GGL2: Un lenguaje específico para SIG

F. González [1], V. González [2]

[1] Freelance, proyecto GearScape, fernando@fergonco.es.
[2] Freelance, proyecto GearScape, vicgonco@inf.upv.es.

RESUMEN

GearScape es un Sistema de Información Geográfica (SIG) que cuenta con un lenguaje llamado Gearscape Geoprocessing Language (GGL) que permite la programación y reutilización de cadenas de geoprocesamiento de forma rápida y sencilla.
A lo largo del último año se han publicado distintas versiones de GearScape, con numeración 0.x, que han sido utilizadas en diversos escenarios. Ello ha permitido la detección de algunos fallos de diseño en GGL que imposibilitaban su aplicación en un gran número de problemas, limitando así su expansión.
Por una parte, las versiones 0.x operan sobre datos organizados en forma tabular (filas y columnas) y, si bien este formato ha sido ampliamente utilizado (shapefiles, ...), la aparición del estándar GML ha hecho que cada vez haya más datos organizados jerárquicamente, como CityGML y GPX (GPS Exchange Format).
Por otro lado, GearScape es el único SIG que permite el uso de GGL por lo que cualquier usuario que esté interesado en realizar geoprocesamiento con éste está forzado a arrancar un nuevo programa.
Para solucionar estos y otros problemas, se ha desarrollado una nueva versión del lenguaje de GearScape, al que se le ha dado el nombre de GGL2. GGL2 es un lenguaje específicamente diseñado para la manipulación de datos SIG, por lo que ofrece construcciones concretas para la creación de geometrías (WKT), procesado de datos vectoriales (SQL espacial), procesado de rasters, extracción de datos XML (XPath), etc.
Además, el uso de GGL2 no está restringido a GearScape y el usuario puede arrancar el editor de GGL2, "conectarlo" a otro software, como por ejemplo a gvSIG, y empezar a procesar las fuentes de datos existentes en dicho sistema.

Palabras clave: geoprocesamiento, GGL, GearScape, SIG, *software libre.*

ABSTRACT

GearScape is a Geographic Information System (GIS) which provides a language, called Gearscape Geoprocessing Language (GGL), that allows the development and reuse of geoprocessing chains in a simple and fast manner.

Throughout the last year, several versions of Gearscape with 0.x numbering have been published and have been used in different scenarios. This has made possible the detection of some design problems that made impossible the application of the language in a high number of problems and, therefore, restricting its expansion.
On one hand, 0.x versions handle only tabular data (rows and columns) and, despite this format has been widely used (shapefiles, ...), with the appearance of the GML standard the amount of hierarchical data, such as CityGML or GPX (GPS Exchange Format), has grown considerably.
On the other hand, GearScape is the only GIS with GGL capabilities, so any user interested in performing geoprocessing with this language must run another application.
In order to solve these and other problems, a new version of the Gearscape language has been developed and it has received the name of GGL2. GGL2 is a language designed specifically for the management of GIS data and, therefore, it provides constructions for geometry creation (WKT), vector data processing (spatial SQL), raster processing, XML data extraction (Xpath), etc.
Moreover, the use of GGL2 is not restricted to Gearscape so the user can run the GGL2 editor, connect it to a GIS application, such as gvSIG and start processing the data sources provided by the system

Key words: *geoprocessing, GGL, GearScape, GIS, free software.*

INTRODUCCIÓN

GearScape es un Sistema de Información Geográfica (SIG) que cuenta con un lenguaje llamado Gearscape Geoprocessing Language (GGL), cuyo objetivo fundamental consiste en la programación y reutilización de cadenas de geoprocesamiento de forma rápida y sencilla.

Es posible considerar GGL como un lenguaje de dominio específico (en adelante DSL por sus siglas en inglés, *Domain Specific Language*), entendiendo como tal a cualquier lenguaje de programación o lenguaje de especificación ejecutable que ofrece, mediante una notación y abstracciones apropiadas, una gran potencia expresiva centrada (y generalmente restringida) en el dominio de un problema particular [1]. Entre las ventajas de los DSLs encontramos la gran cercanía del lenguaje de programación al lenguaje natural utilizado por los expertos del dominio para especificar el problema, las optimizaciones que pueden tenerse en cuenta al considerar un único tipo de problemas, o el hecho de que el código para solucionar un determinado problema puede considerarse, además, su documentación.

Por otra parte, muchas disciplinas cuentan ya con su propio lenguaje de dominio específico, como pueden ser R para la estadística, Mathematica para el álgebra lineal y las matemáticas simbólicas, VHDL para la descripción de componentes hardware o el propio SQL para bases de datos relacionales.

De la misma manera, GGL surge con el fin de acercar la resolución de problemas propios del área de los SIGa los expertos en dicho campo y plantea una serie de ventajas e inconvenientes. En las siguientes secciones se describe brevemente dicho lenguaje, así como los principales problemas que plantea. Posteriormente se muestra cómo la segunda versión del lenguaje (GGL2) los soluciona y presenta nuevas funcionalidades. Finalmente se muestran ejemplos de aplicación del nuevo lenguaje, junto con las conclusiones obtenidas.

GGL

Como se ha comentado anteriormente, GGL es un lenguaje creado con el fin de adecuarse lo máximo posible a la problemática del manejo de datos espaciales en SIGs. De esta manera, implementa un subconjunto de las características del estándar SQL92 [2] y se extiende espacialmente mediante la especificación OGC para fenómenos simples [3][4].

Entre las principales ventajas de este lenguaje encontramos, aquellas propias de un DSL y que se han mencionado en el apartado anterior, entre las que cabe destacar la simplicidad que proporciona un lenguaje declarativo como SQL que, a su vez, disminuye en gran medida la posibilidad de incluir errores en la creación de procesos [5].

Además, a lo largo del último año se han publicado distintas versiones de GearScape, con numeración 0.x, que han sido utilizadas en diversos escenarios. Ello ha permitido la detección de algunos fallos de diseño en GGL que imposibilitaban su aplicación en un gran número de problemas, limitando así su expansión.

Por una parte, puesto que la base principal de GGL es el estándar SQL, las versiones 0.x operan sobre datos organizados en forma tabular (filas y columnas) y, si bien este formato ha sido ampliamente utilizado (*shapefiles*, ...), la aparición del estándar GML ha hecho que cada vez haya más datos organizados jerárquicamente, como CityGML y GPX (GPS Exchange Format).

Por otro lado, GearScape es el único SIG que permite el uso de GGL por lo que cualquier usuario que esté interesado en realizar geoprocesamiento con éste último está forzado a utilizar este SIG, que en ciertos ámbitos puede resultar menos adecuado que otro SIG, o simplemente puede requerir el uso de dos SIG simultáneamente, lo cual puede repercutir de manera negativa en la productividad y estabilidad del sistema en conjunto.

Para solucionar estos y otros problemas, se ha desarrollado una nueva versión del lenguaje de GearScape, al que se le ha dado el nombre de GGL2. En la siguiente sección se describe este nuevo lenguaje, el lugar que ocupa dentro de los SIG y las principales ventajas que presenta.

GGL2

Tal y como se ha descrito en apartados anteriores, GGL2 es un DSL surgido de la necesidad de disponer de un lenguaje específico para GIS y con el fin de solucionar los problemas que plantea la primera versión de GGL, así como proporcionar nuevas funcionalidades, soluciones y herramientas para facilitar y optimizar la especificación de geoprocesos.

Para ello dispone de distintas instrucciones que permiten tratar tanto con datos espaciales tabulares de manera muy similar a la aproximación de SQL, como con datos organizados de manera jerárquica, siendo un ejemplo claro de estos últimos ficheros GML o, más ampliamente, XML.

En las siguientes subsecciones mostraremos la arquitectura del sistema y cuál es su posición con respecto a otros SIG. Además, describiremos las principales características del lenguaje y destacaremos las ventajas más importantes del entorno de edición que proporciona GGL2.

Arquitectura

El entorno de trabajo para GGL2 está diseñado para permitir la conexión con diferentes SIG. Una vez el entorno está conectado a un SIG es posible hacer uso de las fuentes de datos definidas en él.

Para conseguir esto, es necesario que se produzca una comunicación entre ambos componentes. Esta comunicación consiste fundamentalmente en dos llamadas iniciadas por el entorno de GGL2: una de obtención de datos y otra de envío de resultados, tal y como se puede apreciar en la Figura 1.

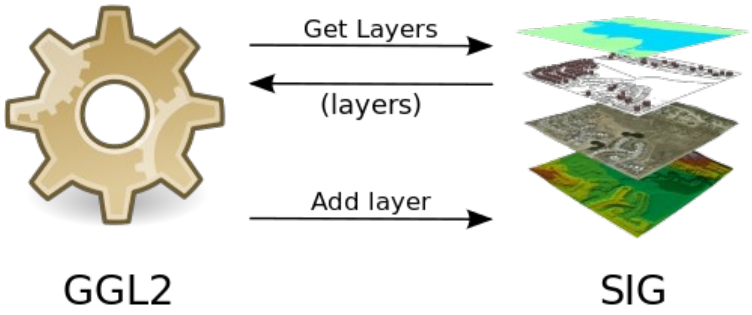

Figura 1: *Conexión de GGL2 a un SIG.*

Además, es importante destacar que esta comunicación se produce de manera completamente transparente al usuario, de forma que éste únicamente se debe preocupar de especificar su geoproceso utilizando las capas del SIG y el entorno de trabajo se ocupará de realizar obtener del GIS la información necesaria para que el sistema se comporte como una única unidad de ejecución. También hay que decir que los datos que se transfieren en esta comunicación no son los propios datos de las fuentes de datos, lo cual sobrecargaría innecesariamente la comunicación, sino que se transfieren referencias a los datos; esto es, la descripción de la ubicación y la forma de acceder a los datos a los que está accediendo el SIG.

En cuanto a la ejecución, el compilador de GGL2 transforma de manera automática el código escrito por el usuario en código Java, que es finalmente ejecutado para obtener los resultados. Si bien actualmente éste es el único comportamiento de GGL2, cabe destacar que el sistema está diseñado para poder transformar el código GGL2 en cualquier otro formato textual. Esto permite que en un futuro el entorno de trabajo se amplíe y permita la generación de código en C, de plugins para una plataforma GIS particular o incluso de documentación sobre los geoprocesos.

Características

GGL2 es un lenguaje que toma como base a Java, por lo que se trata de un lenguaje imperativo y fuertemente tipado. Es decir, por un lado las instrucciones se ejecutan de manera secuencial, modificando el estado del sistema; y por otra parte es posible declarar variables que contienen datos (o referencias a éstos) especificando de manera explícita la estructura y el tipo de esos datos. GGL2 es capaz de manejar fuentes de datos espaciales tanto tabulares como jerárquicas, por lo que define tanto un conjunto de tipos básicos, como la posibilidad de construir tipos compuestos en base a tipos básicos, de manera muy similar a las estructuras de datos en C (*structs*). Por último, el lenguaje proporciona un conjunto de instrucciones que permiten la

realización de operaciones sobre estas variables con el fin de obtener el resultado deseado.

En las siguientes subsecciones definiremos los tipos de datos básicos, así como el mecanismo para construir tipos de datos compuestos, junto con las instrucciones más importantes para el manejo de variables.

Tipos de datos

En primer lugar, en lo que respecta a los tipos de datos básicos, éstos son: booleanos (**`boolean`**), coma flotante en simple (**`float`**) y doble (**`double`**) precisión, bytes (**`byte`**), enteros cortos (**`short`**), enteros (**`int`**), enteros largos (**`long`**), cadenas de caracteres (**`string`**) y geometrías (**`geometry`**):

```
int myInt = 7;
```

Por otro lado, GGL2 proporciona tipos de datos compuestos, entre los que podemos encontrar las secuencias de elementos de un determinado tipo. Por ejemplo, es posible declarar una secuencia de enteros simplemente precediendo el tipo **`int`** de la palabra clave **`sequenceof`**:

```
sequenceof int myInts = int[] [0, 1, 2, 3];
```

Además, también se encuentra disponible otro tipo de datos compuestos: los elementos. Un elemento es una agrupación de atributos o hijos de cualquier tipo, sea este compuesto o básico:

```
element {
      attribute int id,
      attribute geometry geom
}
```

Por último, cabe destacar que GGL2 también proporciona un mecanismo para declarar estos tipos de datos compuestos con un determinado nombre, de forma que se pueda hacer uso de ellos de manera sencilla:

```
define IntSeq as sequenceof int;
define Elem as element {
      attribute int id,
      attribute geometry geom
      };

IntSeq s;
Elem e;
```

Instrucciones

En lo que respecta a las instrucciones del lenguaje, en primer lugar encontramos instrucciones para la lectura y escritura de conjuntos de datos:

```
read SHP 'myShape.shp' to mySHP;

write SHP mySHP to 'myShapeCopy.shp';
```

Estas instrucciones permiten asociar los contenidos de datos externos a variables dentro del lenguaje. En el ejemplo anterior, los contenidos de un *shapefile* llamado "myShape.shp" son asociados a la variable `mySHP` y mediante el uso de ésta será posible leer y procesar los datos de dicho *shapefile*.

Como puede observarse, para cualquier operación de lectura o escritura ha de especificarse el "lector" o el "escritor" que dirigirá la operación (**SHP** en los ejemplos anteriores). Cabe destacar que es posible añadir nuevos lectores y escritores al lenguaje y que no existe ninguna limitación sobre la estructura de los valores que leen o escriben ni sobre la ubicación de dichos valores. Eso permitirá en un futuro crear lectores para formatos basados en GML, o, más general, en XML, pero también permitirá operar con valores de bases de datos remotas, servicios web, sensores, etc.

También resulta interesante destacar que tanto la sintaxis como la semántica del lenguaje es altamente intuitiva y es a la vez una especificación ejecutable y la documentación de aquello que se está ejecutando.

Además de la lectura y escritura de datos, el usuario dispone de instrucciones para la selección de campos y el filtrado de elementos:

```
mySHP select(mySHP/the_geom);

mySHP filter(mySHP/id == 7);
```

También se dispone de instrucciones con la misma funcionalidad que SQL, tales como agrupaciones de elementos o *joins*:

```
mySHP group by(mySHP/id);

mySHP join myDBF on (mySHP/id == myDBF/ref);
```

Por otro lado, cabe destacar que la instrucción **`show`**, muestra el resultado de la expresión posterior por consola. Además, existe la posibilidad de añadir los resultados como una capa nueva añadiendo al final de la instrucción el formato del resultado obtenido y el nombre de la nueva capa:

```
show 'Hello world!';

show results in gis as SHP 'resultsLayer';
```

También es importante notar que cuando el entorno de trabajo está conectado con el SIG, cada una de las fuentes de datos que se encuentran definidas en el SIG se declaran como variables de cualquier programa GGL2 de forma automática e implícita. De esta manera, el siguiente programa:

```
show layer filter(layer/id == 7) in gis as SHP 'filtered';
```

a pesar de tener una única instrucción y no haber declarado la variable `layer` con anterioridad, es perfectamente correcto, siempre y cuando en el SIG se encuentre definida una capa de nombre "layer".

Otra característica importante del lenguaje es la posibilidad de especificar geometrías vectoriales de manera literal haciendo uso de WKT (*Well Known Text*):

```
geometry myGeometry = POINT(0 0);
```

Por otro lado, existen diversas librerías que proporcionan funciones predefinidas al usuario. Para utilizarlas, únicamente hay que importarlas al inicio del fichero GGL2 mediante la instrucción **`import`**:

```
import ggl.math;
```

Entre las librerías disponibles podemos encontrar una librería para operaciones matemáticas (`ggl.math`), una librería para operar con cadenas de caracteres (`ggl.strlib`) y la más importante, una librería para operar con geometrías que sigue especificación OGC para fenómenos simples [3] para su implementación (`ggl.geom`).

Además, es posible crear nuestras propias librerías escribiendo su nombre cualificado al inicio del fichero e implementando diversos algoritmos de la siguiente manera:

```
library ggl.MyLibrary;

alg getBufferArea(geometry g, double d) returns double {
      return getArea(buffer(g, d));
}
```

Posteriormente, es posible llamar a dichos algoritmos desde cualquier otro programa, importando la librería creada, tal y como se muestra en el siguiente ejemplo:

```
import ggl.MyLibrary;

geometry g = POLYGON((0 0, 0 1, 1 1, 1 0, 0 0));
show getBufferArea(g);
```

Además, también es posible utilizar fuentes de datos jerárquicas, de manera que se accede a todo el árbol de la jerarquía mediante el uso de expresiones Xpath, por lo que la siguiente instrucción sería perfectamente correcta:

```
show city/buildings/hospitals/the_geom;
```

Por último, cabe destacar el hecho de que existen más instrucciones que facilitan el uso del lenguaje y que vienen heredadas de los lenguajes imperativos, entre las que cabe destacar la sentencia condicional:

```
if (condition) {
      ...
} elseif (other_condition) {
      …
} else {
      …
}
```

o el bucle *for*:

```
for (int i = 0; i < mySHP/length; i = i+1) {
      ...
}
```

Entorno de trabajo

GGL2 proporciona un entorno de trabajo que permite la edición, ejecución y conexión con el SIG de manera sencilla. Para ello, se hace uso de la plataforma Eclipse, por lo que muchas de las funcionalidades que proporciona dicha plataforma se heredan en GGL2, como pueden ser la búsqueda de recursos, sustitución de texto en ficheros, manejo de repositorios, sistema de actualizaciones, etc. Además, con respecto a la funcionalidad específica del nuevo lenguaje de programación, caben destacar las siguientes funcionalidades:

1. **Autocompleción**: Es posible completar el código de manera automática en función del contexto de edición, pudiéndose incluso autocompletar el nombre de los campos de una capa con la que se está trabajando desde el mismo momento de haberla leído, tal y como se muestra en la Figura 2.

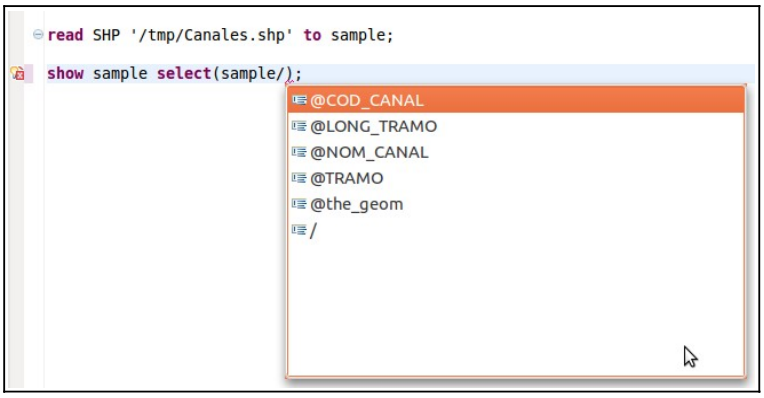

Figura 2: *Autocompleción de los campos de una capa.*

2. **Conexión automática con el Sistema de Información Geográfica**: El entorno de desarrollo conecta de manera automática con el SIG, que actualmente sólo puede ser GearScape. En futuras versiones existirá la posibilidad de conectar con diferentes SIGs y de configurar con cuál se quiere conectar. De cualquier manera, esta conexión se realizará de manera automática sin que el usuario deba de realizar ninguna operación adicional.

3. **Coloración sintáctica**: Como se puede comprobar en la Figura 2, el editor del entorno de trabajo también realiza coloración sintáctica, lo que permite distinguir las palabras clave por su color y, por tanto, la detección temprana de errores tipográficos.

TRABAJO FUTURO

En primer lugar, debido a que GGL2 se encuentra en una fase temprana de evolución, el trabajo futuro admite muchas y diferentes posibilidades. En general, en aquellos casos en los que GGL2 no sea capaz de resolver un determinado problema y exista una instrucción que no sólo resuelva ese problema, sino que de solución también a un amplio conjunto de problemas similares, se añadirá dicha instrucción al lenguaje.

Por otro lado, el lenguaje dispone de la infraestructura necesaria para el manejo de datos jerárquicos. Sin embargo, hasta el momento no existen lectores y escritores que permitan el manejo de dichos datos. Es por esto que entre una de las tareas de trabajo futuro más a corto plazo es la inclusión de nuevos lectores y escritores para XML y, en general, para cualquier formato utilizado ampliamente en el contexto de los SIGs.

Otro desarrollo interesante consiste en ampliar el conjunto de librerías disponibles con el fin de poder abarcar funcionalidades tales como la creación de gráficas, informes, cálculo de rutas, etc.

En lo que respecta al manejo de datos, la inclusión de instrucciones para tratar datos raster se considera uno de los principales objetivos a corto plazo.

Además, como se ha ido comentando a lo largo de todo el documento, también se contempla la posibilidad de, en un futuro cercano, poder conectar con Sistemas de Información Geográfica distintos a GearScape, como puede ser gvSIG, consiguiendo así una mayor interoperabilidad y productividad.

Por último, el diseño de la aplicación, así como el uso de la plataforma Eclipse para el desarrollo permite la transformación de código GGL2 en cualquier otro producto de manera sencilla y con un tiempo de desarrollo relativamente bajo, por lo que a largo plazo se plantea la idea de, además de generar código Java, poder generar código en C (o C++), generar plugins para algún sistema concreto, o incluir los geoprocesos definidos en GGL2 como procesos de un servidor WPS, tal y como ya se hizo con la primera versión de GGL y el servicio WPS de 52º North [5].

CONCLUSIONES

En primer lugar, es importante destacar que se han solucionado de manera eficiente todos los problemas que había planteado el uso de la primera versión de GGL a lo largo de este último año, y que se describen en detalle en la segunda sección de este documento.

Por otro lado, esto se ha hecho desarrollando un nuevo lenguaje específico para SIG que se ajusta a las necesidades del área. Además, debido a la tecnología utilizada para el nuevo lenguaje, la inclusión de nuevas instrucciones que puedan ser necesarias, así como la generación de diferentes productos a partir de una misma especificación del problema, hacen de GGL2 no sólo una herramienta potente, sino también enormemente ampliable y configurable.

Por último, es importante destacar que actualmente el lenguaje se encuentra en una fase de desarrollo bastante temprana y, aún a pesar de disponer de una funcionalidad mínima para poder comenzar el aprendizaje del lenguaje, manejar datos vectoriales e implementar geoprocesos básicos, todavía es pronto para utilizar el lenguaje en un entorno de producción más exigente.

REFERENCIAS

[1] VAN DEURSEN, A.; KLINT, P. y VISSER, J. (2000), “Domain-specific languages: an annotated bibliography”. SIGPLAN Not. 35-6, pp.26-36.

[2] ISO, *Structured Query Language – 92,* International Organization for Standarization, 1992.

[3] OGC, *OpenGIS Implementation Specification for Geographic information – Simple feature access – Part 1: Common architecture*, Open Geospatial Consortium, 2006.

[4] OGC, *OpenGIS Implementation Specification for Geographic information – Simple feature access – Part 2: SQL option*, Open Geospatial Consortium, 2006.

[5] GONZÁLEZ, V.; SCHÄFFER, B; GONZÁLEZ, F. (2009), Publicación y uso de scripts SQL en servidores SQL transaccionales, VI Jornadas Técnicas de la IDE de España, Murcia.