elementary, discrete C_Function f_i that labels the discrete geometries of the coverage with feature labels. Then a feature in a coverage is the subset of discrete geometries with a common *DiscreteC_Function* value, the feature label or feature identifier.

The feature identifier of a geometry may be derived by mapping from other values of the same geometry, but also by mapping from local neighborhood, e.g., exploiting homogeneity, or exploiting discontinuities. Derivation of feature identifiers is discussed elsewhere (e.g., in Image Exploitation Services, Topic 15). The point here is that a coverage can consist of a set of features, or *FeatureCollection*s. Following the above definition, the labeling function represents a partition. If there are several C_Functions $f_{i\ldots k}$, then a projection to each function provides a partition.

Consider, e.g., a binary image. It consists of a foreground feature and a background feature, determined by the single C_Function f_l with the range of {0, 1}. As long as the image is considered as a grid of points, the geometry of foreground and background feature is a set of points.

2.3. An Overview of Coverages

2.3.1. Discrete and Continuous Coverages

The general theme of the following sections is that many useful continuous coverages are constructed using the following steps:

i. Begin with a finite set of ***PointValuePair***s. These form a kind of "seed" for the continuous coverage.

ii. Arrange or cluster the positions of the points in the ***PointValuePairs*** into a regular pattern of non-overlapping geometries. Patterns formed of rectangles or triangles work well. This process is called "gridding" or "triangulating," or tessellation.

iii. For each element of the pattern (that is, each rectangle or triangle), use an interpolation scheme to extend the domain of the coverage. Originally the coverage is defined only at the corners of the rectangles or triangles. Interpolation extends the domain to the interiors of the rectangles and triangles.

The result is that, starting with a discrete coverage, we have created a new coverage defined on a larger, and continuous, space.

2.3.2. The Notion of Evaluation.

Coverages exist to reveal the behavior of an earth phenomenon, or of many related earth phenomena. For this reason, coverages need to (compute, if necessary, and) expose their value vectors at points specified by applications with access to coverages.

Coverages often start with a finite number of observations of the phenomenon. This finite set of observations is called the sample set. The (sample location, sample value) observations are the ***PointValuePairs*** that are the "seed" of Section 2.3.1.

Evaluation is the computation and exposing of the value of the extended ***C_Function*** at a specified point that may fall between points in the sample set.

Evaluator objects may be introduced to accomplish the calculation (called evaluation) of proper value vectors, given input points or geometries. Evaluation usually requires the specification of an evaluation method.

Most evaluation methods are based upon interpolation techniques.

2.4. Special Type of Coverage: GridCoverages

2.4.1. Definition of a Grid

A ***Grid*** is an instance of Grid Geometry, where the structure of Grid Geometry is defined in Topic 1.

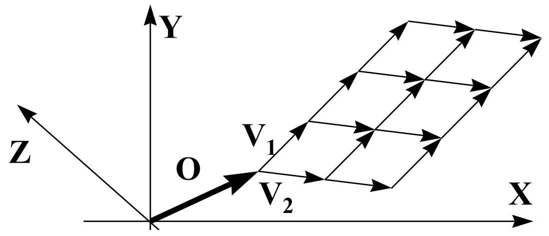


Figure 2-1. Origin of a Grid

In Figure 2-1, there is a 4 by 3 two-dimensional Grid in the 3-space determined by **X**, **Y**, and **Z**. The Grid origin is provided by the vector (or equivalently, the coordinate), **O**. There are two offset vectors labeled $\mathbf{V_1}$ and $\mathbf{V_2}$, (each of them is necessarily a 3-tuple), and the Grid is of size 4 - by - 3, (there are two numbers specifying the size of the Grid because the Grid is determined by two offset vectors).

The offset vectors in a Grid are linearly independent.

The Grid in Figure 2.4.1, like all Grids, is a finite collection of points. In this case the points are of the form: $\mathbf{O} + a\mathbf{V_1} + b\mathbf{V_2}$ where a and b are integers with $0 \le a \le (4 - 1)$ and $0 \le b \le (3-1)$.

The points in a Grid are sometimes referred to collectively as the Grid's Geometry, and individually as *vertices* (the singular is *vertex*) of the Grid, or *cell centers* of the Grid.

The vector (4, 3) is the *size* of the Grid, and we say the Grid has 4 rows and 3 columns.

In GIS, the most common Grid Geometries occur in a two-dimensional space, and are themselves two-dimensional. Grids are often formed using vectors parallel to the coordinate axes of cartographic projections as their offset vectors, for example. (See Topic 2 for a discussion of these spaces.) Three-dimensional Grids are not unusual, however.

2.4.2. Grid Values, Matrices

The basic assumption for ***GridCoverages*** is that some earth phenomenon has been observed at the vertices of a Grid. These observations are the set of Grid values, and they form a matrix (usually a vector- or tuple-valued matrix), called the ***GridValueMatrix***. The sequential enumeration, ***SeqForm,*** that applies to the Grid vertices also applies to the Grid values. Grid values must be of homogeneous type.

2.4.3. Sequential Enumeration

There are many ways to place the vertices into a sequence, for example: row dominant, column dominant, Morton, pyramid sequences, and others.

Each instance of a Grid must be associated with information that identifies its sequential enumeration, or ***SeqForm***.

A sequential format (or ***SeqForm***) is the name of a list data structure that specifies the sequential order of the vectors that form the Grid

“Row dominant” is the default ***SeqForm***, where “row” is defined in Topic 1.

2.4.4. A Grid is a Geometry

Because a Grid is a finite collection of points, it is a geometry, and a candidate for the spatial domain of a discrete coverage.

A feature may have geometry of Grid type.   Orchards and plantations often have trees planted in a Grid geometry, for example, so one could have an orchard feature whose geometry is a Grid.

A Grid geometry is a sequence of n vectors (or points) that collectively are the Grid, where n = (number of rows)*(number of columns), and the vectors are related according to the conditions in Section 2.3.2.

2.4.5. Lattice

A *lattice* is an infinite extension of a Grid. A two-dimensional lattice may be expressed as the collection of points $L = \{ p \mid p = \mathbf{O} + a\mathbf{V_1} + b\mathbf{V_2}$, a and b any integers$\}$.

Lattices extend in an obvious manner to any dimension.

Each Grid, by virtue of its origin and offset vectors, generates a unique lattice.

A two-dimensional Grid may be understood as a rectangle of points in a lattice, where the rectangle is aligned with the lattice offset vectors.

2.4.6. Cell Structures

Suppose we are given a Grid whose points are of the form: $\mathbf{O} + a\mathbf{V_1} + b\mathbf{V_2}$ where a and b are integers with $J \le a \le K$ and $L \le b \le M$. Then, each vertex **V** of the Grid is at one corner of a quadrilateral, where the other corners of the quadrilateral are $\mathbf{V} + \mathbf{V_1}$, $\mathbf{V} + \mathbf{V_2}$, and $\mathbf{V} + \mathbf{V_1} + \mathbf{V_2}$**.** These quadrilaterals are called "cells." Note that the association between vertices and cells is determined, not by the Grid (which is merely a collection of points), but by the vertices together with the offset vectors $\mathbf{V_1}$ and $\mathbf{V_2}$. Note that one row and one column of quadrilaterals are actually outside the convex hull of the Grid.

The collection of these quadrilaterals is sometimes also called a Grid, but in this Abstract Specification, the Grid is composed of the set vertices. The Grid is a set of points, not a set of quadrilaterals.

There is also a cell-center point of view. Shifting the quadrilaterals by half $-\mathbf{V_1}$ and half $-\mathbf{V_2}$ yields a collection of quadrilaterals (often called cells) centered on the Grid points. Users of digital images sometimes take this point of view, where it is assumed that each pixel value is the average radiance of the scene, where the average is taken over a cell centered on a Grid vertex..

The point is that Grids may model cell-centered (or cell averaged) or point-sampled phenomena. Moreover, there is a simple mapping between the two interpretations.

The rest of this treatment of ***GridCoverages*** assumes the point-sampled interpretation, but easily extends to the other settings.

2.4.7. Compatible Grids

Two Grids, G_1 and G_2, are said to be *compatible* if they generate identical lattices. Put another way, G_1 and G_2 are compatible if they have the same offset vectors (but perhaps in a different order), and the difference between their origins can be expressed as the sum of integer multiples of their common offset vectors.

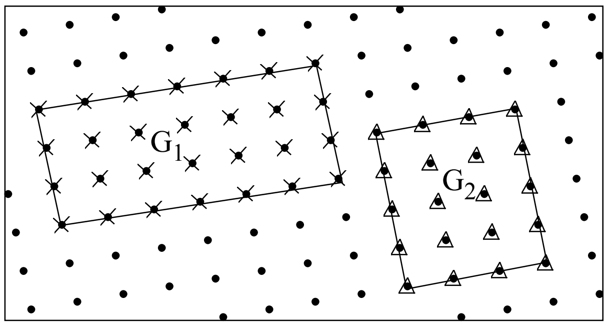


Figure 2-2. Two Grids sharing a common Lattice

2.4.8. The General Idea of GridCoverages

The notion of a ***GridCoverage*** is obvious from the ***GridCoverage*** structure in Topic 1, Feature Geometry.

The general idea of a ***GridCoverage*** is to

i. start with a ***DiscretePointC_Function*** where the points in the spatial domain form a Grid. The spatial domain of this function is the finite list of points forming the Grid.

ii. cluster the points into groups of four, as in paragraph 2.4.6, that surround a Grid quadrilateral with sides parallel to $\mathbf{V_1}$ and $\mathbf{V_2}$.

iii. For each quadrilateral, extend the ***C_Function*** to the interior of the quadrilateral using an interpolation algorithm.

The result is a new ***C_Function*** defined on the convex hull of the Grid

Note that digital earth images are a subtype of ***GridCoverage***. The set of (pixel point location, spectral vector) ***PointValuePairs*** form a ***DiscretePointC_Function***, in which the quadrilateral clustering is the obvious grouping of four points from adjacent rows and adjacent columns in the image. The image (that is, the coverage) can be extended to the convex hull of its pixel points using interpolation within each quadrilateral.

2.4.9. Evaluation and Interpolation in a GridCoverage

Interpolation of Grid values extends the spatial domain from a Grid to the convex hull of the Grid.

There are several interpolation algorithms:

i. Nearest Neighbor: for a given location, p, find the Grid vertex, V, closest to p, and assign f(p) = value vector at V

ii. Bilinear: for a given location, p, suppose p is contained in a Grid quadrilateral whose vertices are V, V + V1 , V + V2 , and V + V1 + V2 . Assume that the value vectors at these vertices are W1, W2, W3, and W4, respectively. There are unique numbers i and j , with $0 \le i < 1$, and $0 \le j < 1$ such that $p = V + iV1 + jV2$. Define the value vector at p to be: $W = (1-i)(1-j)W1 + (i(1-j)W2 + (1-i)jW3 + ijW4$

iii. Optimal: the optimum interpolation technique may be calculated using Wiener filters, and depends on factors that include the spectral power distribution of the scene, the sampling frequency, and the spectral power distribution of noise in the image [2].

iv. Bicubic: Cubic polynomials may be used to approximate optimum interpolation filters. See [2] for details.

These and other interpolation algorithms are allowed at the discretion of developers at the implementation level.

If no interpolation algorithm is specified, nearest neighbor is the default.

Nearest neighbor interpolation realizes a mapping from the spatial domain of grid points – the cell centers in 2.4.6 – to the domain of surfaces, the quadrilaterals in 2.4.6. Corresponding to the point spread function theory in image sampling, this mapping is useful to mediate between a grid coverage and a cell-centered point of view for digital images. Note that this mapping requires no change in representation. It needs only the attachment of the type of the interpolation function to the grid coverage.

2.4.10. GridEvaluator

If a separate service for Grid evaluation is provided, it is a GridEvaluator. The GridEvaluator is responsible for performing the interpolation method appropriate. Alternatively, the GridEvaluator may be an interface on ***GridCoverage*** objects.

2.4.11. Operations on GridCoverages

Besides evaluation, there are several other operations on ***GridCoverages***. The next several sections list some of them.

2.4.12. Families of GridCoverages

Two ***GridCoverages*** are said to belong to the same *family* if they share a common Grid geometry.

If G_1 and G_2 are ***GridCoverages*** in the same family with value sets S and T respectively, then G_1 and G_2 can be combined into a single coverage with value set $S \times T$. (Note: $S \times T$ represents the cross product of S and T, and is equivalent to all vectors of dimension 2 whose first coordinate is an element of S, and whose second coordinate is an element of T.)

2.4.13. Unions and Mosaics of GridCoverages

If their Grids are compatible, two ***GridCoverages*** can be combined into one by a *union* or mosaic operation.

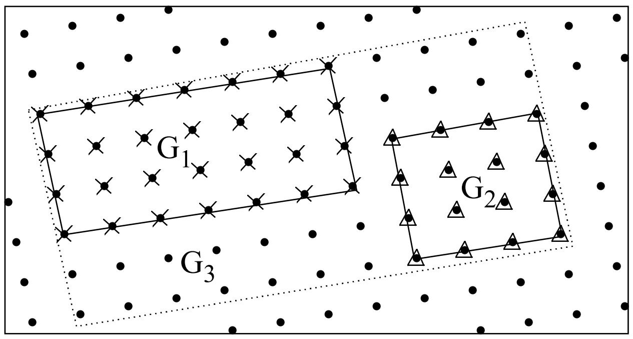


Figure 2-3. G_3 is the Union of G_1 and G_2

Figure 2-3 shows the union of two non-overlapping ***GridCoverages***. Note that points in G_3, but not in G_1 or G_2 can take the value "not defined" by this operation. In the case that G_1 and G_2 overlap, they can be taken as separate bands in a multi-band image, and then, if desired, merged into a single band using techniques of the next paragraph. Alternatively, the second of the ***GridCoverages*** in the union could over-write the first.

2.4.14. Band Separation and Combination

Suppose the value set of a ***GridCoverage*** is the collection of vectors of dimension four, where each coordinate is of type long. This ***GridCoverage*** can behave as if it were four separate coverages, each with a value set consisting of vectors of dimension one, where the single coordinate is of type long. This is achieved by projecting the vector values into each coordinate, one at a time. Each of the four coverages is called a *band* of the original coverage. This idea obviously extends to any dimension. Note that all coverage bands belong to the same family of coverages.

The combining of ***GridCoverages*** in the same family into a single coverage, and the separation of coverages into bands, are conjugate operations.

Coverages are sometimes exposed band-by-band, and sometimes vertex-by-vertex. As with Grid geometries (see Section 3.2.2), each ***GridCoverage*** must be associated with information that describes the sequential enumeration, ***SeqForm,*** of its values.

GridCoverages must support methods to convert from popular sequential enumerations (such as: BandSequential, BandInterleaf, and Morton) to internal representations, and to expose internal representations in at least one popular sequential enumeration.

2.4.15. Features in GridCoverages, and Spatial Relations between Features

GridCoverages modeling point-sampled phenomena can contain features whose geometries are of type "set of points" (see 2.2.21). Features of compatible grids have spatial relations corresponding to Topic 1 (Feature Geometry), but for sets of points these relations are primitive (points can be identical or not).

GridCoverages mapped to the cell-centered point of view (digital images, following 2.4.9.) can contain features whose geometries are of type "set of cells" or "set of pixels" (surfaces). Features of compatible grids have spatial relations corresponding to Topic 1 (Feature Geometry), if their geometry is considered as complete(d) cell complex [10].

In the latter case the completed geometric structure consists of 2D-cells (the pixel cells), 1D-cells representing the edges between the cells, and 0D-cells representing the nodes between the edges. This structure is implicit existing in the set of 2D-cells, i.e., in the *GridCoverage*, and can be made explicit by a convolution [10].

The methods of Topic 1 checking or determining a spatial relation between two features are applicable to features with geometries of gridded (regular shaped) cell complexes as well. Implementations of the methods need to get the additional ability to interpret a *GridCoverage*-feature of surface type as a cell complex. Otherwise special feature extraction services could make the implicit structure explicit. The regular structure simplifies evaluation to a logical overlay, if the two features are in compatible coverages. Features in incompatible coverages need to be transformed in a common grid.

Example. Consider **Error! Reference source not found.**: The grids G1 and G2, interpreted in the cell-center point of view, *touch* each other with a common border of two pixel edges' length.

2.5. Special Types of Coverage: GridCoverages and Their Semantics

GridCoverages are fundamental to many semantically different environments. ***GridCoverages*** can include digital orthorectified images, elevation matrices, and computer display windows, for example. Not every instance of these examples in enabled with mandatory ***GridCoverage*** interfaces, however.

2.5.1. GridCoverage as a Digital Orthorectified Image

A collection of points is a Grid only with respect to a particular spatial reference system. For example, a Grid may have offset vectors named Northing and Easting, where these coordinates carry the semantics of a particular Universal Transverse Mercator grid zone, with explicit spheroid, datum, and units.

Therefore, a Grid "fits" a map of proper scale, when the map is using the same spatial reference system as that of the Grid. (Here, one is thinking of the axes of the spatial reference system as the axes of a cartographic projection.) This means there is a rectangular coordinate system in the plane of the map that may be identified with the Grid coordinates. The Grid may be rotated relative to the map, and have its origin anywhere in the plane of the map.

2.5.2. GridCoverage as a DEM Coverage

A DEM is a digital elevation model, or digital elevation matrix. We assume a DEM is Grid of post locations (in some SRS coordinate system), with an elevation (relative to some vertical datum) assigned to each post. Thus, a DEM is a ***GridCoverage***, where the ***C_Function*** assigns an elevation (a vector of dimension 1) to each Grid point.

Typically, DEM post points form a Grid in latitude/longitude coordinate systems (often with post spacing from 1 to 3 seconds of arc in longitude and latitude; this varies with latitude).

A popular DEM (called Level 1 DTED in NIMA) occupies a one-degree by one-degree patch of the earth. Elevations are provided at posts with three-second spacing in latitude and longitude. Posts begin and end at integer values of longitude and latitude. The Grid structure therefore is 1201 by 1201 posts.

Another popular DEM (called Level 2 DTED in NIMA) occupies a 15-minute by 15-minute patch of the earth. Elevations are provided at posts with one-second spacing in latitude and longitude. Posts begin and end at longitude and latitude values that are integer multiples of 15 minutes. The Grid structure is therefore 901 by 901 posts.

Interpolation of Grid posts to perform evaluation of elevations at arbitrary points within the patch of each DEM may be performed by a variety of methods. Included is the bilinear algorithm.

2.5.3. GridCoverage as a Computer Display Window

A rectangular computer display window can be thought of as a rectangular array of pixels in the screen (row, column) coordinate system. (We may assume there is a Spatial Reference System that assigns each pixel to an earth location.)

The function that assigns each pixel to a color vector is therefore a ***C_Function***.

2.5.4. GridCoverage Semantics

As we have seen, ***GridCoverages*** can model many semantically distinct situations. Each ***GridCoverage*** must be associated with information that identifies the basic semantics of the vertex-to-value relationship.

As examples, three semantically distinct situations are described below. Each situation assumes values to be of type vector. This list of semantics may not be complete.

- Grid cell coverage: Each vertex represents a cell in a two- (or higher)-dimensional array of rectangular Grid cells. Each vertex value vector represents the constant coverage value within the corresponding Grid cell. Each coordinate in the value vector carries semantics that is explicitly stated.
- Images: Each vertex represents a pixel in a raster structure. Either the offset vectors are expressed in terms of the axes of a cartographic projection (that is, the image is an ortho-rectified raster), or the semantics of the offset vectors is not exposed (that is, the pixels in the image are not [yet] related to locations.) The pixel values are vectors: each coordinate represents an image band, and the value of the coordinate carries semantics (such as luminance or radar cross section) that is explicitly stated.
- Digital matrices: Each vertex represents a post location in a two-dimensional rectangular array, or matrix, of vector values. Each vertex's vector value provides the stored values of some vector of parameters at the corresponding Grid post. Each coordinate in the value vector carries semantics (e.g., elevation in meters above mean sea level, or temperature at noon in degrees Kelvin) that is explicitly stated.

2.6. Modification of GridCoverages (Simple Pixel Modification)

Interfaces for seven varieties of simple pixel manipulations are recognized. They are:

i. Radiometric Correction Service;

ii. Noise Removal Service;

iii. Contrast Enhancement Service;

iv. Spatial Feature Manipulation Service;

v. Multi-band Image Manipulation Service;

vi. Statistics and Histogram Calculation Services; and

vii. Special Transformation Services.

The following seven sections provide a summary of each service listed above. This specification does not list all the different techniques/methods associated with each service. The enumeration of the different techniques/methods is left up to the implementation specifiers of these services.

2.6.1. Radiometric Correction Services

Radiometric correction services are those that are required to correct an image for radiometric errors or anomalies that are introduced by factors such as:

i. Changes in scene illumination due to the terrain imaged,

ii. Atmospheric attenuation of the signal received at the sensor,

iii. Viewing geometry changes within a scene; and

iv. Sensor characteristics [7].

These services are usually applied to lower levels of image data. (i.e., those operations through Imagery Level 1R) (See[4] and [7] for an explanation of Imagery Levels). They do not include image to image balancing for mosaicking and change detection studies.

2.6.2. Noise Removal Services

Noise removal services are those that are required to remove unwanted anomalies in an image due to such conditions as sensor system operating anomalies, limitations of the signal digitization, or data recording process. In most applications, this service is implemented on lower levels of image data, similar to the radiometric correction service. An example of such a service is the correction of Landsat Multispectral Scanner (MSS) data that has the appearance of striping, due to a detector either malfunctioning or being saturated.

2.6.3. Contrast Enhancement Services

Contrast enhancement services are services used to modify the dynamic range of the digital values an image in order to improve its interpretability. Several different techniques of contrast stretching have been developed to implement this service, including the following:

i. Gray level thresholding;

ii. Density level slicing;

iii. Pseudocoloring; and

iv. Different techniques of contrast stretching.

2.6.4. Spatial Feature Manipulation Services

A Spatial Feature Manipulation Service should be composed of techniques that provide manipulations of an image over a selectable pixel range. The following is list of the more common manipulation methods that are considered part of this service:

i. Spatial filtering (both low-pass and high-pass);

ii. Convolution Filtering;

iii. Edge enhancement; and

iv. Fourier analysis.

2.6.5. Multi-band Image Manipulation Service

A Multi-band image Manipulation Service provides methods for processing of imagery from multi-, hyper-, and ultra-spectral sensors through the manipulation of individual bands or the combinations of bands. The techniques that make up this service have usually been implemented to enhance the information content of a set of images (or bands of an image). As defined in this proposal, this service is not intended to include classification services, which could, themselves, invoke this service. A common set of such techniques, but not an exhaustive list, includes the following:

i. Spectral ratioing;

ii. Principal and canonical component transformations;

iii. Spectral indexing;

iv. Intensity-hue-saturation color space transformation; and

v. Decorrelation stretching.

2.6.6. Statistics and Histogram Calculation Services

This set of services is intended to provide standard statistical computations that can be performed on images. They include, the following:

i. Minimum and maximum of an image;

ii. Descriptive Statistics of an image (e.g., the mean and standard deviation);

iii. Multi-band cross correlation matrices; and

iv. Histogram generation.

2.6.7. Special Transformation Services

The category referred to as, "Special Transformation Services", allows specifiers at the implementation level to include a set of services not listed in the Abstract Specification.

2.7. Special Type of Coverage: Triangulated Irregular Network (TIN)

The next several sections relate to TINs, or Triangulated Irregular Network coverages.

2.7.1. The Basic Idea of a TIN

The basic idea of a TIN is to extend a ***DiscretePointC_Function*** to a continuous ***C_Function*** on the convex hull of the points in the domain of the ***DiscretePointC_Function***, and to do this even when the discrete points are not in a rectangular array. ***GridCoverages*** extend a discrete coverage to a continuous one by exploiting the regular rectangular structure of the underlying Grid. TINs exploit a triangulation process to partition the convex hull of the points in the spatial domain into triangles. Each triangle is formed by three points in the ***DiscretePointC_Function***'s spatial domain. The trick is to cluster the given points into groups of three, so that the triangles formed by them partition the convex hull in a "most natural fashion".

The partitioning of space into triangles forms an irregular tessellation. It is often desired that the triangulation be performed so that the resulting triangles be as close to equilateral as possible; that is, to avoid long narrow triangles.

A triangular tessellation consists of:

i. A set of points, each associated with a (point, value) PointValuePair.

ii. A set of line segments, consisting of a pair of points in (i) joined by a line segment.

iii. A set of triangles, formed by three non-collinear segments in (ii) that partition of the region surrounding the points.

A set of Delaunay triangles satisfies these requirements. Delaunay triangles are most easily described in terms of a dual structure called Thiessen Polygons (also called Voroni polygons and Proximal Regions.)

Each triangle in a Delaunay triangulation (or in any triangular tessellation of a spatial domain) is a ***ValueTriangle***. That is, it carries values on each of its vertices. These values may be interpolated to yield values for each point within the triangle. The interpolation scheme usually used employs barycentric coordinates.

The result is a ***C_Function*** on the convex hull of the original set of points that is continuous, and extends the ***DiscretePointC_Function***.

2.7.2. Thiessen Polygons

A finite collection of points on a plane determines a partition of the plane into an equal number of polygons in a natural way. Imagine the plane to be a map of a large school district with many schools, where each school is a point. Suppose there is a rule that says a student who lives a point p must attend the school nearest point p. The school district boundaries that result are the Thiessen Polygons. Each Thiessen polygon contains exactly one point of the finite collection we started with.

Thiessen Polygons can be created with an algorithm: select nearest neighbor pairs of points, and draw the perpendicular bisectors between them, then assembling these bisectors into polygons that surround each point in the finite collection.

Or, one may imagine an empty balloon placed at each point in the finite collection. Slowly blow up all the balloons at the same rate. When two balloons touch, imagine them sharing a growing flat boundary that contains the perpendicular bisector of the line connecting the centers of the two balloons. When three or more balloons touch at a point, a node is formed that terminates that end of all the perpendicular bisectors that meet there. Continuing to blow up the balloons until all space is occupied yields the Thiessen Polygons. There is one polygon for each balloon (that is, one polygon for each point in the finite collection of points we started with), and one starting point in each polygon. Each polygon shares each of its edges with exactly one other polygon. Let us call two Thiessen polygons that share an edge “next door neighbor” polygons.

2.7.3. The Thiessen Polygon Network

A “network” is composed of

i. A finite collection of points that form the nodes of the network

ii. A collection of disjoint line strings that start and end at distinct nodes of the network

A Thiessen Polygon tessellation provides a network. The line strings are the straight “balloon edges” where two balloons share a common line, and the nodes are the points where these edges connect to other “balloon edges.”

2.7.4. Duality

The Thiessen Polygons form an irregular tessellation of the plane. There is a dual tessellation formed of triangles.

Note that each polygon in the Thiessen tessellation contains exactly one point from the original finite collection in its interior. (It plays the role of “school” in the above example, where the polygon is the school boundary.)

The dual tessellation is formed by the line segments that join two schools whenever the two schools share a common boundary point. That is, two schools are joined by a segment if their districts are “next door neighbor” polygons. Note that these line segments are perpendicular to the line forming their common boundaries, and are bisected by the common boundary.

The duality between Delaunay Triangles and Thiessen Polygons is reflexive, that is, they are duals of one another. See [5] for details.

2.7.5. Delaunay Triangles

It is easy to show that the dual tessellation of a Thiessen tessellation is formed of triangles. That is, it is a triangulation of the plane. This triangulation is called a collection of Delaunay triangles.

The Delaunay triangles are the generally preferred triangulation of a finite set of points.

In summary, if one is given a finite collection of points on the plane, and desires a Delaunay triangulation, one should follow the steps:

i. Form the Thiessen Polygons surrounding each point using the proximity condition.

ii. Form the dual tessellation, yielding Delaunay triangles.

2.7.6. ValueTriangleEvaluator

Each Delaunay triangle is a ***ValueTriangle***, that is, it is a triangle determined by three non-collinear vertices, and each vertex is associated with a ***PointValuePair***.

A ***ValueTriangleEvaluator*** is a service or interface that, when given any point in a ***ValueTriangle***, computes a value for that point based on an interpolation of the values at the three vertices. The next sections discuss a technique for doing this with barycentric coordinates.

2.7.7. Barycentric Coordinates

Let **P**, **Q**, and **R** denote the vertices of a triangle. We assume that the vertices are not co-linear. For any point, **S**, in the triangle, there is a unique triple of numbers, i, j, and k , with $0 \le i \le 1$, $0 \le j \le 1$, and $0 \le k \le 1$, and with $i + j + k = 1$, such that

$$\mathbf{S} = i\mathbf{P} + j\mathbf{Q} + k\mathbf{R}$$

The coordinates (i, j, k) are called the barycentric coordinates of **S**.

The name “bary-centric” comes from the fact that using the equation above, **S** is the center of mass of a triangle with point masses of size i, j, and k at the corners **P**, **Q** and **R** respectively. As one allocates a unit of mass to the three corners, the center of mass can occupy any point in the triangle. For details, see [6].

2.7.8. Interpolating a ValueTriangle

Given a value triangle composed of the ***PointValuePairs*** ($\mathbf{P}_1$, V_1), ($\mathbf{P}_2$, V_2), and ($\mathbf{P}_3$, V_3),, and a point, **S**, inside it, one may compute a value vector at **S** by following these steps:

i. Compute the barycentric coordinates of **S**: the coordinates are (i, j, k), where
   $\mathbf{S} = i\mathbf{P}_1 + j\mathbf{P}_2 + k\mathbf{P}_3$

ii. Assign the value $V = iV_1 + jV_2 + kV_3$ to the point **S**.

That is, V is the interpolated value at **S**, and (**S**,V) is a new ***PointValuePair***

Note that the vector arithmetic in step ii. of the algorithm above exploits the fact that each coordinate of the value vectors is of type numeric. (This is a property of the type "vector"). If there were coordinates of other types (e.g., strings), a rule must be stated how to evaluate these coordinates at **S**.

If **S** is not in the plane containing $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$, a message to that effect is generated.

2.7.9. TIN Coverages

A TIN is a coverage

i. With a ***PointC_Function*** defined by interpolating a given finite set of ***PointValuePair***s.

ii. Whose interpolation algorithm is based on interpolating ***ValueTriangles***.

iii. Whose ***ValueTriangles*** are determined by a triangulation process.

iv. Whose triangulation process may use Delaunay triangulation and its dual Thiessen Polygons.

v. Whose spatial domain is usually the convex hull of the points of the given set of ***PointValuePair***s.

TINs are popularly used to model continuous phenomena given a sample set of ***PointValuePair***s. Continuous phenomena include elevation, temperature, salinity, magnetic north vector, and so on.

2.8. Special Type of Coverage: Nearest Neighbor

2.8.1. The Basic Idea of a Nearest Neighbor Coverage

The basic idea of a Nearest Neighbor Coverage is to extend a ***DiscretePointC_Function*** , g, to a discontinuous step function, f. The step function, f, is to be defined on the convex hull of the points $\{x_1, x_2, \ldots, x_n\}$forming the domain of the ***DiscretePointC_Function***. Let $\{(x_1, v_1), (x_2, x_2), \ldots, (x_n, v_n)\}$ be the sample points (that is, the ***PointValuePair***s) that define the ***DiscretePointC_Function***, g. For each point, p, in the convex hull of the spatial domain, define $f(p) = v_i$, where x_i is the point in $\{x_1, x_2, \ldots, x_n\}$ closest to p.

Note that the Thiessen polygons generated by the set $\{x_1, x_2, \ldots, x_n\}$form the "steps" of the Nearest Neighbor Coverage step function.

A nearest neighbor coverage is one whose coverage function is a Nearest Neighbor step function.

2.9. Special Type of Coverage: Lost Area Interpolation

2.9.1. The basic idea of Lost Area Interpolation

The basic idea of Lost Area Interpolation is to extend a ***DiscretePointC_Function***, g, to a continuous function, f. The function, f , is to be defined on the convex hull of the spatial domain of g. Let $\{x_1, x_2, \ldots, x_n\}$ be the spatial domain of g, and let $\{V_1, V_2, \ldots, V_n\}$ be the Thiessen polygons generated by the set $\{x_1, x_2, \ldots, x_n\}$, where V_i, contains x_o.

Suppose it is desired to calculate f(p), where p is in the convex hull of $\{x_1, x_2, \ldots, x_n\}$. Begin forming the Thiessen polygons generated by $\{x_1, x_2, \ldots, x_n\}$, and then by adding p to the set $\{x_1, x_2, \ldots, x_n\}$, and forming the Thiessen polygons for the set of n+1 points: $\{x_1, x_2, \ldots, x_n, p\}$. The two sets of polygons are identical, except that each of the "next door neighbor" polygons to the polygon containing p "loses area" to the new polygon containing p. Suppose that polygon P_i lost area V_i .

The interpolation method is to form the weighted average, so each value contributes to the value at p according to the amount of area its polygon lost to the polygon at p. More formally, if

i. The ***DiscretePointC_Function*** is characterized by the ***PointValuePairs*** (or sample points): $\{(x_1, v_1), (x_2, x_2), \ldots, (x_n, v_n)\}$.

ii. The Nearest Neighbor Thiessen polygons to p are $\{V_1, V_2, \ldots, V_k\}$ among the Thiessen polygon set formed by $\{x_1, x_2, \ldots, x_n, p\}$.

iii. The corresponding Thiessen polygons generated by $\{x_1, x_2, \ldots, x_n\}$ are $\{V'_1, V'_2, \ldots, V'_k\}$

iv. The area lost by the ith polygon is $V'_i - V_i$

v. The total area lost is $\Sigma\ (V'_i - V_i)$ where the sum is over i from 1 to k (that is, the sum is over all polygons that lost area to the polygon containing p). Note that this sum is the same as the area of the Thiessen polygon containing p.

Then the interpolated value at p is

$$f(p) = (\Sigma\ v_i * (V'_i - V_i)) / \Sigma\ (V'_i - V_i)$$

where the summations are over the same range: i = 1, …, k.

A coverage whose ***C_Function*** is based on lost area interpolation is called a lost area interpolation coverage.

2.10. Special Type of Coverage: Segmented Line Coverages

The next several sections provide abstract models for Segmented Line Coverages

2.10.1. The Idea Behind Segmented Line Coverages

Segmented Line Coverages are used to model phenomena that vary continuously or discontinuously along line strings, or along line strings in the context of a network.

The idea is to create a ***PointC_Function*** whose spatial domain consists of all the points in all of the line strings in the network. The value vectors returned by the ***PointC_Function*** are used to reveal the character of the phenomena being modeled.

For example, the ***PointC_Function*** might model road attributes that may change smoothly or sharply, and also model "road furniture," that is, discrete or continuous objects that are associated with roads, such as signs, signals, and curbs.

An essential ingredient is a tool to unambiguously reference a position along a line string that belongs to the network. This is the role of the arc-length parameterization.

2.10.2. Line Strings and Parameterizations

General one-dimensional curves are the proper setting for the following discussion, but for this version of the Abstract Specification, Line Strings are the only one-dimensional objects recognized. (It is intended to add splines, polynomial curves, ellipses, and other curves at a later time.)

Line Strings in the context of Segmented Lines are assumed to be simple. That is, in the context of Segmented Lines, Line Strings do not self-intersect.

A parameterization of a line string, S, is a one-to-one continuous function, q, from a closed interval [0,L] of the real line onto the points S. That is,

i. q maps continuously and one-to-one from [0,L] onto the points of S.

ii. q(0) is one end of S; it is called the Start of S.

iii. q(L) is the other end of S; it is called the End of S.

iv. as x moves smoothly from 0 to L, q(x) moves smoothly along the string S from its Start to its End.

Generally, length is used as the parameter. That is, q is usually given the additional requirement:

- The length along S from Start to q(x) is x. This implies that L is the length of the Line String.

Because the function, q, captures the geometry of the line string S, segmentation may be performed on the domain of q, that is, on the interval [0,L].

2.10.3. Segmentation and Segments

The following discussion, of necessity, presents mathematical details for the representation of values along a segmented line string. These details are not requirements on the implementation of segmented line functionality. Instead, they are presented here to convey the required behavior of segmented line coverages. Alternate (and superior) implementations are permitted.

Let q be a parameterization of a line string S. Assume q maps [0,L] onto the points of S. A segmentation of S is a set $P_0, P_1, \ldots, P_n$, of points in [0,L] with

i. $P_0 = 0$

ii. $P_0 \leq P_1 \leq \ldots \leq P_n$

iii. $P_n = L$

This segmentation breaks the interval [0,L] into n pieces. If $P_k = P_{k+1}$, we say the kth piece is a "special" point.

The segmentation can be interpreted as breaking the line string S into segments and "special" points. The points along S where the breaks occur are:

i. $q(P_0)$ = Start of S

ii. $q(P_1), \ldots, q(P_{n-1})$ (these are points along S, in order of the parameterization)

iii. $q(P_n)$ = End of S

A segmentation also assigns a value vector to each end of the segment intervals it forms, and also assigns a value vector to its special points.

A typical segment is associated the portion of the line string between two consecutive points in a segmentation (a segment may also be a special point). The kth segment (starting with the 1st) is associated with the portion of the line string $q[P_{k-1}, P_k]$. That is, the kth segment has:

i. A start parameter, P_{k-1}

ii. A start point, $q(P_{k-1})$.

iii. A start value vector, S_{k-1}

iv. An end parameter, P_k

v. An end point, $q(P_k)$

vi. An end value vector, E_{k-1}

In the case that the start parameter and end parameter are equal (that is, we are at a special point in the segmentation), the last three items in the list in the prior paragraph are redundant, as they are equal to the first three items. The advantage of allowing special points is that it provides for isolated phenomena to be modeled, such as the occurrence of a fire hydrant along a road, together with a vector of properties of the hydrant..

$S_{k-1}$and E_{k-1} are the value vectors assigned by the segmentation to the start points and end points respectively, for the kth segment. They are equal if the kth segment is a special point.

2.10.4. Segment Evaluator and Interpolation

Value vectors, $S_{k-1}$and E_{k-1} are assigned to each end point of a segment in a segmentation. Value vectors are also provided at isolated points. A segment evaluator is a service or interface on a segment that computes and yields a value vector for any given point in the segment.

Suppose a point P in the line string S is given. The mission of a segment evaluator is to calculate and return a value vector that is specific to P. There are three cases.

If P happens to be a special point, we take the unique value vector of that special point as the value vector of P.

Next, we consider the case where P is a common endpoint of two segments (P will be the end point of the (k-1)th segment, and the start point of the kth), but where P is not an isolated point. In this case, the default evaluator takes the value vector at P to be the average of the two value vectors assigned at that point. That is, the value vector at P is one half E_{k-1} plus one half S_k.

It remains to consider the case where P falls in the interior of an interval. In this case one normally performs a linearly weighted average of the value vectors at the start and end of the interval containing P (a linear interpolation). Other interpolation methods are allowed. The following paragraph provides the details when linear interpolation is desired.

Suppose that P falls in the kth interval of S. Suppose further that q(p) = P, that is, that p is the parameter corresponding to P. Then we have $P_{k-1} < p < P_k$. Define

ValueVector$(P) = (P_k – p)/(P_k - P_{k-1}) S_{k-1} + (p - P_{k-1})/ (P_k - P_{k-1}) E_{k-1}$.

Other interpolation methods are permitted.

In linear interpolation, we are exploiting the fact that each coordinate of the value vectors is of type numeric, so that the arithmetic may be executed. If certain coordinates were not numeric, rules for computing their values at an arbitrary point P in S must be provided.

2.10.5. The SegmentedLineCoverage

A ***SegmentedLineCoverage*** is composed of a network of line strings, each of which is segmented.

A ***SegmentedLineCoverage*** must be able to expose the number of segmented line strings of which it is composed, and be able to expose these segmented line strings by some index or iterator.

A ***SegmentedLineCoverage*** should expose the Spatial Reference System by which its points (coordinates) are related to earth locations.

A ***SegmentedLineCoverage*** exposes an interpolation type to be used in segment evaluation. The default is linear interpolation.

A ***SegmentedLineCoverage*** exposes a relationship with a service (or an interface) to perform segment evaluation (using the mandated interpolation type).

The ***C-function*** associated with a ***SegmentedLineCoverage*** is the mapping from points in the union of the line strings, S, to value vectors. This mapping is partly proscribed (at special points, and at the endpoints of intervals), and partly computed (using the segment evaluator.) If P is a point not on S, then the ***C_Function*** maps P to "not defined."

2.11. Special Type of Coverage: Images

2.11.1. Images and Coverages

Coverages and images are somewhat difficult to treat in a unified manner, partly because there are many frequently encountered special cases that beg to be treated separately for reasons of efficiency or elegance. At this essential model level, we resist temptations to consider special cases, and address only the general setting. We will pause in our discussion to point out familiar landmarks as we pass them, however.

Another reason coverages and images are difficult to treat in a uniform manner is the large differences in granularity and content between certain instances of images and coverages.

Nevertheless, this Topic of the Abstract Specification shows that the single type of coverages-and-images (which we denote generically as "coverages") is appropriate to conceptualize as one important class of objects.

Because Earth Images have many special interfaces, they are to be treated in detail in the next Topic, Topic 7: Earth Images.

2.11.2. Images as GridCoverages

A digital image, or any raster-structured object taking values on a Grid of points, such as a matrix, may be considered a GridCoverage. The weakness in this view is that many (non-orthophoto) images have complex Spatial Reference Systems that require special attention. Complex photogrammetric Spatial Reference Systems are to be treated in Topic 7, Earth Imagery.

2.11.3. Some Examples of Image Coverages

The list of coverage examples below illuminates both the breadth of the image coverage type, its span of granularity, and its importance.

i. A digital image of a house (24 bit color, line interleaved).

ii. A forward looking infrared (FLIR) sensor image generated in a jet fighter.

iii. A computer display of parcel boundaries .

iv. A scanned black-and-white paper map with its spatial reference system (relating scan coordinates to the map Grid) .

v. A scanned schematic drawing (here, the spatial reference system might model the abstract coordinate system imposed by the scanner, and a mapping between certain points on the scanned drawing to real world locations).

vi. A patch of Earth terrain where only radar cross-section is "seen"

vii. A scanned image of a geologic cross-section map.

At first these seem to be distinct object types. Yet at the essential level, all are objects of the same type: image coverage.

2.11.4. Examples of Spatial Reference System Factors in Image Coverages

The Spatial Reference System necessary to model natural images can be very complex. It may need to take into account such factors as:

i. motion compensation (allowing a moving perspective center with a push-broom sensor)

ii. terrain relief displacement and camera distortion

iii. atmospheric and air-water-interface defraction

iv. the theory of conformal transformations for cartographic projections

v. the theory of phase history processing for synthetic aperture radar

vi. the rules of classical cartography

vii. more complex cartographic rules defining aggregation, displacement, omission, names placement, generalization, …

viii. the composition of several factors, perhaps caused by the iterative use of images and coverages as the basis for new images and coverages

ix. exotic image formation technologies.

For those coverages using the visible earth surface as the spatial domain, and using photography (or digital imaging) as the image coverage generation process, the study of the Spatial Reference System is called photogrammetry (the study of mapping object space into image space).

In fact, the notion of coverage includes an even larger part of photogrammetry. A photograph (or a digital image) may itself be used as the subject of another photograph, and in this setting, Spatial Reference Systems can be extremely deep and complex.

Another Spatial Reference System could model the geometry involved in photomosaicking.

As this Abstract Specification matures, the essential and abstract models for image coverages will be moved from this Topic 6, into Topic 7, The Earth Image.

2.11.5. Types of Spatial Reference Systems Supporting Image Coverages

For those coverages using the points that belong to the geometries in a feature collection as the Project World, the Spatial Reference is often taken to include a local vertical projection. (A local vertical projection changes coordinates, if necessary, to (ϕ, λ, h) [that is, longitude, latitude and elevation],and then just drops the h.)

As an example how the general notion of a Spatial Reference System might be formalized, consider the Central Projection subtype of Image Coverage. This subtype is frequently used in photography. That is, let us assume that we are modeling a three-dimensional space with coordinates (x,y,z), with a two dimensional image whose Cartesian coordinates are (r,c), for the row and column coordinates of the image plane in the camera. Several additional relationships are needed to compute the central projection equations, such as the location $(x_0,y_0,z_0)$of the central projection point, the tilt and pitch of the principal axis, and the focal length. The situation is shown in Figure 2-4. Given these values, a SRS can be computed such that if (x,y,z) are the coordinates of a photo-identifiable point on the earth, then the SRS associates (x,y,z) to the point (r,c) , where (r,c) are the coordinates of the image of that point on the photograph. See [2].

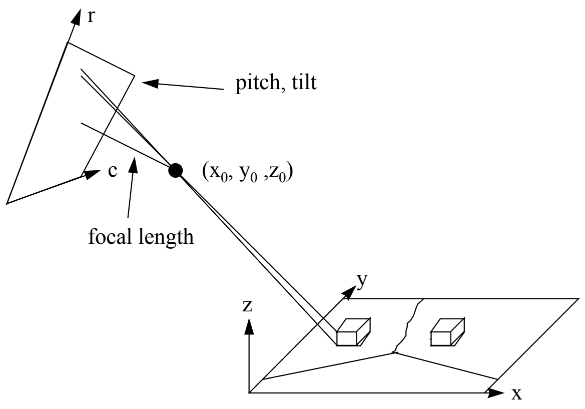


Figure 2-4. The Central Projection Function *G*

In this case, the SRS might embody the form of the classic projective equations shown below, for appropriate values of the coefficients a_1, …, a_8. An interface for the SRS could be defined by specifying the sequence and types of these coefficients.

$$r = (a_1x + a_2y + a_3)/(a_7x + a_8y + 1)$$

$$c = (a_4x + a_5y + a_6)/(a_7x + a_8y + 1)$$

Equation 2-1. The Central Projection Equations

It is not necessary in general to have a formal, closed form, equation for the relation modeled by a Spatial Reference System. Often in GIS technology we do not have a complete definition of an SRS. The next section provides an example.

2.11.6. The Stored Function Approach to Spatial Reference Systems

A collection of tuples (x,y), where x is a point in a earth spatial reference system, and y is a point in the spatial domain of a coverage, is said to be a set of checkpoints.

Often, the domain of a spatial reference system is the collection of points and corners in our Project World where checkpoints are given.

If earth SRS coordinates are 3-dimensional, and if the spatial domain of a coverage is 2-dimensional, then each checkpoint determines a 5-tuple (namely, the 3-tuple specifying a location in the Project World, followed by its corresponding 2-tuple in the coverage domain. A finite collection of such 5-tuples is a "stored function" formulation of a spatial reference system. This idea extends to any other combination of dimensions. The important point is that an interface to spatial reference systems can be built by formalizing the structure of its stored function.

2.11.7. Modeling a Spatial Reference System as a Rational Function

There are many ways to extend (interpolate or extrapolate) a stored spatial reference system to a function defined on all points mapping into a spatial domain. In the discipline of photogrammetry, rational functions are often used. A rational function, *Q*, is the quotient of polynomial functions. In photogrammetry, we are interested in functions that express the (r,c) coordinates of image space (the spatial domain of the image coverage) in terms of the (x,y,z) coordinates of object space (the spatial reference system in this setting.). Equation 2-2 is the general case.

$$Q(x,y,z) = (r,c) \quad \text{where } c = \frac{\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=0}^{n} a_{ijk}(x^i y^j z^k)}{\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=0}^{n} c_{ijk}(x^i y^j z^k)} \quad \text{and } r = \frac{\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=0}^{n} b_{ijk}(x^i y^j z^k)}{\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=0}^{n} c_{ijk}(x^i y^j z^k)}$$

Equation 2-2. A rational function expression of a coverage generation function

The point is that an interface to a spatial reference system can be built on the standardization of the expression of the coefficients a, b, and c.

2.12. Extending the Range of Coverages from Vectors to Tuples

2.12.1. Vectors

Vectors have the same structure as coordinates. Each coordinate value in a vector is usually taken to be measurement along a spatial axis, and is of homogeneous type. (As examples, coordinate types may include: int (32 bit long integers) or float (64 bit IEEE floating point)). A vector of dimension n has n coordinates.

2.12.2. Tuples

A tuple is an ordered sequence of values, where each coordinate is of homogeneous type. The type used by a particular coordinate might be non-numeric, such as String, or ***OGCGeometry***.

2.12.3. Tuple-Valued Coverages

Up to this point, the coverages under discussion took values from a collection of vectors. Following sections discuss coverages whose values come from a homogeneous collection of tuples.

2.13. Schema Mapping: How Coverages Model PropertyNames and PropertyValues

2.13.1. Coverages Can Model Features

Suppose C is a coverage, and f is a ***C_Function*** of C. (Note, a coverage may have more than one ***C_Function***.) Then the domain of f is some spatial domain. So far, we have allowed this spatial domain to be quite abstract. Usually, however, the spatial domain consists of the points (or, of the geometries) of features in a feature collection of interest. Note that we are assuming that the features of interest are features with geometry. In this section we also assume that all the features of interest are of the same type; that is, they have the same PropertyName/PropertyValue schema.

For the moment, suppose that the graph of f is a collection of ***PointValuePairs*** (so f is a ***PointC_Function***). That is, we are supposing that there is a feature collection of interest, and the set of points that belong to the geometries of the features in the collection is the spatial domain of a coverage.

The value of f, the ***C_Function***, at a point, p, is usually constructed so that it reveals or models the properties of the feature (or features) (if any) whose geometry occupies the point p. That is, f(p) is usually determined by the properties of a feature, F, where p is a point in the geometry of F.

In fact, one often defines f(p) to be the PropertyValue tuple, using the schema of the feature, F, whose geometry contains p.

2.13.2. The Fundamentals of Schema Mapping

In the usual case, the ***C_Function***, f, maps points p in the spatial domain to tuples. The tuples are the columns of PropertyValues of the features in the feature collection of interest. The point p maps to the PropertyValue tuple of feature F if p is a member of the geometry of F. This approach maps the schemas of features to the schemas of coverages, and is called *schema mapping*.

It is usually assumed that the features are all of the same type, so that the tuples in the range are homogeneous. If the features in the feature collection of interest are of different types, it may be necessary to identify "rows of interest" in the schemas of the various types of feature with rows in the coverage value tuple, so that, at least in these rows, the values are homogeneous.

Note that the geometries of the features of interest must all live in a common coordinate space, S. If p is an element of S, but p does not belong to the geometries of any of the features of interest, it may be of interest to define or calculate a value tuple for f(p). This is especially true if p is in the convex hull of the geometries of the features of interest. There are several approaches:

i. Require that every PropertyValue tuple coordinate have a value named "not defined", and define f(p) to be the tuple with value "not defined" in each coordinate.

ii. Interpolate a value for f(p) from nearby points. When this approach is selected, there must be a method for the coverage to reveal the interpolation method used.

iii. Create a value tuple for f(p) using some other rule. When this approach is selected, there must be a method for revealing the rule employed.

It is also possible that there is a point, p, that belongs to the geometries of two or more features in the collection of features of interest. Again there are several approaches:

i. Restrict the geometries on the features of interest until they no longer overlap.

ii. Define f(p) to be the average of the value tuples, where the average is taken over all the features where p is a point in that feature's geometry. (Of course, this approach demands that each coordinate in the value tuple can be averaged in some way.)

iii. Create a value tuple for f(p) using some other rule.

2.13.3. Schema Mapping; the General (Default) Case

In the general setting, a ***C_Function***

i. may be modeling features of more than one type,

ii. may have its spatial domain consist of points on geometries of more than one type

iii. may have coordinates in its value tuples of types that do not support arithmetic (e.g., of type string.)

Under these circumstances, it can difficult to extend the ***C_Function***, f, to the convex hull of its spatial domain.

Therefore, the default schema mapping presents itself:

f(p) = a tuple matching the schema of F (or rows of interest of F) if p belongs to the geometry of a single feature, F, and

f(p) = "undefined" in each coordinate of its value tuple if p belongs to none of the geometries of the feature collection.

The above definition leaves open how to define f(p) if p belongs to the geometries of two or more features. It is easiest to demand that the feature collection satisfies the Partition Condition:

The Partition Condition: We say a collection of features satisfies the Partition Condition if the geometries of the features in the collection are pairwise disjoint.

The satisfaction of the partition condition ensures that the ***C_Function***, f, is completely defined by the two rules of section 1.13.3.2, because it precludes the possibility of overlapping geometries.

If the Partition Condition is not met, then for some p in the coordinate space of the feature geometry, p belongs to the geometries of two or more features, and some additional rule must decide the value of *R*(x). We will see some examples of such rules below.

2.14. Rules of Type 1, 2, and 3

2.14.1. Creating Coverages from Feature Collections

In this section, we suppose that we are given a collection of features of interest. We further assume that the features are of a common type, and that all are features with geometry. The task is to create a ***C_Function***, f, that models (to the extent possible) the information carried by the feature collection.

Let S denote the coordinate space containing the geometries of the features of interest. It is often desired to create a ***C_Function*** that is defined on all of S, or at least on the convex hull of the geometries of the features of interest.

Let p be an element of S. There are three types of rules that determine the value, f(p), that f takes at p:

i. Rules of Type One treat the case when p belongs to the geometry of exactly one feature of interest. Rules of Type One define the fundamental semantics of the coverage.

iv. Rules of Type Two treat the case when p belongs to the geometries of two or more features. (Of course, rules of Type Two are not needed if the coverage satisfies the Partition Condition.)

v. Rules of Type Three treat the case when p is a point in no feature geometry.

There are a large number of rules of each type in common use in GIS technology. Because the three types of rules are independent of each other, rules of the three types can be "mixed and matched" to create a very large number of complete rules for the ***C_Function*** f. The following examples are meant to convey the generality of coverage evaluation processes. The examples are not meant to constrain the types of ***C_Function***s that can be conceived or implemented.

The names "rule of type one", rule of type two" and "rule of type three" are not mnemonic. Implementers are encouraged to find names that carry greater semantic value.

2.14.2. Example Rules of Type One

Rules of Type One are usually the easiest, but they require one to be specific about the semantics of the situation. Each of the following rules shows how the schema of a feature collection is exploited to create a schema for a coverage. This knowledge exposes a great deal of the semantics of the coverage. Here are some examples.

Rule 1.1 [Subset of Schema Rule] We assume here that we are given a discrete collection of features whose geometries are distinct single points. Each feature has a schema providing values for spectral radiance, temperature, and humidity at that point (using an SRS to find the intended location on earth.) A coverage can be derived from this feature collection , using the set of points obtained from the geometries of its members as a spatial domain. Here is an example rule of Type One: For a point p in the spatial domain (that is, the point p is contained in the finite collection of points that form the geometries of the features in the collection) define f(p) = spectral radiance at the point p. Note that the coverage schema is "inherited" from a subset of the schema of the feature collection. Also note that the spectral radiance may be a single integer between 0 and 255, a color vector, a hyperspectral construct, or any of a large number of other information types, and that the type employed must be exposed.

Rule 1.2 [Function on Schema Values Rule] We assume again that we are given a discrete collection of features whose geometries are distinct single points. Again, we assume several properties, $P_1, ..., P_n$, (perhaps including geometry) are contemplated. Again, given a point p in the spatial domain (which we assume is the union of the point geometries of the feature collection), we need to articulate the rule by which f(p) is calculated. By the assumptions for Rules of Type One, p is the geometry of exactly one of these features, say F. Let $V_1, ..., V_n$ be the value vectors of properties $P_1, ..., P_n$, for the feature F. Define f(p) to be the value of a previously specified function of $V_1, ..., V_n$. (The function might compute the average of the value vectors, for example).

An example of a coverage evaluation processes of type 1.2 might be as follows. Suppose the features of interest are disjoint parcel features, and let each parcel have the following schema: a PropertyName "Assessment" taking value "Assessed Value," a PropertyName "geometry" taking value "polygon", a Property Name "perimeter" taking value "perimeter length," a PropertyName "BoundaryStreets" taking a list of string values for each street bounding part of the parcel, and a PropertyName "Area" taking value "acres." One may construct a ***PointC_Function***, f, for a coverage as follows:

F(p) = "true" if {p is a point in a parcel sharing a boundary with Main Street} and {the Assessed Value of that parcel is greater than or equal to $10,000}

F(p) = "false" otherwise

The coverage so constructed might be named "*Valuable Properties along Main Street*."

Another coverage built on the same feature collection could be:

F(p) = (perimeter length (acres) * (assessment) where perimeter length, acres, and assessment are values of the parcel containing p.

A third coverage built on the same feature collection could be:

F(p) = "true" if area(polygon) = (acres) [where "area" is the function mapping a geometry to its area]

F(p) = "false" otherwise.

The coverage so constructed might be named "*AreaIntegrityCheck*"

Rule 1.3 [Project World Rule]. The Project World may be a collection of observations of instrumented values (such as temperatures) at various disjoint locations. Let the set of these point positions be a spatial domain. Define f(x) = the instrument value at *x*.

Rule 1.4 [Full Inheritance Rule]. Start with any feature collection where the geometries of the features are pairwise disjoint. Let the union of the points in the geometries be the spatial domain. Define f(p) = schema of the unique feature whose geometry contains p.

Rule 1.5 [Identity Schema Rule]. Start with any feature collection where the geometries of the features are pairwise disjoint. Let the union of the points in the geometries be the spatial domain. Define f(p) = OID [the object identifier] of the unique feature whose geometry contains p.

It should be clear that there are many flavors of Rules of Type One, and that the list above is by no means exhaustive. The rules may take advantage of the geometry, topology, schemas, property values, or other constructs of a feature.

2.14.3. Rules of Type Two

Rules of Type Two are necessary because feature geometries may overlap. As was the case with Rules of Type One, some cognizance of the nature of the coverage semantics is necessary to state Rules of Type Two. For p in this paragraph, we assume p is in more than one feature's geometry. The object is to specify f(p).

Here are some example Rules of Type Two:

Rule 2.1 [Averaging Rule]. Suppose the features of interest are overlapping digital images, with all of the images sharing a common lattice. Suppose p is a point where several images overlap (and disagree about the spectral radiance at p.) One could define f(p) = average (spectral radiance at p) where the average is taken over all features (images) defining a value at p.

Rule 2.2 [Double Exposure Rule]. Suppose the features of interest are overlapping digital images, with all of the images sharing a common lattice. Suppose p is a point where several images overlap. One could define f(p) = sum (spectral radiance at p) where the sum is taken over all features (images), thus defining a value at p. (This rule may be appropriate for modeling double exposure effects, and may have more use in radar image coverages than photographic images because of "building layover").

Rule 2.3 [Sweepstakes Rule]. Suppose there are many features, F_1, F_2, … ,F_n, whose geometries contain p. Sort these features according to some rule, and let feature F be the first element in the sorted list. Now, define f(p) using the PropertyValues of F. For example, if F has properties P_1, ..., P_n that take values V_1, ..., V_n. We could define F(p) to be a previously defined function of V_1, ..., V_n.

Rule 2.3.1 [Occlusion Rule]. The sort in Rule 2.3 may be on nearness to a perspective center (close objects mask farther objects in photography).

Rule 2.3.2 [False Perspective Rule]. The sort in Rule 2.3 may use a z-buffer to create a "view" of an otherwise hidden surface

Rule 2.3.3 [Priority Rules]. The sort in Rule 2.3 may be based on some other priority, such as freshness of date-time stamp (newer features mask older ones.)

Obviously, there can be many other Rules of Type Two.

2.14.4. Rules of Type Three

Rules of Type One and Type Two define f(p) for every p in the union of the geometries of the features of interest. It is sometimes important to establish values of f(p) when p is an element of the coordinate space of those geometries, but not actually a member of any feature's geometry. Rules of Type Three (which include interpolation rules as a special case) accomplish this. As seen in the examples below, Rules of Type Three usually extend Rules of Type One and/or Type Two, and thus do not require a semantic knowledge of the features. In the remainder of this paragraph, let p be a point in the coordinate space containing the geometries of the features of interest, but not a member of any feature's geometry.

Rule 3.1 [No Extension Rule] For every such p, define f(p) = "not defined"

Rule 3.2 [Nearest Neighbor Rule] Let q be the nearest point in the union of the feature geometries to p. Define f(p) = f(q) where f(q) is determined by a Rule of Type One or a Rule of Type Two

Rule 3.3 [Standard TIN Rule] Perform a Delaunay Triangulation on the points in F, and pick the triangle in which x is contained. The vector p is a linear combination of the vectors at the corners of the triangle. Define f(p) to be the same linear combination of the values of f on the three corners. (This is barycentric interpolation.) If there is no triangle containing x, choose one of the following:

Rule 3.3.1 [Unextended TIN Rule] Define f(p) = "not defined"

Rule 3.3.2 [Triangle Extension Rule] Define f(p) using a linear extension (regression) of the nearest triangle. (This is barycentric extrapolation.)

Rule 3.3.3 [Nearest Neighbor Extrapolation Rule] Let q be the nearest point to p that is given a value using a previous rule, and define $R(\mathrm{x}) = R(\mathrm{y})$

Rule 3.4 [Bilinear Grid Interpolation Rule] If the points in the union of the features' geometries forms a "Grid" pattern, and p_1, ..., p_4 are four the four corners of a "Grid square" containing p, compute f(p) using bilinear interpolation in that cell.

Rule 3.5 [Bicubic Grid Interpolation Rule] If the geometries of the features form a "Grid" pattern, compute $R(\mathrm{x})$ using the nine surrounding points and bi-cubic convolution

Rule 3.6 [Regression Rule] Choose a circle centered at p that contains four or more points in the union of the feature's geometries. Perform a regression analysis of f on those points to determine a value for f(p).

Rule 3.7 [Trilinear Interpolation Rule] If the union of the geometries of the features forms a three dimensional Grid pattern, and p_1,...,p_8 are the surrounding eight points enclosing p, define f(p) to be the tri-linear interpolation of $f(p_1)$, ..., $f(p_8)$.

Clearly Rule 3.7 and other rules can be extended to any dimension, and there are many more potential Rules of Type Three. The examples given here are not meant to constrain the implementation of coverages.

2.15. A Discussion of Soft Boundaries

2.15.1. Soft Boundaries

Geospatial information is often conveyed using features with soft boundaries. For example, it is difficult to delineate exactly the spatial extent of the following features:

The Rocky Mountains	The Piedmont
The North Channel entrance to Manila Bay	A coniferous forest
An urbanized region	A region with sandy loam soil
The portion of the San Diego Harbor with bottom type "mud"	A wetland

Table 2-1. Some Features with Soft Boundaries

We insist that the Project World provide crisp boundaries to features such as these, whether they are to be modeled using features with Geometry, or within a coverage. The crisp boundary in the Project World maps to a crisp boundary modeled in the feature's Geometry or in its Coverage. In either case, it is understood that some or all of the crisp boundary is approximate or artificial. The degree of knowledge about the true boundary is conveyed in metadata at the feature collection level, or in PropertyValues carried with the feature's schema.

2.16. Families of Coverages

2.16.1. Definition of a Family of Coverages

We say that a set of coverages, $\{C_1, ..., C_n\}$ is a *family* of coverages if all the coverages in the list use the same space of coordinates for their geometries, and use the same SRS to relate those coordinates to earth locations. For example, three coverages might center on the modeling of demographic regions, lines of transportation, and retail business locations. If the coordinates of the Cartesian coordinates of the geometries in all three coverages are related by a single SRS to a common state plane, we say that all three coverages belong to the same family.

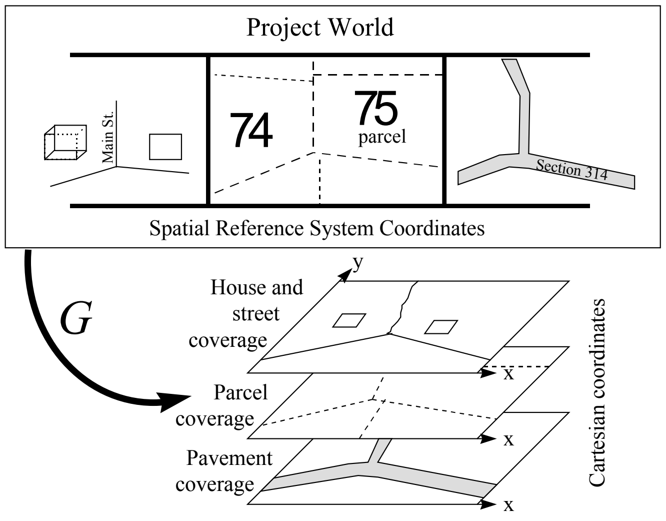


Figure 2-5. A Single Project World May Spawn a Family of Coverages

The reason families of coverages are useful, is that they can be mentally stacked as in Figure 2-5, and are ready to support the calculus of coverages. That is, families of coverages are automatically geometrically registered in the sense of the next section.

2.17. The Geometric Registration of Coverages

2.17.1. Definition of Registration

We often need a concept that is stronger that that of a family of coverages. Sometimes we are confronted with two coverages that we would like to "stack" in the sense of a family of coverages, but where such stacking is impossible. Suppose, for example, the coverages are two different photographs, taken from different angles. Each photo is a central projection from the three-dimensional earth to a two-dimensional photograph, and the two projections are different.

Here, even though stacking is impossible, it is possible to "geometrically register" the two coverages.

We say Coverage C_1 is geometrically registered to coverage C_2 if the following condition is satisfied:

> if x is a corner in C_1 and y is the corresponding corner in C_2 (that is, x and y refer to the same real world location), and if X is the set of all locations that SRS1 relates to x, and Y is the set of all real world locations that SRS2 relates to y, then there must be a location z such that: $z \in X \cap Y$

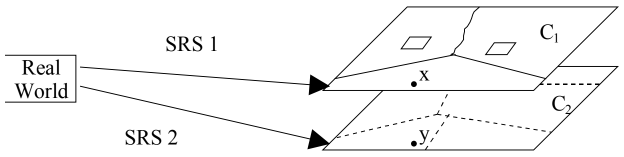


Figure 2-6. The Geometric Registration of Coverages

If two coverages are modeling overlapping portions of the earth, they can sometimes be brought into geometric registration by a change-of-coordinate mapping on one or both of the coverages' coordinate spaces.

2.17.2. The Photogrammetric Registration of Coverages and Images.

Most earth images and some coverages that are designed to support perspective displays use a Spatial Reference System that is based on a central projection (sometimes called a projective transformation) that maps 3-dimensional coordinates to 2-dimensional coordinates. That is, the Spatial Reference System maps real world (that is, geodetic) (x,y,z) coordinates to the Cartesian (r,c) coordinates of an image plane. Using the theory of projective transformations, one may find constants $a_1, \ldots, a_8$ so that Equation 1 holds.

Moreover, suppose there are two such coverages whose footprints overlap. Let their Spatial Reference Systems be G_1 and G_2. Now, one can construct 16 constants: $a_1, \ldots, a_8$ and $b_1, \ldots, b_8$ so that

i. $a_1, \ldots, a_8$ generate coordinates (r_1,c_1) on the first coverage using Equation 1

ii. $b_1, \ldots, b_8$ generate coordinates (r_2,c_2) on the second coverage using the same Equation, replacing a's with b's

iii. if (x,y,z) is any point imaged by both photographs (i.e., is in both coverages), and if $(r_1,c_1) = G_1(x, y, z)$ and $(r_2,c_2) = G_2(x, y, z)$, then $r_1 = r_2$ (that is, there is no y-parallax), and the x-parallax, $c_1 - c_2$, is directly related to the value z.

[Note that the planes containing both perspective centers also contain lines in both image spaces defined by the equations r_1 = constant and r_2 = constant, respectively. These planes are the equipolar planes.]

When 16 constants have been so constructed, we say the coverages have been brought into photogrammetric registration.

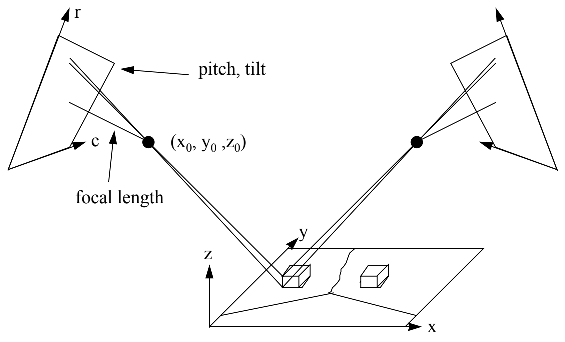


Figure 2-7. The Situation for Photogrammetric Registration

2.18. The Calculus of Coverages

2.18.1. Introduction to Calculus of Coverages

The power of coverages is their ability to model and make visible spatial relationships between, and the spatial distribution of, earth phenomena.

There are, fundamentally, two types of operations on coverages that are used to exploit this power: unary operations, and binary operations. A unary operation on a coverage modifies the coverage to create a new coverage. The modification can be morphological (i.e., the Spatial Reference System is modified), or phenomenological (i.e., the ***C_Function*** is modified), or both. A binary operation on a pair of geometrically or photogrammetrically registered coverages creates a new coverage by logical, arithmetic, or rule-based calculations performed on the coverages' PropertyValues.

2.18.2. Unary Operations on Coverages

A coverage may be modified (thereby creating a new coverage) by the combination of the following unary operations:

i. Compose the Spatial Reference System with a change of coordinates on the Cartesian space to create a new Spatial Reference System. (This may be done to bring the coverage into geometric registration with another coverage.)

ii. Introduce new features whose shapes form a geometric buffer around features with selected PropertyValues.

iii. Extend the domain of the ***C_Function*** by using a Rule of Type 3.

iv. Form a sub-coverage by deleting selected features or deleting selected rows from the schema.

v. Replace the range of the ***C_Function*** with a different PropertyValue from the feature schemas.

vi. Add a coordinate to the tuple range of R and fill it with a logical and/or arithmetic expression of several PropertyValues.

vii. Replace the domain of the ***C_Function*** with a larger (or smaller) set.

viii. Add (or remove) topology.

ix. Change the feature schemas.

As an example of a unary operation, suppose a coverage models the components of a building. We might define f(p) = 1 if x belongs to a feature geometry whose feature schema reveals it to be a subtype of the plumbing class of features, and f(p) = 0 otherwise. If we think of "1" and "0" as "visible" and "invisible", the result is a kind of x-ray "view" of the building, where only the plumbing is visible.

The idea here is to include all of the unary operations in Dana Tomlin's Map Algebra [8]and more.

2.18.3. Binary Operations on Coverages

Given two coverages C1 and C2 that are in the same family of coverages, we can construct a new coverage C3 by any of the following operations:

i. Define C_3 to be the complete geometric cross product of C_1 and C_2. That is, the geometries of C_3 are precisely the geometries that can be formed by intersecting a geometry of C_1 with one of C_2, and the schema on such a shape is the concatenation of the corresponding two schemas.

ii. Define C_3 to be a partial geometric cross product of C_1 and C_2. That is, the geometries of C_3 are precisely the geometries that can be formed by intersecting a geometry of C_1 with one of C_2, where the schemas of the two shapes satisfy a given assertion. The schema on the resulting geometry is a predefined function of the corresponding two schemas. (Example: Given a vegetation coverage and a transportation coverage, build a new coverage of all the roads in or adjacent to forests, and assign their color by the rule: <orange> for two lane roads; <red> for greater than two lanes.)

The idea here is to include all of the binary operations in Dana Tomlin's Map Algebra [8] and more.

Given two coverages C_1 and C_2 that are photogrammetrically registered, we can construct a new coverage C_3 by the following operation:

i. Create a stereoscopic view of the overlap in the two coverages. Create a Project World by forming a crisp "mind's eye" view of the features and their schemas in the stereoscopic model. Capture these features in an OpenGIS feature collection, that is, in a new coverage.

2.19. Interfaces on Coverages

2.19.1. Evaluation

The primary behavior of Coverages is that of *evaluation*.

Evaluation is the generic name for the methods by which a coverage is queried, so that the information it models may be exposed and shared. Typically, an evaluation method accepts an element of the spatial domain of a ***C_Function***, and (computes, if necessary, and) exposes the corresponding value vector.

The evaluation method on a coverage depends on the type of coverage at hand. This abstract specification discusses a few types, but is certainly not exhaustive.

Evaluation may be implemented as a collection of stand-alone services, as interfaces on Coverages, or in any other way deemed best by specifiers at the implementation level.

2.19.2. Inverse Evaluation and Shapes

Given a value vector, **v**, from a Coverage's range, Inverse Evaluation is the calculation and exposure of the set { P | P is in the Coverage's Spatial Domain, and the Value of the ***C_Function*** at P is **v**}. The result of this calculation is sometimes called a "shape." Examples of applications of the Inverse Evaluation interface include the extraction of contours from an elevation coverage, the extraction of classified regions in an image using pixel clusters, and the exposure of a land parcel shape given its identification number.

2.19.3. Service Capabilities

A coverage must provide the following services

i. Output the value of one or more specified properties, at one or more specified locations within the coverage's spatial domain.

ii. Output the schema for the geospatial data captured by the coverage, including the types and ranges of values generated by the evaluation interface.

iii. Output data defining the limits of the coverage spatial domain. In two dimensions, the extent might be represented as a rectangle (e.g., minimum bounding rectangle), an area bounded by a polygon, and/or an area bounded by an ellipse. In three dimensions, the extent might be represented as a parallelepiped and/or polyhedral solid. The coverage extent data needed includes the number of coordinates in that extent (e.g., 1, 2, 3, or 4 dimensions), the identities of these coordinates, and the reference system of that extent.

iv. Output property quality estimates. Property quality estimates might be in the form of the Property Accuracy Metadata Components currently specified in Section 3.6 of Topic 9: Quality. The property quality estimates output must include estimates for any errors committed by the implementation of this coverage.

v. Output the recorded relationships of this coverage to other coverages.

vi. Output other metadata of the coverage..

vii. Output one or more locations, at which the coverage defines a specified value for each of a set of the properties defined by that coverage.

viii. Output the type of the coverage, possibly including the contents and types of coverage data recorded, interpolation and extrapolation methods, schema mapping rules, etc.

Note: The following paragraph is not a requirement for RFP 5.

Services that create new coverages, from one or more existing coverages, are also often needed to meet higher level application needs. Desirable coverage generation services include creating a new coverage that defines the values of a:

i. Modified property that is a specified function of the existing value of one property

ii. New property that is a specified function of the existing values of two or more selected properties

iii. Modified property that is a specified function of the values of one existing property over a small region centered on each location in the new coverage

iv. Existing property over a coverage extent that is a subspace of the existing coverage extent

v. Existing property over a coverage extent that is a union of the existing extents of two or more coverages

vi. Existing property where locations in the new coverage use a different reference system

vii. Existing property where the new coverage is of a different coverage type

viii. Modified values of a selected existing property, at selected locations

ix. A combination of the above

2.20. References for Section 2

[1] Cook, Steve, and John Daniels, Designing Objects Systems: Object-Oriented Modelling with Syntropy, Prentice Hall, London, 1994, xx + 389 pp.

[2] Rosenfeld, A., and A. C. Kak, Digital Picture Processing, Second Ed., Vol. 1, Academic Press, 1982.

[3] Rumbaugh, James, Michael Blaha, William Premeerlani, Frederick Eddy, William Lorensen, Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, New Jersey, 1991, xii + 500 pp.

[4] Robert Reeves, Editor, Manual of Remote Sensing, American Society of Photogrammetry and Remote Sensing

[5] Laurini, R., and D. Thompson, Fundamentals of Spatial Information Systems, Academic Press, 1992.

[6] Hilton, P. J., and S. Wylie, Homology Theory, an Introduction to Algebraic Topology, Cambridge University Press, 1960.

[7] Earth Imaging Working Group of the OpenGIS Consortium. 1996. A Request for Information: OpenGIS Imagery. Wayland, Massachusetts.

[8] Tomlin, Dana, Map Algebra

[9] OpenGIS™ Abstract Specification, OpenGIS™ Project Documents 99-100 through 99-116, available through www as <http://www.opengis.org/techno/specs.htm>.

[10] Winter, Stephan, and Frank, Andrew U., Topology in Raster and Vector Representation, GeoInformatica, to appear in issue 1, 2000

3. Abstract Model for The Coverage Type and its Subtypes

3.1. Class name: Coverage

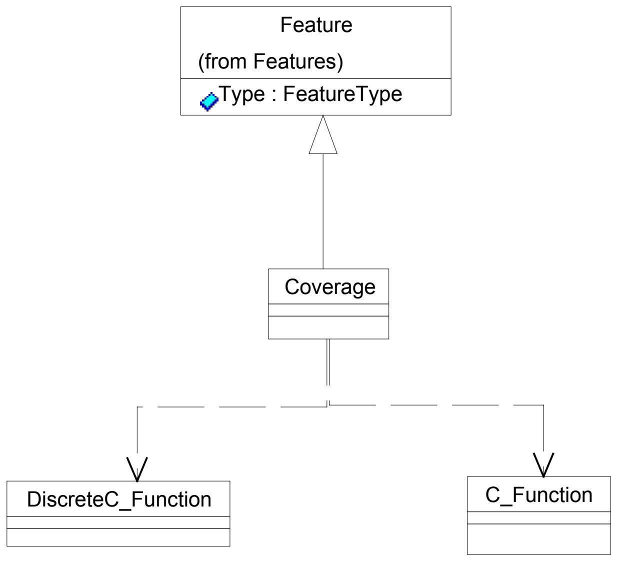


Figure 3-1. Context Diagram for *Coverage* Class.

3.1.1. Category: Coverages

3.1.2. Documentation:

A coverage is a subtype of a feature. A coverage has one or more pairs of distinguished properties that define functions, called Coverage Functions. Each Coverage Function is characterized by its domain, and by the assignment of values to elements of the domain. Two kinds of Coverage Function are allowed: the DiscreteC_Function and the C_Function.

3.1.3. Hierarchy:

Superclasses:	Feature

3.1.4. Public Use:

C_Function

DiscreteC_Function

3.2. Class name: C_Function

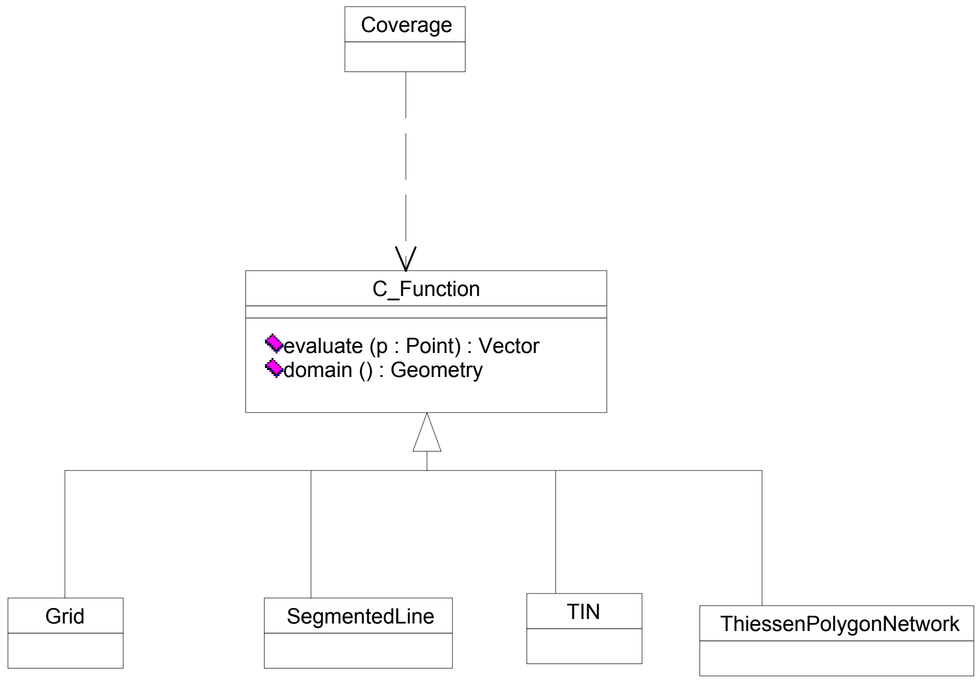


Figure 3-2. Context Diagram for *C_Function* Class

3.2.1. Category: Coverages

3.2.2. Documentation:

The domain of a C_Function consists of a (usually infinite) collection of points in a coordinate space. The domain often consists of the points in the union or convex hull of a finite collection of geometries. A C_Function maps points to vector values thus creating a vector-valued coverage. Four special types of C_Function are shown in Figure 3.2. Only one of them, Grid, is in the scope of RFP 5.

3.2.3. Hierarchy:

Superclasses:	none

3.2.4. Associations:

<no rolename> : ***DiscretePointC_Function*** in association: Equivalence

3.2.5. Public Interface:

Operations:	evaluate
	domain

3.2.6. Operation name: evaluate

Public member of:	***C_Function***
Return Class:	Vector
Arguments:	Point p
Documentation:	The evaluate method returns the C_Function value for the passed point. Evaluate is often delegated to an Evaluator Service of the appropriate Interpolation type

3.2.7. Operation name: domain

Public member of:	***C_Function***
Return Class:	Geometry
Documentation:	The domain operation returns the geometry elements constituting the domain of the coverage function.

3.3. Class name: DiscreteC_Function

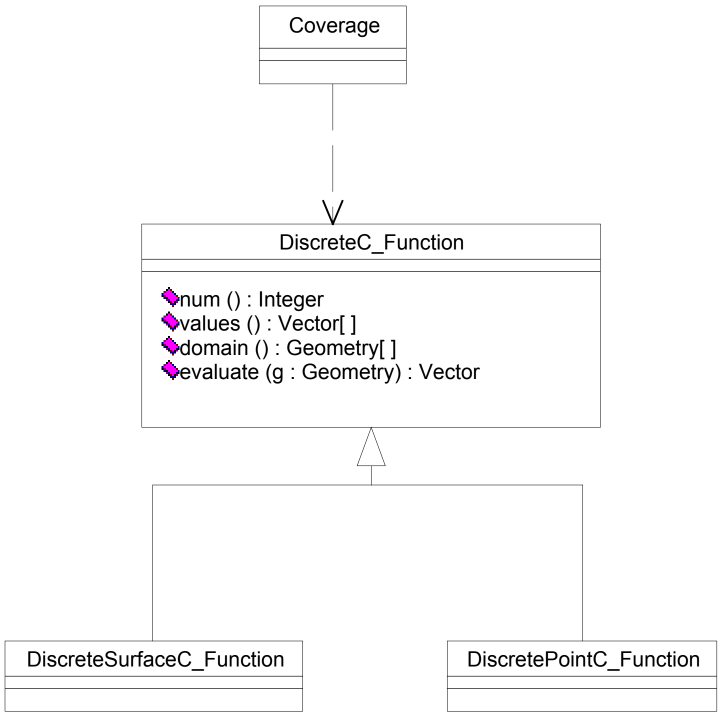


Figure 3-3. Context Diagram for *DiscreteC_Function* Class

3.3.1. Category: Coverages

3.3.2. Documentation:

The domain of a DiscreteC_Function consists of a finite collection of geometries. The DiscreteC_Function maps each geometry to a value. For now, the values may be assumed to be homogeneous vectors (a vector is a tuple whose coordinates are numeric.) A DiscreteC_Function is thus a discrete or step function as opposed to the (normally) continuous nature of a C_Function. Discrete functions are ones that can be explicitly enumerated in a datatype as (input, output) pairs.

Note that when the spatial domain is a finite collection of point geometries, both a DiscreteC_Function and a C_Function can be defined, and both result in the same object.

3.3.3. Hierarchy:

Superclasses:	none

3.3.4. Public Interface:

Operations:	num
	values
	domain
	evaluate

3.3.5. Operation name: num

Public member of:	DiscreteC_Function
Return Class:	Integer
Documentation:	The num method returns the cardinality of the domain.

3.3.6. Operation name: values

Public member of:	DiscreteC_Function
Return Class:	Vector[ ]
Documentation:	The *values* method returns the finite collection of vectors that are the values returned by the DiscreteC_Function. This is the image of the function.

3.3.7. Operation name: domain

Public member of:	DiscreteC_Function
Return Class:	Geometry[ ]
Documentation:	The domain method returns the finite geometry collection whose elements are the domain of the function.

3.3.8. Operation name: evaluate

Public member of:	DiscreteC_Function
Return Class:	Vector
Arguments:	Geometry
Documentation:	For each geometry of the domain, the evaluate method returns the appropriate value vector. If the geometry passed is not in the domain then an error has occurred.

3.4. Class name: DiscretePointC_Function

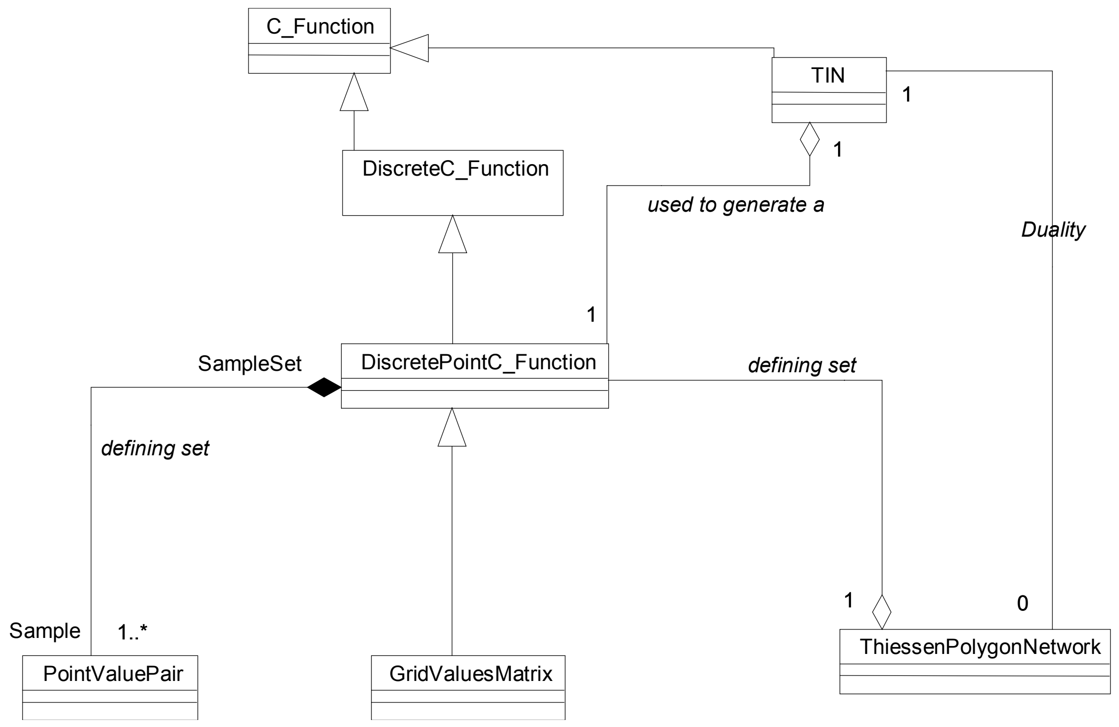


Figure 3-4. Context Diagram for *DiscretePointC_Function* Class.

3.4.1. Category: Coverages

3.4.2. Documentation:

A DiscretePointC_Function is a logical subtype of both DiscreteC_Function and a C_Function. It is a Coverage Function characterized by a finite domain consisting of points, so that the Coverage Function can be represented by a list of PointValuePairs.

3.4.3. Hierarchy:

Superclasses:	DiscreteC_Function

3.4.4. Associations:

Sample : ***PointValuePair*** in association: defining set

<no rolename> : ***TIN*** in association: used to generate a

<no rolename> : ***ThiessenPolygonNetwork*** in association: defining set

<no rolename> : ***C_Function*** in association: Equivalence

3.5. Class name: DiscreteSurfaceC_Function

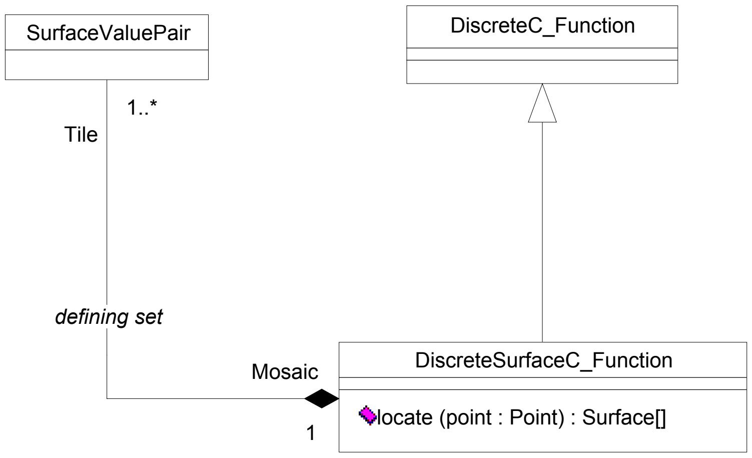


Figure 3.5. Context Diagram for *DiscreteSurfaceC_Function* Class.

3.5.1. Category: Coverages

3.5.2. Documentation:

A DiscreteSurfaceC_Function is a Coverage Function whose domain is a finite set of surfaces and whose range is a homogeneous set of vector values. The Coverage Function is characterized by a finite set of SurfaceValuePairs.

3.5.3. Hierarchy:

Superclasses:	DiscreteC_Function

3.5.4. Associations:

Tile:	***SurfaceValuePair*** in association: defining set

3.5.5. Public Interface:

Operations:	locate

3.5.6. Operation name: locate

Public member of:	DiscreteSurfaceCoverage
Return Class:	Surface[ ]
Arguments:	Point point
Documentation:	The locate method returns the list of surfaces in the domain which contain the point passed.

3.6. Class name: GeometryValuePair

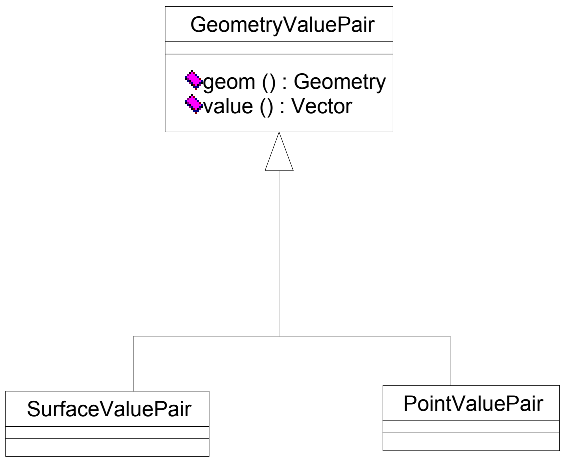


Figure 3-5. Context Diagram for *GeometryValuePair* Class.

3.6.1. Category: Coverages

3.6.2. Documentation:

A GeometryValuePair is an abstract class that is used as an element of a list that defines the relationships of a discrete function by listing pairs of independent and dependent variables. Each independent variable is a geometry and each dependent variable is a vector. Thus each member of this class consists of two values: one independent (a geometry) and one dependent (a vector). GeometryValuePairs are used to implement DiscreteC_Functions.

3.6.3. Hierarchy:

Superclasses:	none

3.6.4. Public Interface:

Operations:	geom
	value

3.6.5. Operation name: geom

Public member of:	***GeometryValuePair***
Return Class:	Geometry
Documentation:	The geom method returns the independent variable of the pair.

3.6.6. Operation name: value

Public member of:	***GeometryValuePair***
Return Class:	Vector
Documentation:	The value method returns the dependent variable of the pair.

3.7. Class name: PointValuePair

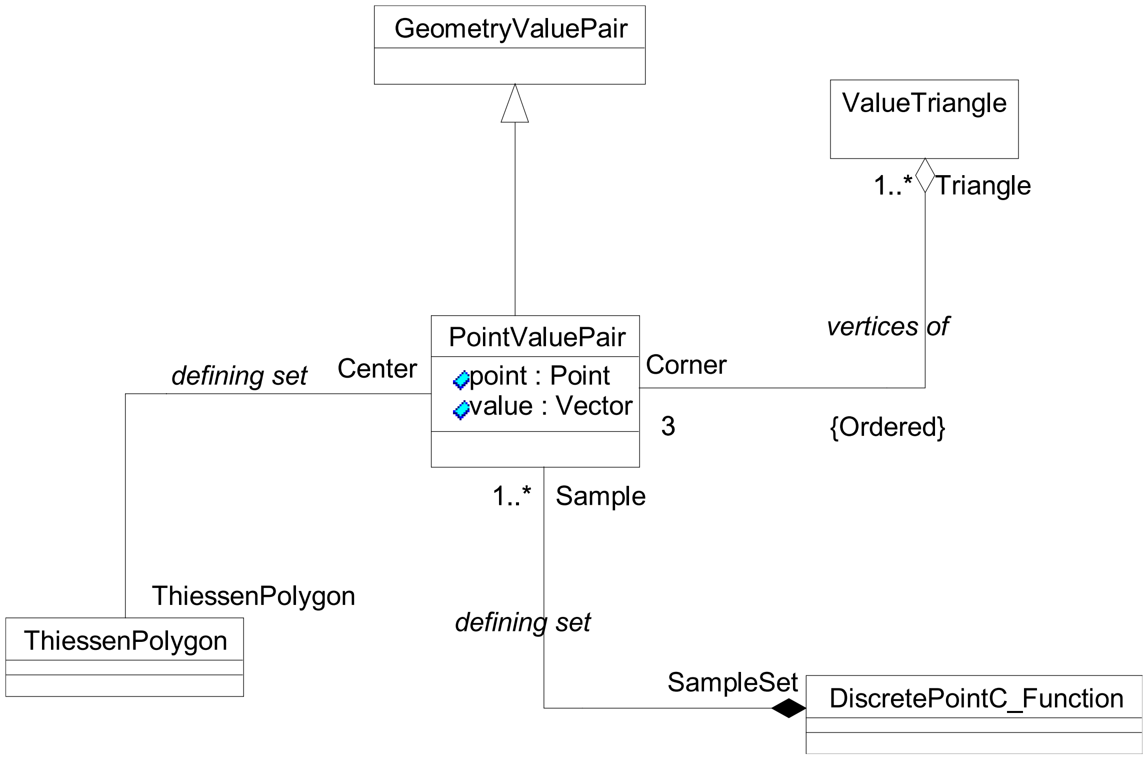


Figure 3-6. Context Diagram for *PointValuePair* Class.

3.7.1. Category: Coverages

3.7.2. Documentation:

A PointValuePair is an ordered pair where the first element is a point in the spatial domain of a Coverage Function and the second element is a value that expresses the value of the Coverage Function at that point.

3.7.3. Hierarchy:

Superclasses:	GeometryValuePair

3.7.4. Associations:

SampleSet : ***DiscretePointC_Function*** in association: defining set

Triangle : ***ValueTriangle*** in association: vertices of

ThiessenPolygon : ***ThiessenPolygon*** in association: defining set

3.7.5. Public Interface:

3.7.6. Attributes:

point :	Point
Documentation:	The point (0-dimensional geometry) associated with the PointValuePair
value :	***Vector***
Documentation:	The value vector associated with the PointValuePair.

3.8. Class name: SurfaceValuePair

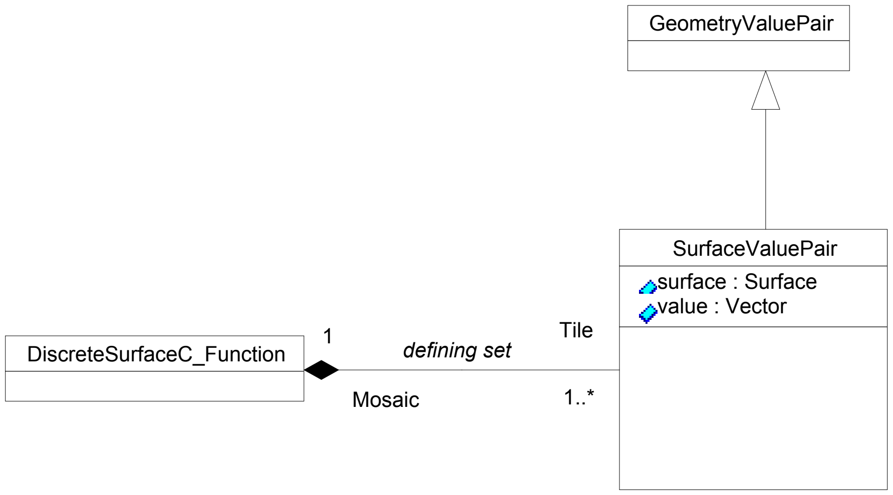


Figure 3.8. Context Diagram for *SurfaceValuePair* Class.

3.8.1. Category: Coverages

3.8.2. Documentation:

A SurfaceValuePair associates a surface with a value. SurfaceValuePairs are used in the definition of discrete (step function) surface coverage functions.

3.8.3. Hierarchy:

Superclasses:	GeometryValuePair

3.8.4. Associations:

Mosaic : ***DiscreteSurfaceCoverage*** in association: defining set

3.8.5. Public Interface:

3.8.6. Attributes:

surface :	Surface
Documentation:	The attribute surface is a 2-dimensional geometry and represents a set that takes a common value.
value :	Vector
Documentation:	The attribute value is a vector of observations or phenomena values within or on the surface.

3.9. Class name: TriangulatedIrregularNetwork (TIN)

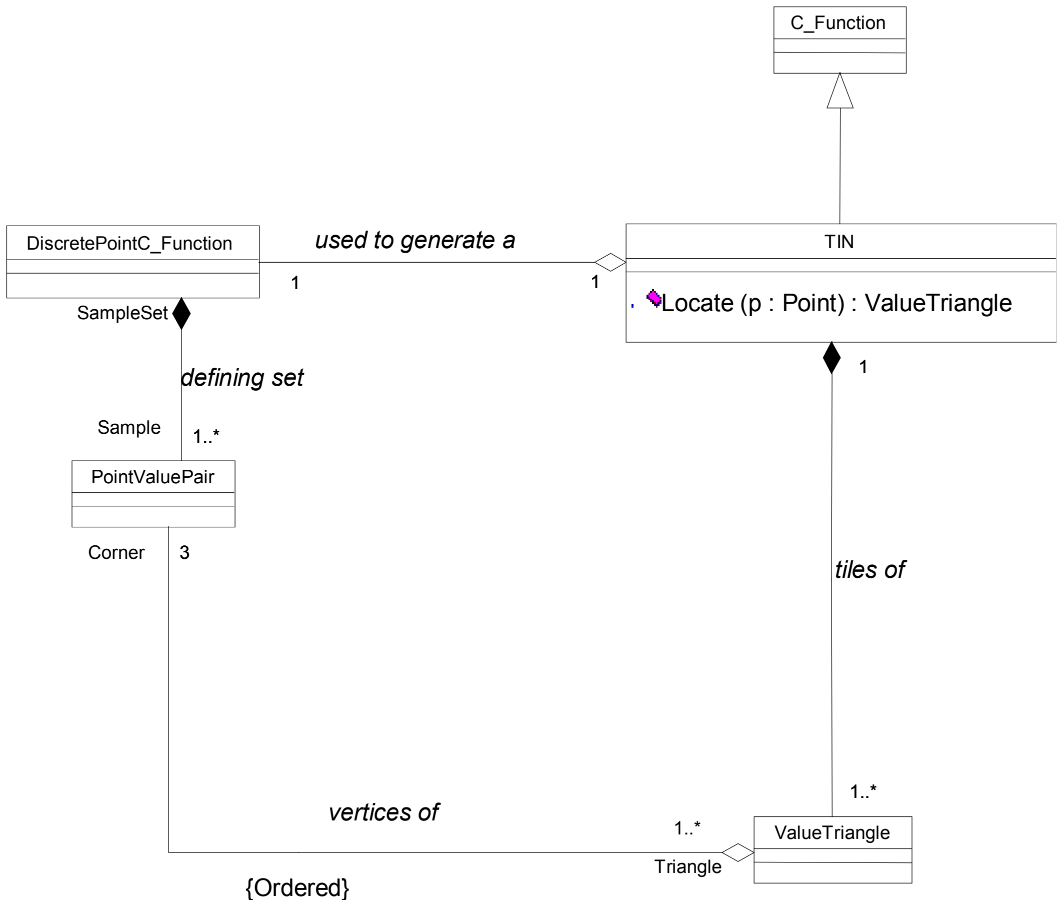


Figure 3-7. Context Diagram for *TrianglulatedIrregularNetwork (TIN)* Class.

3.9.1. Category: Coverages

3.9.2. Documentation:

A Triangulated Irregular Network (TIN) is a C_Function characterized by a tessellation model (usually of the earth's surface) comprised of a network of non-overlapping triangles. The vertices of the triangles form a set of irregular spaced posts. The Delaunay Triangulation method is commonly used to produce TIN tessellations with triangles that are approximately equiangular in shape. The C_Function values are computed by interpolation within each triangle in the tessellation using the values provided at each corner; that is, the C_Function values are produced by a method on ValueTriangles.

3.9.3. Hierarchy:

Superclasses:	***C_Function***

3.9.4. Associations:

<no rolename> : ***DiscretePointC_Functio***n in association used to generate a

<no rolename> : ***ValueTriangle*** in association: tiles of

<no rolename> : ***ThiessenPolygonNetwork*** in association: Duality

3.9.5. Public Interface:

Operations:	locate

3.9.6. Operation name: locate

Public member of:	TIN
Return Class:	***ValueTriangle***
Arguments:	Point p
Documentation:	The locate method returns the triangle in which the passed point is located, together with the values assigned to the corners of that triangle; that is, it returns a ValueTriangle

3.10. Class name: ValueTriangle

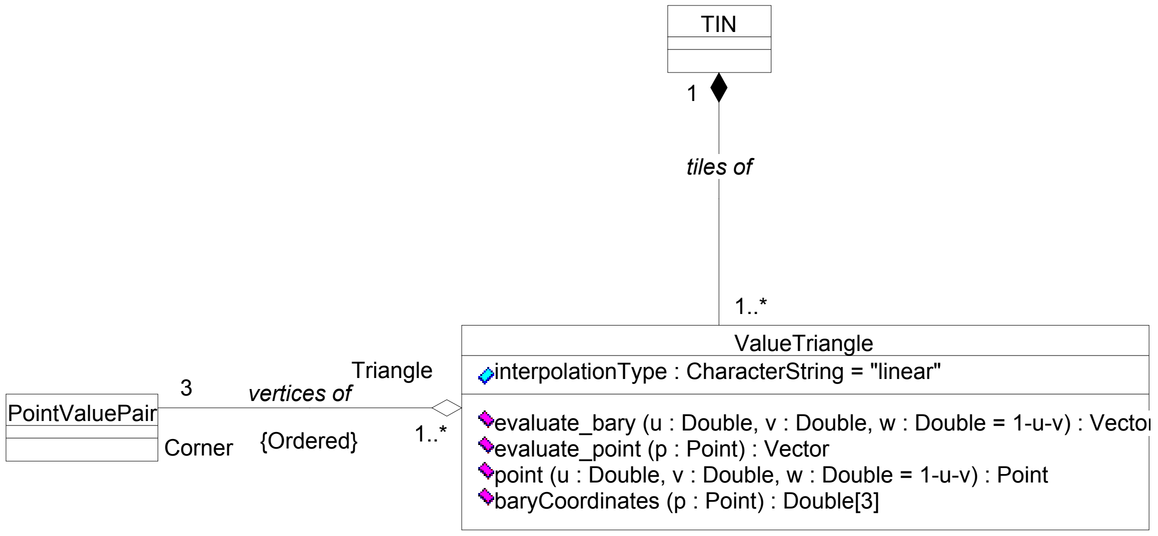

TIN
1
tiles of
1..*
ValueTriangle
interpolationType : CharacterString = "linear"
evaluate_bary (u : Double, v : Double, w : Double = 1-u-v) : Vecto
evaluate_point (p : Point) : Vector
point (u : Double, v : Double, w : Double = 1-u-v) : Point
baryCoordinates (p : Point) : Double[3]
PointValuePair
3
vertices of
Triangle
Corner
{Ordered}
1..*

Figure 3-8. Context Diagram for *ValueTriangle* Class

3.10.1. Category: Coverages

3.10.2. Documentation:

A ValueTriangle consists of three non-collinear points and their associated values used in interpolation of a coverage function. ValueTriangles are usually associated with a TIN or a Triangulated Spline.

Constraint: The three associated points in the ***PointValuePairs*** are not co-linear

3.10.3. Hierarchy:

Superclasses:	none

3.10.4. Associations:

<no rolename> : ***TIN*** in association: tiles of

Corner : ***PointValuePair*** in association: vertices of

3.10.5. Public Interface:

Attributes:	interpolationType
Operations:	evaluate_bary
	evaluate_point
	point
	baryCoordinates

3.10.6. Attributes:

interpolationType :	CharacterString = "linear"
Documentation:	The attribute "linear" indicates that interpolation of values inside a ValueTriangle is to be performed using barycentric averaging of the values at the Triangle's vertices.

3.10.7. Operation name: evaluate_bary

Public member of:	***ValueTriangle***
Return Class:	Vector
Arguments:	Double u
	Double v
	Double w
Documentation:	This operation generates a value at a point inside a ValueTriangle, given the barycentric coordinates of that point. The evaluation is based on an interpolation method (linear interpolation is the default) of the values at the three corners of the triangle.
Qualification:	$0 \leq u, v, w \leq 1$, $u+v+w=1$
Exceptions:	OutOfRange

3.10.8. Operation name: evaluate_point

Public member of:	***ValueTriangle***
Return Class:	Vector
Arguments:	Point p
Documentation:	The operation returns the interpolated value vector at the passed point. Evaluate_point might be delegated to an Evaluator Service of the appropriate Interpolation type.
Exceptions:	OutOfRange

3.10.9. Operation name: point

Public member of:	***ValueTriangle***
Return Class:	Point
Arguments:	Double u
	Double v
	Double w
Documentation:	This operation returns the point in the triangle whose barycentric coordinates are values passed.

3.10.10. Operation name: baryCoordinates

Public member of:	***ValueTriangle***
Return Class:	Double[3]
Arguments:	Point p
Documentation:	This operation exposes the barycentric coordinates of the point passed.

3.11. Class name: ThiessenPolygonNetwork

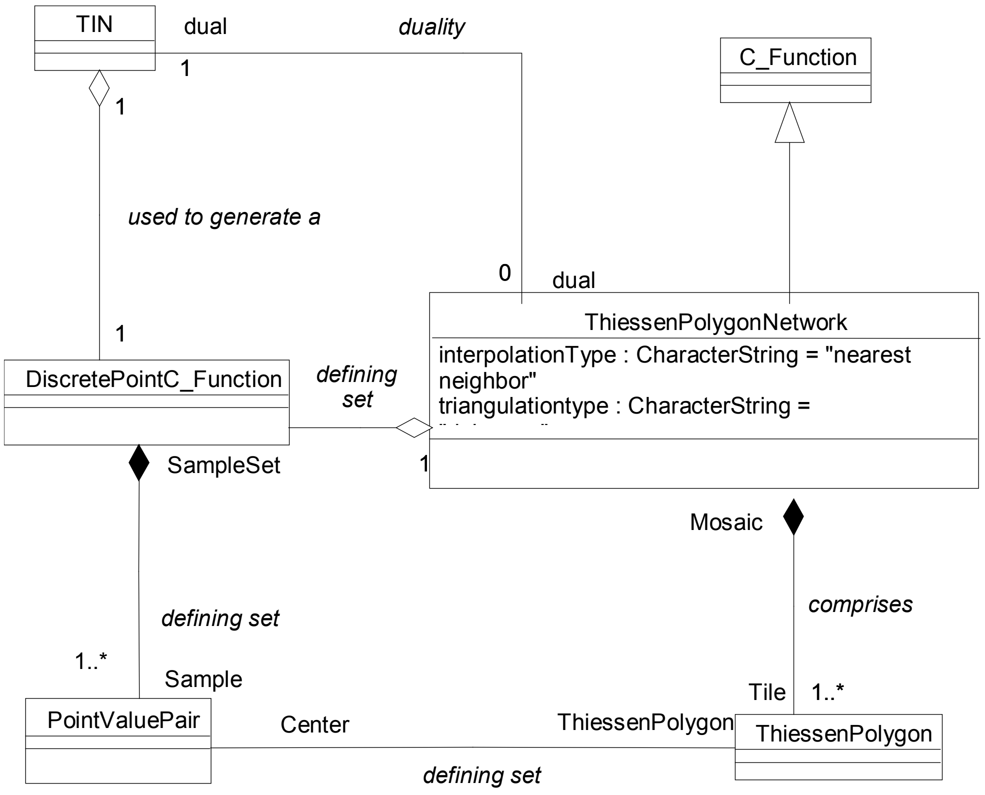


Figure 3-9. Context Diagram for *ThiessenPolygonNetwork* Class

3.11.1. Category: Coverages

3.11.2. Documentation:

A ThiessenPolygonNetwork could also be called a ThiessenPolygonTessellation, or yet more appropriately, a ThiessenPolygonC_Function. This C_Function is characterized by a tessellation of a spatial domain using Thiessen Polygons. The network structure is exposed in the dual TIN of the polygon tessellation. The ***C_Function*** is defined using an interpolation method defined from the ***PointValuePairs*** that are associated with the centers of the polygons. The most common interpolation methods are "nearest neighbor", and "lost area".

3.11.3. Hierarchy:

Superclasses:	***C_Function***

3.11.4. Associations:

<no rolename> : ***DiscretePointC_Function*** in association: defining set

Tile : ***ThiessenPolygon*** in association: comprises

Dual: ***TIN*** in association: Duality

3.11.5. Public Interface:

Attributes:	interpolationType : CharacterString = "lost area"
	triangulationtype : CharacterString = "deluanay"

3.12. Class name: ThiessenPolygon

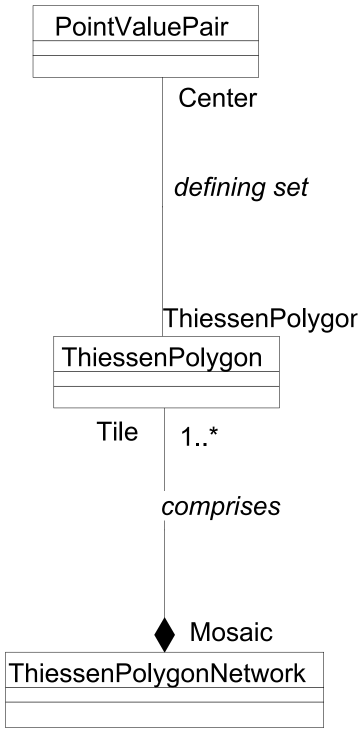


Figure 3-10. Context Diagram for *ThiessenPolygon* Class.

3.12.1. Category: Coverages

3.12.2. Documentation:

A Thiessen Polygon is generated from a defining set by forming the set of points that are closer to a specific point than any other point in the defining set. The specific point is called the Center of the resulting polygon. The boundaries between neighbor polygons are the perpendicular bisectors of the lines between their respective Centers. Drawing the lines between centers whose Polygons are neighbors forms the triangles in the dual TIN. Thiessen Polygons are also referred to as Voronoi Diagrams or Proximal Sets.

3.12.3. Hierarchy:

Superclasses:	none

3.12.4. Associations:

Mosaic:	***ThiessenPolygonNetwork*** in association: comprises
Center:	***PointValuePair*** in association: defining set

3.13. Class name: Grid

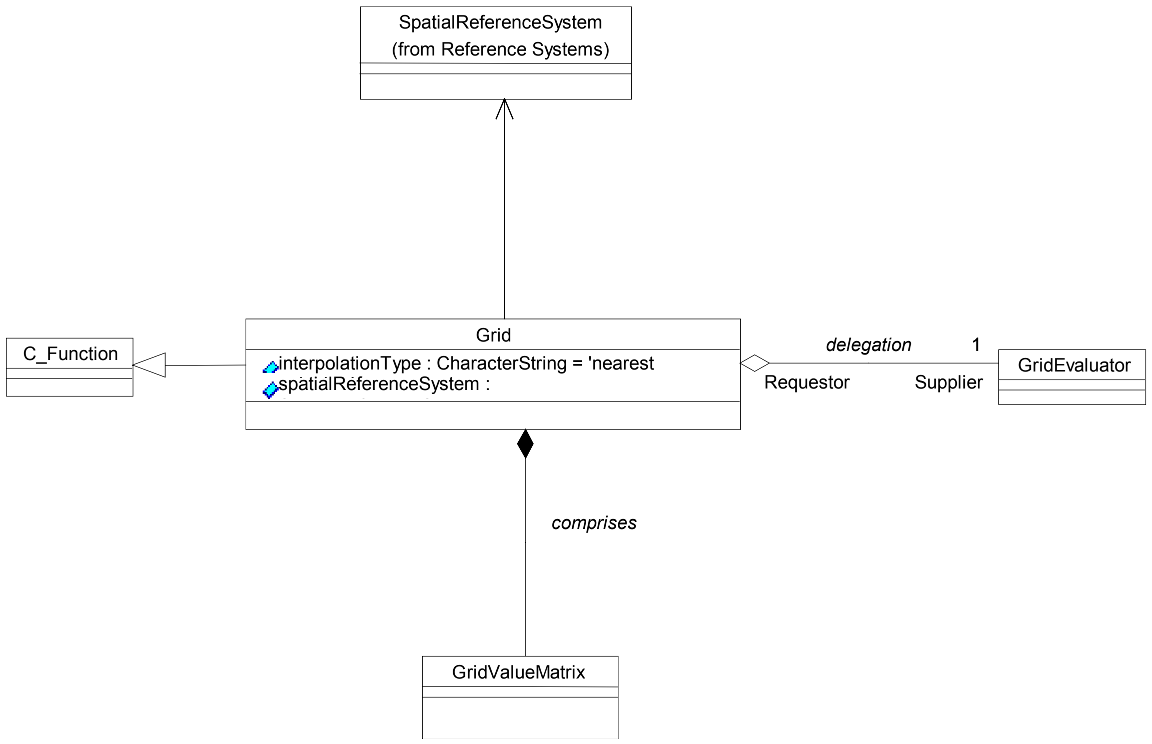


Figure 3-11. Context Diagram for *Grid* Class

3.13.1. Category: Coverages

3.13.2. Documentation:

A Grid is a ***C_Function*** defined from a finite collection of PointValuePairs whose points form an evenly spaced rectangle (using the offset vectors) in the coordinate system of the grid geometry. The domain of the C_Function is often the convex hull of the collection of points. The value of the C_Function at a point is computed with a GridEvaluator that employs an evaluation algorithm.

The Grid is thus constructed from a DiscretePointC_Function that is characterized by the finite collection of PointValuePairs. These PointValuePairs, and the logic that organizes them into a rectangle, form the GridValueMatrix.

3.13.3. Hierarchy:

Superclasses:	***C_Function***

3.13.4. Associations:

Supplier : ***GridEvaluator*** in association: delegation

<no rolename> : ***GridValueMatrix*** in association: comprises

3.13.5. Public Interfaces:

Attributes:	interpolationType
	spatialReferenceSystem

3.13.6. Attributes:

interpolationType :	CharacterString = 'bilinear'
spatialReferenceSystem :	SpatialReferenceSystem

3.14. Class name: GridValueMatrix

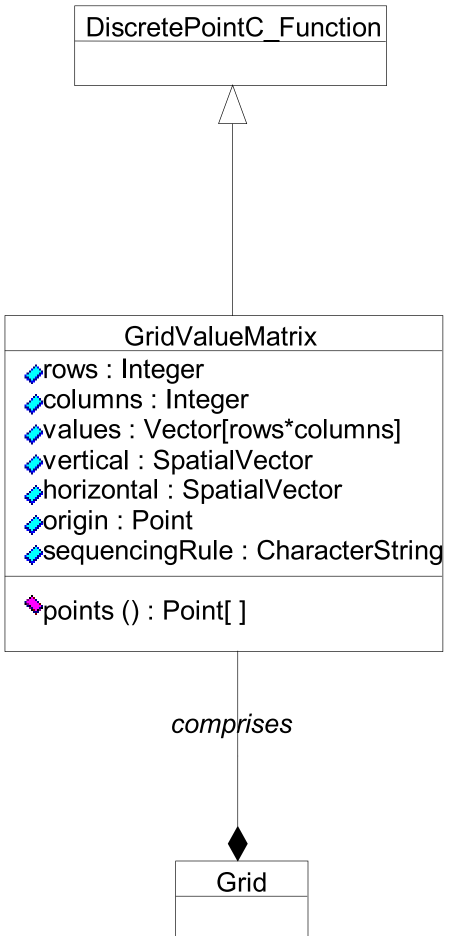


Figure 3-12. Context Diagram for *GridValueMatrix* Class

3.14.1. Category: *Coverages*

3.14.2. Documentation:

A member of the GridValueMatrix class is associated with a collection of PointValuePairs that make it a subtype of DiscretePointC_Function. In addition, a GridValueMatrix is associated with the defining origin and offset and size vectors of a grid geometry. The geometry represented by the various offset vectors is in the image plane of the Grid.

3.14.3. Hierarchy:

Superclasses: DiscretePointC_Function

3.14.4. Associations:

<no rolename> : ***Grid*** in association: comprises

3.14.5. Public Interface:

Attributes:	row
	columns
	values
	vertical
	horizontal
	origin
	sequencingRule
Operations:	points

3.14.6. Attributes:

rows:	Integer
Documentation:	The attribute rows is the number of rows in the matrix.
columns:	Integer
Documentation:	The attribute columns is the number of columns in the matrix. Similar structures might be added for higher dimensional matrices.
values:	Vector[rows*columns]
Documentation:	The attribute values is an array of vectors of length rows * columns. The vectors represent observations or phenomenon values at the respective grid locations.
vertical:	SpatialVector
Documentation:	The attribute vertical is the second offset vector that defines the direction and distance between elements of the same column. Similar structures could exist for higher dimensional GridValueMatrix.
horizontal:	SpatialVector
Documentation:	The attribute horizontal is the first offset vector that defines the direction and distance between elements of the same row.
origin: Point	
Documentation:	The attribute origin is the point in the initial row and initial column.
sequencingRule:	CharacterString
Documentation:	The sequencingRule attribute describes how the points in the grid geometry are ordered for association to the "values" attribute. The default is row major, other values will be from enumeration to be determined at a later date.

3.14.7. Operation name: points

Public member of:	GridValueMatrix
Return Class:	Point[ ]
Arguments:	num : Integer = rows*columns
Documentation:	An alternative (grid specific) implementation of the "*domain*" geometry function inherited from DiscreteC_Function. The operation returns the geometry of the GridValueMatrix. The order of the points returned is consistent with the sequencing rule.

3.15. Class name: GridEvaluator

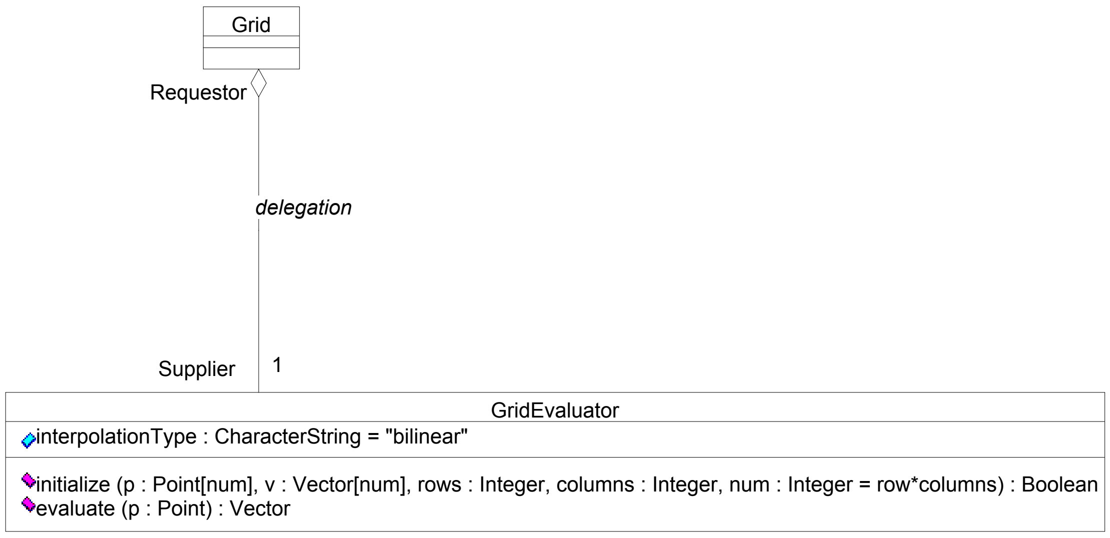

Grid
Requestor
delegation
Supplier
1
GridEvaluator
interpolationType : CharacterString = "bilinear"
initialize (p : Point[num], v : Vector[num], rows : Integer, columns : Integer, num : Integer = row*columns) : Boolean
evaluate (p : Point) : Vector

Figure 3-13. Context Diagram for *GridEvaluator* Class.

3.15.1. Category: Coverages

3.15.2. Documentation:

A GridEvaluator is a service that can be applied to Grids to determine the values of their Coverage Function at the passed Points of such a Coverage.

Stereotype:	service

3.15.3. Hierarchy:

Superclasses:	none

3.15.4. Associations:

Requestor: Grid in association: delegation

3.15.5. Public Interface:

Attributes:	interpolationType
Operations:	initialize
	Evaluate

3.15.6. Attributes:

interpolationType : CharacterString = "nearest neighbor"

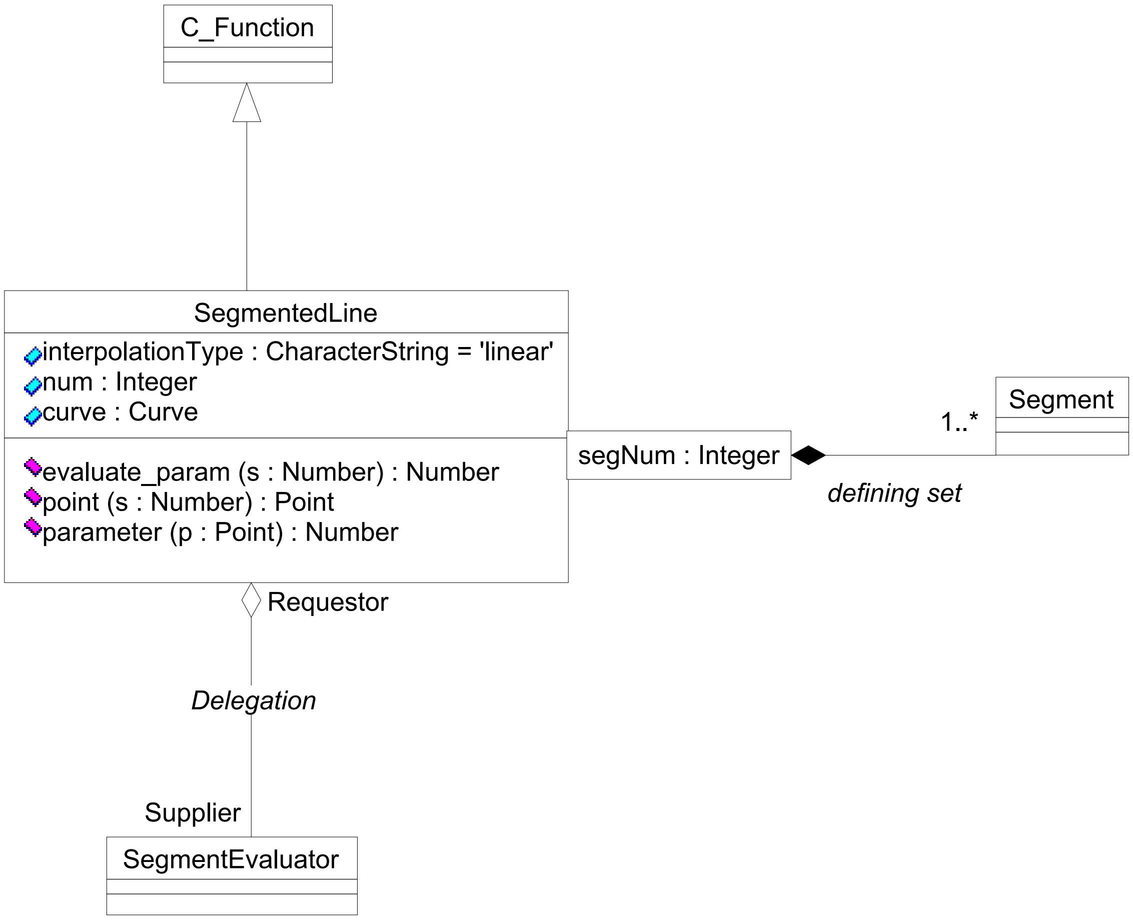
3.15.7. Operation name: initialize

Public member of:	GridEvaluator	
Return Class:	Boolean	
Arguments:	Point[num]	p
	Vector[num]	v
	Integer	rows
	Integer	columns
	Integer	num = rows*columns
Documentation:	This method takes point and value arrays from a grid and internally creates the stored function. It may require control parameters such as the number of rows and columns.	

3.15.8. Operation name: evaluate

Public member of:	GridEvaluator
Return Class:	Vector
Arguments:	Point p
Documentation:	This method evaluates the stored function at the passed point and returns the value vector.

3.16. Class name: Segmented Line



C_Function
SegmentedLine
interpolationType : CharacterString = 'linear'
num : Integer
curve : Curve
evaluate_param (s : Number) : Number
point (s : Number) : Point
parameter (p : Point) : Number
segNum : Integer
1..*
Segment
defining set
Requestor
Delegation
Supplier
SegmentEvaluator

Figure 3-14. Context Diagram for *SegmentedLine* Class.

3.16.1. Category: Coverages

3.16.2. Documentation:

A SegmentedLine is a C_Function whose spatial domain is a parameterized curve.

3.16.3. Hierarchy:

Superclasses: ***C_Function***

3.16.4. Associations:

<no rolename> : Segment in association: defining set

Supplier : ***SegmentEvaluator*** in association: Delegation (Service)

3.16.5. Public Interface:

Attributes:	interpolationType
	Num
	Curve
Operations:	evaluate_param
	point
	parameter

3.16.6. Attributes:

interpolationType :	CharacterString = 'linear'
num :	Integer
Documentation:	The attribute "num" is the number of segments used to define the C_Function.
curve:	Curve
Documentation:	The attribute "curve" is the domain of the C_Function. From the Abstract Specification Topic Volume 1, every curve has an associated parameterization. The parameterization of this curve within the segmented line provides the necessary information for the linearity.

3.16.7. Operation name: evaluate_param

Public member of:	SegmentedLine
Return Class:	Vector
Arguments:	Number s
Documentation:	The evaluate_param method returns the C_Function value of the point on the curve associated to the passed parameter.

3.16.8. Operation name: point

Public member of:	SegmentedLine
Return Class:	Point
Arguments:	Number s
Documentation:	The point method returns the point associated to the passed parameter.

3.16.9. Operation name: parameter

Public member of:	SegmentedLine
Return Class:	Number
Arguments:	Point p
Documentation:	The parameter method returns the point associated to the passed parameter.

3.17. Class name: SegmentEvaluator

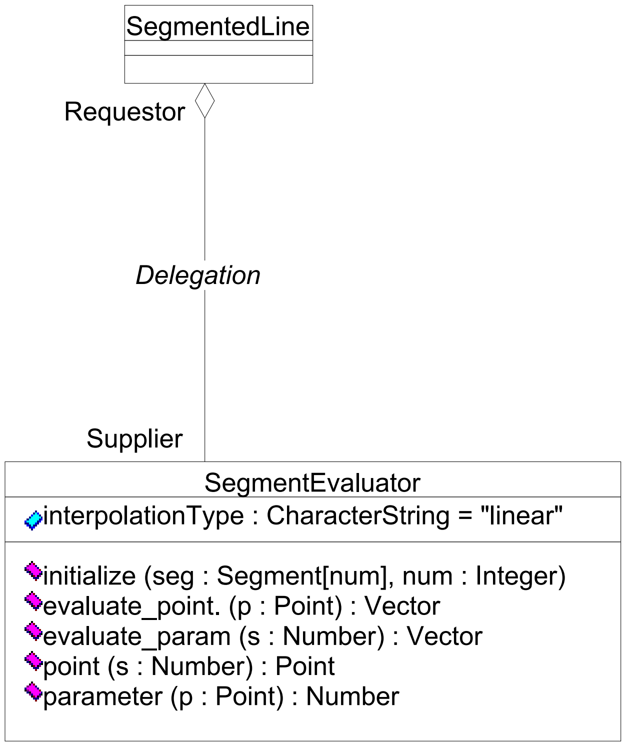

SegmentedLine
Requestor
Delegation
Supplier
SegmentEvaluator
interpolationType : CharacterString = "linear"
initialize (seg : Segment[num], num : Integer)
evaluate_point. (p : Point) : Vector
evaluate_param (s : Number) : Vector
point (s : Number) : Point
parameter (p : Point) : Number

Figure 3-15. Context Diagram for *SegmentEvaluator* Class.

3.17.1. Category: Coverages

3.17.2. Documentation:

Segment Evaluator is a service used to calculate the segment's ***C_Function*** at a passed point.

Stereotype:	service

3.17.3. Hierarchy:

Superclasses:	none

3.17.4. Associations:

Requestor : SegmentedLine in association: Delegation (Service)

3.17.5. Public Interface:

Attributes:	interpolationType
Operations:	Initialize
	Evaluate_point
	Evaluate_param
	Point
	Parameter

3.17.6. Attributes:

interpolationType : CharacterString = "linear"

3.17.7. Operation name: initialize

Public member of:	SegmentEvaluator
Arguments:	Segment [num]
	Integer num
Documentation:	The initialize operation initializes the segment evaluator with the segments and their end values.

3.17.8. Operation name: evaluate_point

Public member of:	SegmentEvaluator
Return Class:	Vector
Arguments:	Point p
Documentation:	The evaluate_point operation returns the C_Function value at a passed point.

3.17.9. Operation name: evaluate_param

Public member of:	SegmentEvaluator
Return Class:	Vector
Arguments:	Number s
Documentation:	The evaluate_param operation returns the C_Function value at a passed parameter. The default is the linear interpolation (using the parameter axis) of the values at the segment's ends.

3.17.10. Operation name: point

Public member of:	SegmentEvaluator
Return Class:	Point
Arguments:	Number s
Documentation:	The point method returns the point associated to the passed parameter.

3.17.11. Operation name: parameter

Public member of	SegmentEvaluator
Return Class:	Number
Arguments:	Point p
Documentation:	The parameter method returns the parameter associated to the passed point.

3.18. Class name: Segment

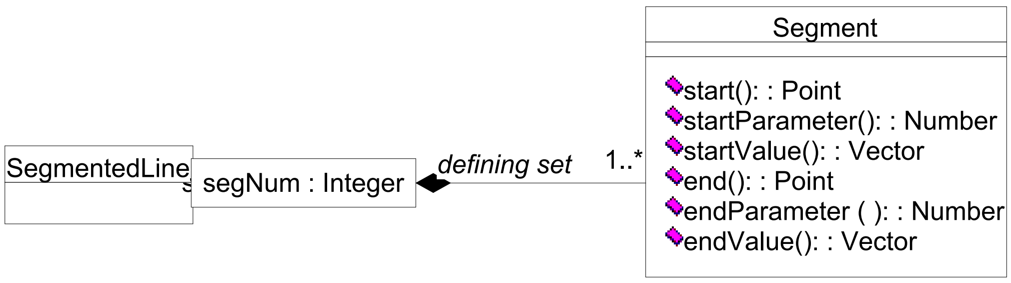


Figure 3-16. Context Diagram for *Segment* Class.

3.18.1. Category: Coverages

3.18.2. Documentation:

A segment is a one-dimensional geometric object that has a beginning (start) and end point.

3.18.3. Hierarchy:

Superclasses:	none

3.18.4. Associations:

<no rolename> : ***SegmentedLine*** in association: defining set

3.18.5. Public Interface:

Operations:	start():
	startParameter():
	startValue():
	end():
	endParameter ( ):
	endValue():

3.18.6. Operation name: start():

Public member of:	Segment
Return Class:	Point
Documentation:	The start method returns the beginning point in this segment.

3.18.7. Operation name: startParameter():

Public member of:	Segment
Return Class:	Number
Documentation:	The startParameter method returns the parameter of the beginning point of this segment.

3.18.8. Operation name: startValue():

Public member of:	Segment
Return Class:	Vector
Documentation:	The startValue method returns the vector value of the C_Function at the start point of this segment.

3.18.9. Operation name: end():

Public member of:	Segment
Return Class:	Point
Documentation:	The end method returns the last point in this segment.

3.18.10. Operation name: endParameter ( ):

Public member of:	Segment
Return Class:	Number
Documentation:	The endParameter method returns the parameter of the last point of this segment.

3.18.11. Operation name: endValue():

Public member of:	Segment
Return Class:	Vector
Documentation:	The endValue method returns the vector value of the C_Function at the last point of this segment.

4. Future Work

In addition to completion of the currently outstanding request for Implementation Specifications for Coverages (RFP5 - Access to OpenGIS Coverages), future work will involve alignment with ongoing work in the areas of geometry (Topic 1), spatial reference systems (Topic 2), coordinate transformations (Topics 3 and 16), quality (Topic 9), metadata (Topic 11), image exploitation services (Topic 15), etc.

5. Appendix A. Well Known Structures

Well Known Structures for Coverages are TBD. It is the purpose of Implementation Specification proposals to define Coverage Well Known Structures.