Una arquitectura para manejar datos continuos en SIG

Luis Polasek, Arturo Zambrano, Silvia Gordillo

LIFIA- Facultad de Informática, UNLP

CC 11 (1900) La Plata, Buenos Aires, Argentina

[pola, arturo, gordillo]@sol.info.unlp.edu.ar

Abstract

Las aplicaciones geográficas en muchos casos implican la definición y manipulación de información continua. Esto es de suma relevancia en áreas como meteorología y aplicaciones ambientales. Los sistemas de información geográfica carecen de herramientas que permitan el uso de este tipo de información, lo que muchas veces impide contar con un análisis completo. En este trabajo se describe una arquitectura orientada a objetos que permite el diseño y manipulación de datos continuos.

1. Introducción

Los Sistemas de Información Geográfica (SIG) proveen las herramientas necesarias para manipular y analizar entidades espaciales, es decir, que poseen una posición en el espacio. En general los SIG manejan dos modelos de datos diferentes, el modelo de vector y el de raster [Laurini93] [Aronof91]. Mientras que el modelo de vector se caracteriza por el reconocimiento de objetos individuales a los cuales se les asigna una topología (punto, línea o polígono) y una posición, en el modelo de raster se define una región determinada, se la subdivide en subregiones más pequeñas e indivisibles y se codifica un valor que describe un atributo relevante en esa área.

Si bien algunas operaciones resultan más complejas que otras, cualquiera de los dos modelos puede usarse en un SIG de una manera eficiente; incluso existen sistemas en los cuales es posible combinar información de ambos modelos.

Sin embargo estas dos perspectivas para definir la información geográfica no siempre resultan suficientes. Existe una gran variedad de aplicaciones en donde la información que se requiere es en esencia "continua"; por ejemplo cualquier aplicación de tipo climatológica deberá tener la capacidad de analizar fenómenos continuos, por ejemplo la temperatura, la humedad, etc. Si bien es cierto que existen modelos matemáticos para tratar con la continuidad de los fenómenos, también es cierto que no existen buenos modelos computacionales para tratar con este tipo de información [Kemp97].

El ejemplo más inmediato de un dato continuo puede ser una onda o cualquier señal analógica que varía en función del tiempo. Matemáticamente se trata de una función continua $f(t)$, donde t es el tiempo. Si tomamos dos instantes de tiempo no iguales t_0 y t_1, tenemos:

$$f(t_0) = a \quad y \quad f(t_1) = b \qquad \text{y supongamos que } a < b$$

Por ser una función continua sabemos que f toma todos los valores posibles comprendidos en [a , b] cuando se la evalúa en el intervalo $[t_0 , t_1]$.

La misma idea puede extenderse de una a dos dimensiones, definiendo el dominio de la función en un plano coordenado. Esta visión permite definir fenómenos continuos en el contexto de aplicaciones geográficas, a estos fenómenos se los denomina campos continuos.

Si pensamos que *f* representa el valor correspondiente a algún fenómeno físico que puede medirse en una región terrestre, es posible definir una función que se evalúe cuando se necesita conocer un valor en cualquier punto p_i perteneciente a dicha región.

Los datos pertenecientes a los campos continuos pueden ser de dos tipos:

i Escalar: los valores retornados por la función son números.

ii Vectorial: los valores retornados por la función son vectores.

Un valor escalar es un dato de una dimensión, por ejemplo un número.

Un valor vectorial es un dato de n-dimensiones, por ejemplo el gradiente en un punto, que se expresa mediante una tupla de la forma:

$$(d_1, d_2, ..., d_n)$$

La solución que se adopta para manipular información de este tipo es realizar una discretización sobre los datos. De esta forma se toma una muestra de los valores del fenómeno en algunas posiciones dentro del área de estudio y el resto de los valores se calculan a través de algún método de estimación definido; éste método de estimación utiliza los valores de la muestra como punto de partida para el cálculo.

En la mayoría de las aplicaciones SIG existentes en la actualidad, el proceso descripto anteriormente se realiza fuera del sistema, ya que no se cuenta con los mecanismos necesarios ni para representar los valores de la muestra, ni para realizar las estimaciones y mucho menos para operar con información de este tipo. Así entonces, los puntos adquiridos son guardados en la base de datos del sistema, cuando se desea realizar una consulta referente al valor de la función en un punto que no pertenece a la muestra se toman sus valores y se los exporta a alguna aplicación específica para calcular el valor deseado mediante interpolación, luego se importan los valores producidos de manera que pueden luego ser o no incorporados a la base de datos (para acelerar futuras consultas).

En [Gordillo98] se presenta una arquitectura para especificar campos continuos en aplicaciones SIG contemplando los aspectos antes mencionados. El objetivo del presente trabajo es definir una serie de operaciones esenciales sobre los campos continuos y en tal sentido, extender dicha arquitectura de manera de obtener la capacidad de manipular la información.

El trabajo está organizado de la siguiente manera en la Sección 2 se resume la arquitectura presentada en [Gordillo98] y que constituye el punto de partida para la definición de las operaciones sobre campos continuos. En la sección 3 se describen las operaciones básicas de campos continuos. En la sección 4 se describe la arquitectura de diseño para estas operaciones. Finalmente en la sección 5 se presentan las conclusiones y los trabajos futuros.

2. Representación de Campos Continuos

El modelo presentado en [Gordillo98] es un modelo orientado a objetos que define un conjunto de clases y relaciones para especificar aplicaciones geográficas. Una de las características de este modelo es que permite representar campos continuos como una clase de objeto en el sistema. La Figura 1 muestra esta arquitectura.

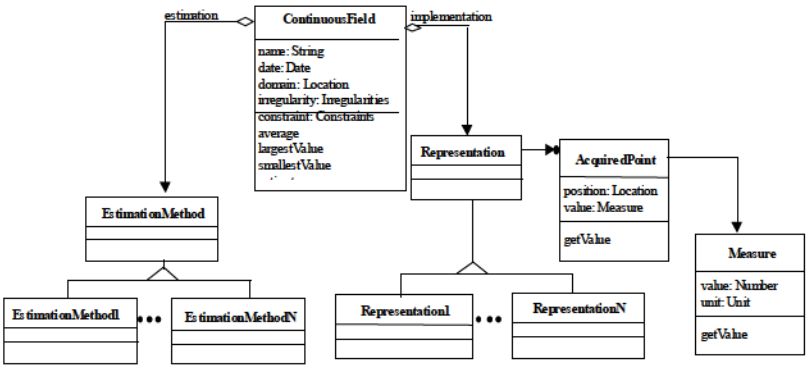


Figura 1. Representación de Campos Continuos

La clase *ContinuousField* contiene información básica como por ejemplo el fenómeno que está representando, la fecha en que fue creado y el dominio que representa la región de estudio. Además un campo continuo puede representarse de distintas maneras, dependiendo de la distribución de los puntos de la muestra y del tipo de método estimación que desee usarse para conocer el valor del campo en un punto que no pertenece a la muestra. Por ejemplo mediante una ***grilla regular*** donde los puntos de la muestra están distribuidos en los vértices de una cuadrícula. Otra posible representación podría ser una ***grilla irregular*** de puntos, o un TIN (Triangulated Irregular Network). Para modelar las distintas representaciones que puede tener un campo continuo en esta arquitectura se utilizó el pattern Bridge [Gamma95]. Como se desprende de lo anterior, en un campo continuo también puede variar la forma en la que se calculan los valores del campo en puntos que no pertenecen a la muestra. La forma en que se calculan esos valores es mediante el ***método de estimación*** del campo continuo y determina cómo se utiliza la muestra del campo para hallar los valores deseados. En este caso se aplicó el patrón de diseño Strategy [Gamma95]. Además todo campo continuo se halla definido en una región geográfica que llamaremos dominio del campo.

Esta arquitectura permite modelar mediante objetos los campos continuos dentro de una aplicación SIG, pero no es suficiente para realizar consultas complejas y recuperar la información. Para esto es necesario proveer de un conjunto de operaciones y mecanismos de consulta apropiados.

Suponiendo que existe un conjunto suficiente de operaciones válidas para campos continuos podemos pensar cualquier consulta como la aplicación sucesiva de varias operaciones entre los campos intervinientes; en la siguiente sección se explica detalladamente qué es una operación, qué tipos de operaciones se realizan sobre estos datos y cómo se resuelven.

3. Operaciones sobre Campos Continuos

En esta sección se presenta un conjunto de operaciones sobre campos continuos, teniendo en cuenta la variedad en la complejidad de las mismas.

La clasificación propuesta se basa en la complejidad de los factores que se deben tener en cuenta a la hora de resolver una operación determinada. Esta complejidad, en la mayoría de los casos está dada por la cantidad de campos continuos que intervienen en la operación, dado que esto define una serie de consideraciones que se deben tener en cuenta en el momento de resolverla. Así, la idea es dividir a las operaciones en unarias y n-arias.

Tenemos entonces dos grandes tipos de operaciones:

i Unarias: donde interviene un campo continuo y posiblemente algún otro valor.

ii N-arias: donde intervienen 2 o más campos continuos.

3.1 Operaciones Unarias

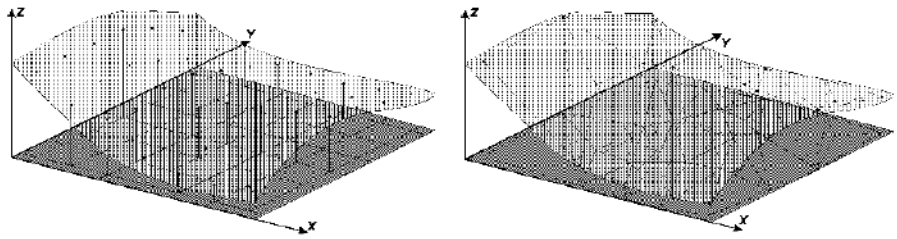
En la mayoría de los casos las operaciones unarias trabajan con los valores de un campo continuo para obtener un valor representativo de la muestra. Algunos ejemplos de operaciones unarias son:

- Evaluate: retorna el valor asociado a una posición (posiblemente una estimación)
- Average: retorna el promedio de valores del campo.
- Maximum: retorna el mayor valor en el campo.
- Minimum: retorna el menor valor en el campo.
- Slope: retorna la pendiente en una posición del campo.
- Select: retorna un nuevo campo continuo cuyo dominio es aquel que cumple ciertas condiciones.
- ConvexHull: retorna el cerco convexo de las muestras de un campo.
- RepresentationChange: retorna un nuevo campo continuo con la misma muestra y dominio del campo original, pero con una representación diferente.

Aquí se ve claramente cómo en algunas operaciones interviene sólo el campo continuo, por ejemplo ***average, minimum***, y en otras es necesario proveer un parámetro más, como en la operación ***select:***, que requiere además un criterio de selección.

Como ejemplo presentamos la operación ***RepresentationChange.*** Un posible uso de esta operación es obtener, a partir de una representación dada (Grilla Regular), una nueva (TIN).

En la Figura 2 se puede observar este cambio de representación de un campo continuo. Nótese que el campo continuo sigue siendo el mismo, es decir modela los mismos datos, la única diferencia es la forma en que se representan y posiblemente el método de estimación empleado.



Grilla Regular *TIN*

Figura 2. Cambio de representación (Grilla Regular -> TIN)

Dado que la complejidad en la resolución de operaciones unarias es relativamente baja, estas operaciones estarán implementadas dentro de la clase ***ContinuousField.***

3.2 Operaciones N-arias

Estas operaciones resultan de mayor complejidad a la hora de su resolución. Algunas de las operaciones típicas que se incluyen en esta categoría son:

- Average: retorna un campo continuo donde cada valor asociado a una posición es el promedio de los valores asociados a la misma posición en cada campo de entrada.
- Maximun: retorna un campo continuo donde cada valor asociado a una posición es el máximo de los valores asociados a la misma posición en cada campo de entrada.

- Minimun: retorna un campo continuo donde cada valor asociado a una posición es el menor de los valores asociados a la misma posición en cada campo de entrada.
- Arithmetic Operations: la idea es incluir todas aquellas operaciones definidas por Dana Tomlin en su MapAlgebra [Tomlin90].
- Union: retorna un campo continuo cuyo dominio es la unión de los dominios asociados a varios campos continuos
- Intersection: retorna un campo continuo cuyo dominio es la intersección de los dominios asociados a varios campos continuos.
- Difference: retorna un campo continuo cuyo dominio es la diferencia de los dominios asociados a dos campos continuos.

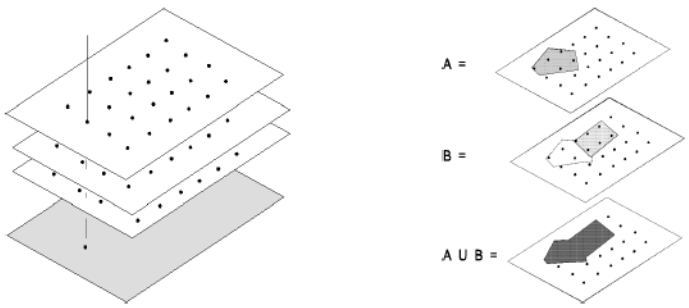
Un aspecto importante al tratar las operaciones n-arias es que trabajar con varios campos continuos puede involucrar que estos se encuentren definidos en diferentes dominios, es decir las regiones que representan no son exactamente las mismas; que los valores de sus respectivas muestras no hayan sido tomados en las mismas posiciones; y que se hayan utilizado diferentes representaciones y métodos de estimación. Todas estas posibilidades deberán ser tenidas en cuenta cuando se trate de operar con varios campos continuos. Concretamente los siguientes cuatro aspectos deberán ser analizados.

- Definición de la muestra del resultado.
- Elección de una representación apropiada.
- Definición del método de estimación (acorde a la representación).
- Definición del dominio sobre el cual estará definido el nuevo campo continuo.

El análisis que involucran los aspectos anteriores depende además del tipo de operación que se está resolviendo. Específicamente, no se pueden analizar de la misma manera operaciones que involucran toda una región del campo, es decir, que se definen a partir del dominio , que las que específicamente operan sobre posiciones puntuales.

Claramente entonces, estas operaciones pueden ser sub-clasificadas en:

- Operaciones de zona o dominio: como Intersection, Union y Difference.
- Operaciones puntuales: como Average, Maximum, Arithmetic Operations, etc. Las llamamos puntuales porque son aplicadas sobre la misma posición en todos los campos continuos simultáneamente, como puede observarse en la Figura 3.



Una operación puntual Una operación de zona

Figura 3. Diferentes tipos de operaciones n-arias

4. Modelización de Operaciones de Campos Continuos

La aplicación de una operación sobre uno o varios campos continuos debería producir resultados (escalares, vectores, campos continuos) que pueden ser utilizados posteriormente para otras operaciones. El marco para que los resultados de una operación sean utilizados por otra, está dado por las consultas que un usuario realizará a través de un lenguaje específico. Es importante aclarar que en este trabajo no se habla de la definición de ningún lenguaje particular, sino de la especificación de las operaciones que luego serán utilizadas por un lenguaje para resolver un requerimiento particular. Una operación se aplica sobre un conjunto de campos continuos de entrada. A su vez los campos resultantes pueden ser utilizados por otras operaciones. Vemos entonces que hay una relación recursiva en la resolución de un requerimiento específico, ya que un resultado puede depender del resultado de otros.

De esta manera se forma una estructura similar a la de un árbol de evaluación, donde se van obteniendo resultados parciales que luego son combinados con otros para determinar el resultado final.

Con el objetivo de modelar los diferentes tipos de consulta se utilizó el patrón de diseño Composite [Gamma95], la Figura 4 muestra la definición de esta estructura.

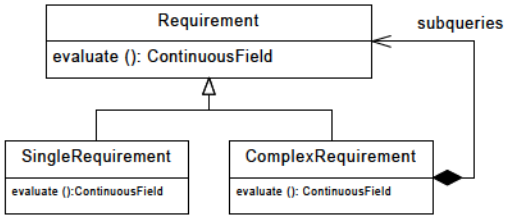


Figura 4. El pattern Composite aplicado a la jerarquía de Requirement.

La clase *SingleRequirement* tiene la responsabilidad de retornar un campo continuo cada vez que es requerido para alguna evaluación.

Un *ComplexRequirement* toma como entrada los resultados de la evaluación de uno o más *requeriments* y les aplica alguna operación.

Los requerimientos de manipulación con diferentes niveles de complejidad pueden resolverse mediante la aplicación sucesiva de operaciones primitivas entre campos continuos. Estas operaciones se resuelven en una serie de etapas de evaluación que producen resultados parciales utilizables en las siguientes fases.

Anteriormente mencionamos que para definir un campo continuo es necesario tomar en cuenta diversos aspectos, como la muestra, la representación, método de estimación y dominio que lo definen. Para no recargar todo este conocimiento en los *Requeriments*, incluimos en nuestra arquitectura una nueva clase de objetos que llamamos *Evaluators*. Dichos objetos evalúan operaciones aplicadas a un conjunto de campos continuos y son los encargados de definir en cada etapa cual será el dominio del campo continuo resultado, cual será su representación, muestra, etc.

Estos objetos son configurados con la operación que deben aplicar y un conjunto de colaboradores que definirán las características del campo continuo resultante.

En las siguientes secciones mostraremos como se resuelve cada uno de estos aspectos para las operaciones n-arias.

4.1 Obtención del Dominio

El objetivo en esta fase es determinar la región geográfica que define al campo continuo resultante, para ello recurrimos a las operaciones de zona que se aplican sobre los dominios de los campos continuos de entrada y generan como resultado un dominio para el campo continuo resultante.

Estas operaciones serán la ***unión, intersección y diferencia*** aplicadas a los polígonos que definen el dominio de los campos continuos de entrada (ver Figura 3).

En la definición de la solución, se utilizó el pattern Strategy (ver Figura 5) para permitir que los evaluadores sean configurados acorde al dominio de campo resultante que se espera obtener.

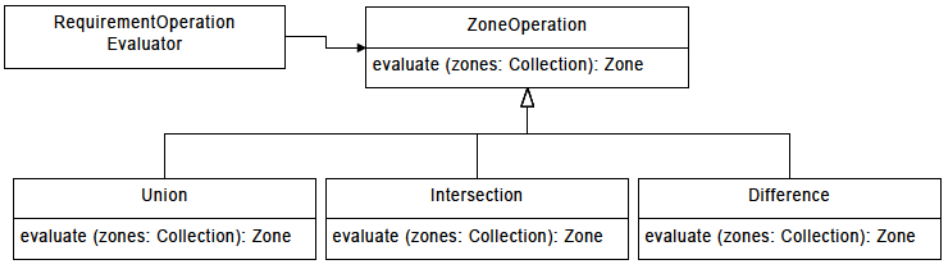


Figura 5 Jerarquía de estrategias para obtener el dominio

4.2 Obtención de la Muestra

Dado que es muy probable que al trabajar con diferentes campos continuos nos encontremos con muestras que han sido tomadas en diferentes puntos geográficos, es importante definir una estrategia a seguir para generar los puntos que van a conformar la muestra del campo continuo resultado. De la selección de esos puntos dependerá la densidad de la muestra y la precisión del campo generado. Esta estrategia deberá se una de las detalladas a continuación:

Muestra Arbitraria: Es posible que en ciertas ocasiones se desee generar la muestra del campo resultado en ciertos puntos elegidos por conveniencia, en este caso la densidad y la regularidad de la muestra dependerá de los puntos elegidos, así como la precisión en los datos dependerá de cuántos de esos puntos elegidos coincidan con las muestras de los campos de entrada.

Muestra Diferencia: Los puntos en el campo continuo resultante serán aquellos que pertenezcan a uno de los campos continuos, pero no al otro.

Muestra Unión: Con esto nos referimos al caso en que se decide que la muestra del campo resultado esté compuesta por todos puntos que pertenecen las muestras de todos los campos. Es decir se realiza la unión de los conjuntos que forman las muestras de los campos de entrada. Debería ser utilizada cuando se desea obtener un campo resultado con una muestra bastante densa, esta condición de densidad puede ser beneficiosa a la hora de interpolar datos, pues los métodos de estimación trabajan mejor con datos cercanos al punto a interpolar, pero tiene como desventaja que muchos de los valores que se extraerán de los campos de entrada deberán ser interpolados en los campos continuos de entrada con la lógica pérdida de precisión en los puntos que conforman la muestra.

Muestra Intersección: En este caso se toma como puntos para la muestra del campo resultado aquellos puntos que se encuentran en las muestras de todos los campos de entrada, se realiza la intersección de los conjuntos que representan las posiciones de las muestras en los campos de entrada. Esta opción proveerá una muestra para el campo resultado cuya densidad es menor o igual a la de los campos de entrada. Como principal ventaja se puede observar que para generar los valores asociados al campo resultado no será necesario realizar interpolaciones, con lo cual no se pierde calidad en la muestra. Como contraparte, las estimaciones generadas a partir del campo continuo resultante podrían llegar a ser menos precisas que en el caso anterior, donde se tienen más puntos en la muestra para realizar la interpolación.

Para proveer la posibilidad de configurar los ***Evaluators*** con las diferentes formas de obtener la muestra se utilizó el patrón de diseño Strategy [Gamma95]. De esta manera se gana en flexibilidad, pudiendo configurarlos con la estrategia más conveniente de acuerdo a los campos de entrada y a la operación que se aplicará, como se muestra en la Figura 6.

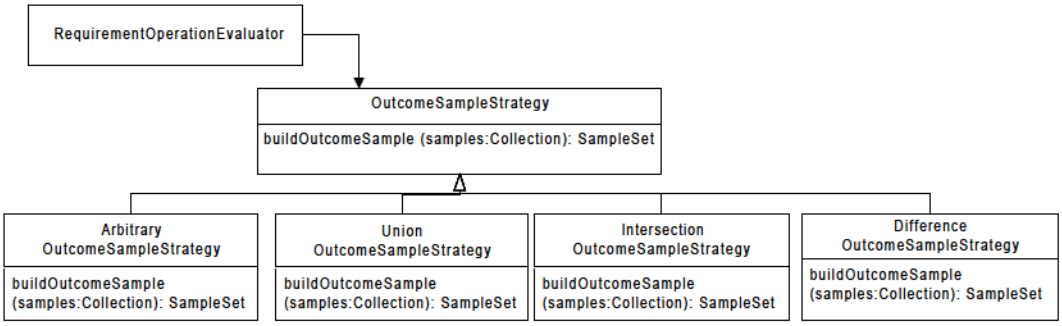


Figura 6. Jerarquía de estrategias para la obtención de la muestra

4.3 Obtención de los Valores de la Muestra

Para obtener los valores asociados a cada punto de la muestra resultante es necesario aplicar una función a los valores de las muestras de los campos continuos de entrada (operación puntual). Concretamente las operaciones puntuales se comportan generando un valor asociado a la posición p del campo continuo resultado, que es función de los n valores, cada uno asociado a la posición p de los n campos continuos de entrada (ver Figura 3). El resultado obtenido es asociado a la posición p del campo continuo resultante y pasa a formar parte de su muestra. De esta manera se determinan los valores correspondientes a los puntos de la muestra del nuevo campo. En la Figura 7 mostramos algunas de las operaciones puntuales que se pueden utilizar para determinar el valor asociado a los puntos de la muestra resultante y con las cuales puede ser configurado el evaluador.

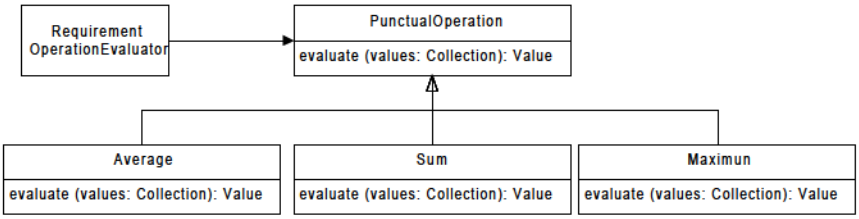


Figura 7. Jerarquía de Operaciones Puntuales

4.4 Arquitectura Final

En función de todas las consideraciones a tener en cuenta en la resolución de las operaciones, la arquitectura final se presenta en la Figura 8.

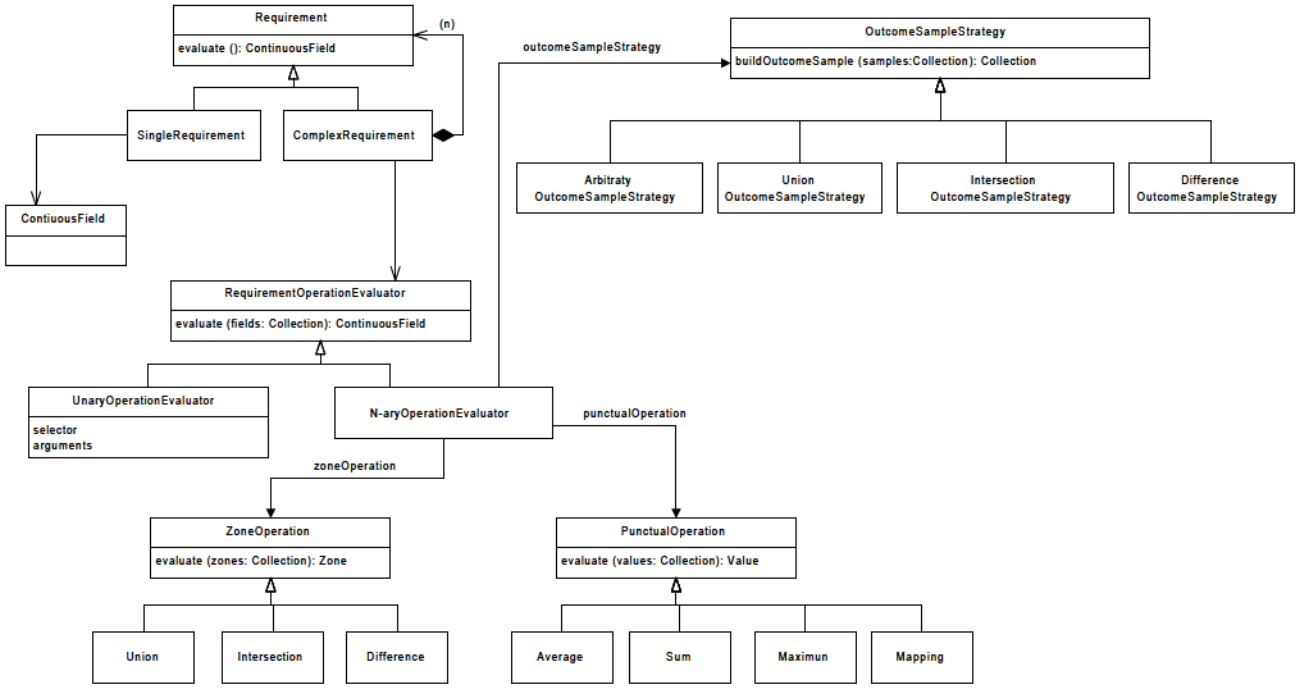


Figura 8. Arquitectura completa para realización de consultas y operaciones entre campo continuos

Donde se puede observar la integración de todos los elementos que hemos ido presentando por separado. La jerarquía establecida por la clase *Requeriment* está definida, como dijimos, por el patrón de diseño *Composite*. Los *ComplexRequirement* poseen un evaluador encargado de aplicar una dada operación a un conjunto de campos continuos, y se pueden configurar con diferentes estrategias para la obtención de muestras y dominios.

En cada etapa de evaluación se produce un nuevo campo que es utilizado como resultado parcial para calcular el resultado final de la consulta. El objeto responsable de determinar la "forma" del nuevo campo continuo es el evaluador, que esta configurado con una serie de colaboradores:

- Una instancia de alguna de las subclases de *OutcomeSampleStrategy*, que le permite definir qué puntos pertenecen a la muestra del campo continuo resultante.
- Una instancia de una subclase de *ZoneOperation*, que determina cual será el dominio sobre el que estará definido el nuevo campo.
- Una instancia de una subclase de *PunctualOperation*, que calcula el valor asociado a cada punto de la muestra del nuevo campo.

En el diagrama de interacción de la Figura 9 se muestra cómo una instancia de la clase *RequirementOperationEvaluator* genera un campo continuo a partir de dos campos de entrada. Vemos que el evaluador delega la responsabilidad de elegir los puntos de la muestra a una instancia de una subclase de *OutcomeSampleStrategy*. Para la definición del dominio el objeto evaluador utiliza una instancia de una subclase de *ZoneOperation* que toma como entrada los dominios de los campos de

entrada y genera el dominio del campo resultado. Luego, para determinar el valor asociado a cada punto (posición) que pertenece a la muestra resultante se realiza la siguiente operación:

- Se consulta el valor del fenómeno asociado a esa posición en todos los campos continuos de entrada (el valor obtenido podría ser producto de una estimación).
- Una instancia de una subclase de *PunctualOperation* genera un nuevo valor basándose en todos los valores obtenidos en el paso anterior.
- El valor obtenido se asocia a la misma posición en el campo continuo resultante.

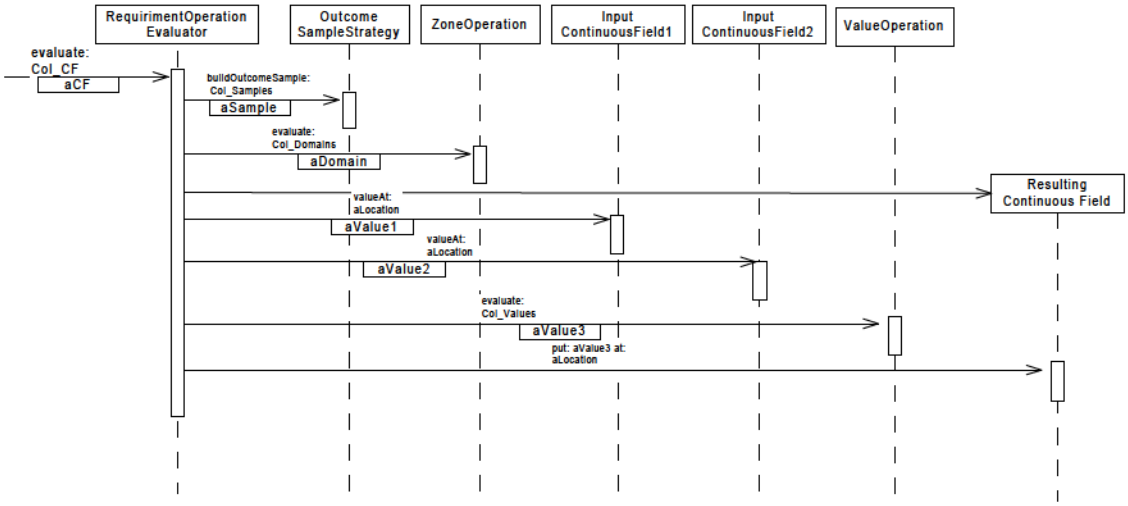


Figura 9 Diagrama de Interacción: una etapa de evaluación

5. Conclusiones y Trabajo Futuro

En este trabajo se presentó una arquitectura que permite operar con información continua. La misma se encuentra parcialmente implementada en un lenguaje orientado a objetos. En particular las operaciones puntuales, es decir las que operan sobre valores en determinadas posiciones del campo, están probadas.

Se está trabajando ahora en la implementación de operaciones de zona y se están estudiando los algoritmos necesarios para obtener mayor eficiencia.

Además se está integrando todo el manejo de campos continuos con el de objetos discretos de forma de obtener no solo la posibilidad de representar información continua, sino también de combinarla con información proveniente de los modelos de raster y vector.

6. Bibliografía

[Aronof91] S. Aronoff, “Geographic Information Systems : A Management Perspective”. WDL Publications, 1991

[Gamma95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: "Design Patterns. Elements of reusable Object-Oriented Software". Addison Wesley, 1995.

[Gordillo98] S. Gordillo, F. Balaguer. “Refining an object-oriented GIS design model: Topologies and Field Data”. ACM-GIS’98. November 6-7, 1998. Washington, D.C., USA.

[Kemp97] K. Kemp. “Fields as a Framework for integrating GIS and environmental process models”.
Part 1: “Representing spatial continuity”.
Part 2: “Specifying field variables”.
Transactions in GIS, 1997, vol. 1, nº 3, p. 219-234 and 235-246.

[Laurini93] R. Laurini, D. Thompson, “Fundamentals of Spatial Information Systems”. Academic Press Professional, 1993.

[Tomlin90] C. D. Tomlin: “Geographic Information Systems and Cartographic Modeling”. Prentice-Hall, Inc., 1990. New Jersey.