CAPITULO 2

Conceptos Generales y Tecnologías de Soporte

Este capítulo tiene como objetivo presentar y explicar los conceptos relacionados con esta tesis para poder introducirlos por medio de una definición previa. Así mismo se presentarán las tecnologías de soporte para GeoMVisio. Se introducirán y explicarán las funciones que estas realizan para poder cotejar sus ventajas y desventajas en cuestión de las necesidades requeridas.

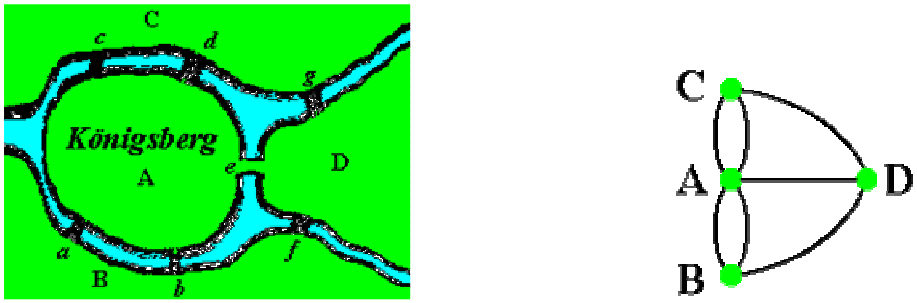
2.1 Conceptos Generales

Como primera parte de este capítulo se explican y muestran los conceptos generales relacionados con GeoMVisio para tener un preámbulo de lo que se estará comentando a lo largo de esta tesis.

2.1.1 Grafos

Un grafo es un conjunto de vértices unidos por arcos que nos permiten representar de una manera más sencilla conceptos que indiquen conectividad. Se pueden presentar por ejemplo mapas, rutas de carreteras, organigramas corporativos o la relación social en un grupo de personas

En la Figura 2.1 podemos ver un grafo que representa un pequeño mapa en donde la tierra separada por el agua es representada en la gráfica por cuatro vértices verdes unidos por arcos que representan la unión que existe entre los diferentes vértices.



Mapa Grafo

Figura 2.1

Representación de un mapa ejemplo con un grafo.

2.1.2 Vértices

Los vértices, conocido también como nodo representan los elementos principales que componen un grafo. A partir de este momento, solo se utilizara el concepto de vértices. Estos elementos indican el propósito de un grafo y pueden representar desde integrantes de una familia hasta estrellas en una galaxia, con un vértice para cada uno de los elementos. Los vértices tienen dos características la adyacencia y el grado de uso.

Dos vértices están adyacentes si están unidos por un arco. En la Figura 2.2 observamos que el vértice *A* es adyacente a los vértices *C* y *B* pero no lo es con el vértice *E*. El grado de uso de los vértices nos indica la cantidad de arcos que se conectan con un vértice; entre más arcos toquen ese vértice más valor tiene dentro del grafo. En la Figura 2.2 vemos que el vértice *B* tiene un grado de tres mientras que los vértices *D* y *F* solo tienen un grado dos.

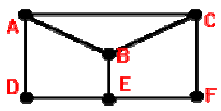


Figura 2.2

Grafo que ejemplifica funciones de los vértices.

2.1.3 Arcos

Los arcos son los elementos que denotan la conectividad y relación entre los vértices. Estos elementos pueden ser dirigidos o no dirigidos; cuando son dirigidos se representan por arcos con puntas que representan la dirección. Cuando no son dirigidos se representan ya sea con arcos sin punta o con punta en los dos lados del arco. Los arcos no precisamente pueden tener una forma recta o una apariencia estética ya que el objetivo primordial es unir dos o más vértices.

El camino que estas forman es la cantidad de arcos que van formando una trayectoria y un ciclo es un arco que termina en el mismo lugar que inicia. Un ejemplo de este tipo de arcos se presenta en la Figura 2.3 donde podemos ver en la imagen que los arcos forman un tipo de elipse ya que se dirigen al punto de origen.

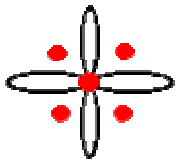


Figura 2.3

Grafo que representa funciones de los arcos.

2.1.4 Hipergrafo

Antes de ver que es un hipergrafo primero tenemos que entender el concepto de subgrafo. Un subgrafo es un grafo dentro de un grafo, es decir, un conjunto de dos o más vértices relacionados por arcos de manera estructurada que en

conjunto representan un vértice dentro del grafo principal, cabe mencionar que os subgrafos a su vez también pueden contener subgrafos.

Ahora, dentro de los elementos de un grafo puede existir un vértice que haga referencia a un subgrafo pero sin mostrar toda la información, este tipo de vértices es un hipergrafo. Los Hipergrafos actúan como las ligas que se manejan en las páginas en Internet, donde al interactuar con estas nos dirigen a alguna dirección de Internet.

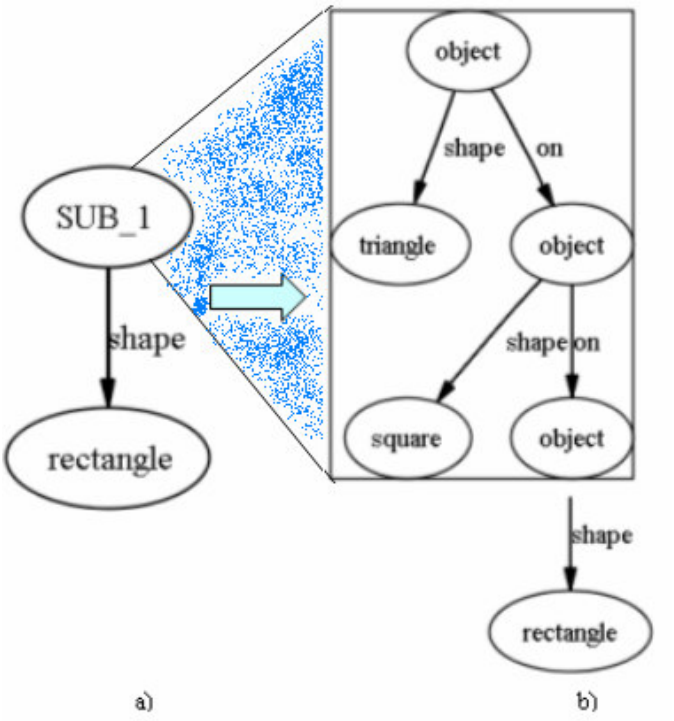


Figura 2.4

Representación de un hipergrafo.

En la imagen de la Figura 2.4a) podemos ver un caso hipergrafo, se observa un vértice con la leyenda de *SUB_1* que hace referencia a otro grafo y al interactuar con este por medio del *Mouse*, expande la grafica de un vértice a un subgrafo que se muestra en índice b de la Figura 2.4. Los dos grafos

mostrados son lo mismo la única diferencia es que ahora se tiene todos los elementos incluyendo el subgrafo.

2.2 Tecnologías de Soporte

Una vez con los conceptos generales ya presentados y explicados se parte a la introducción de las tecnologías que están relacionadas con GeoMVisio, así como aquellas herramientas que son de particular interés para la implementación del software.

2.2.1 Graphviz

Su nombre viene a partir de una conjunción de las palabras en ingles *Graph Visualization* que significa visualización de gráficos. Fue desarrollado en los laboratorios AT&T por John Ellson en los Estados Unidos de Norteamérica [Graphviz, 2005]. Graphviz es un conjunto de herramientas que nos permite representar información estructural por medio de diagramas gráficos o de redes. Una herramienta que pueda dibujar grafos automáticamente es de gran importancia para aplicaciones que dependan de la visualización de grafos que representen algo significativo, han tenido éxito en campos tales como ingeniería de software, base de datos y diseño de redes de Internet, entre otros.

Esta tecnología visualiza los grafos de diferentes maneras. Existen diferentes visualizadores de código libre accesibles en Internet tales como WebDot, Grappa, y Graphviz. También cuenta con un conjunto de interfaces grafos,

herramientas auxiliares y librerías para cumplir con su función. Graphviz usa el lenguaje Dot para la implementación de las imágenes [Graphviz, 2005].

Esta herramienta obtiene su información en formato de texto del lenguaje *Dot*, sin embargo crea diagramas en diferentes formatos de visualización como imágenes *JPG* o *GIF* , *SVG* para páginas de Internet, *Postscript* para PDF (para visualizar en la herramienta Adobe), también soporta *GXL* y *XML*. Esto permite crear programas para modificar y desplegar un grafo de manera variada, así también como para buscar información dentro de éstos.

2.2.1.1 Ejemplos de Salidas de Graphviz.

La diferencia de salidas depende de la función a la que están aplicadas. Para poder ver la diferencia se muestran algunas de estas salidas.

El caso más sencillo de un grafo estructurado es representado por el conocido *"Hello World"* que muestra solo dos vértices unidos por un arco como se muestra en la Figura 3.4.

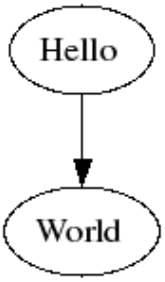


Figura 2.5

Grafo más simple representado por Graphviz.

Otro tipo de salida es mostrada en la Figura 2.6 donde nos muestra dos subgrafos que unidos al grafo un raíz. Este tipo de salida puede mostrarnos visualmente ejemplos como la actividad de dos elementos distintos que estén interactuando al mismo tiempo para un objetivo en común, por ejemplo la ejecución de dos procesos simultáneos. De esta forma podemos ver la actividad de cada uno.

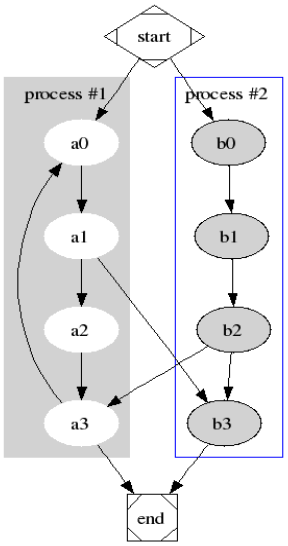


Figura 2.6

Dos procesos representados por medio de Graphviz.

Otro ejemplo es la representación de estructuras de datos. Este tipo de salida nos permite visualizar imágenes estructuradas que simbolizan casos como una base de datos relacional, una red de pozos petroleros o una arquitectura de clases de algún lenguaje de programación. Como podemos ver en la Figura 2.7 las imágenes generadas pueden contener más información que solo una etiqueta que indique lo que representa cada vértice, también contienen datos concretos sobre todas las actividades dentro del grafo por parte de cada vértice.

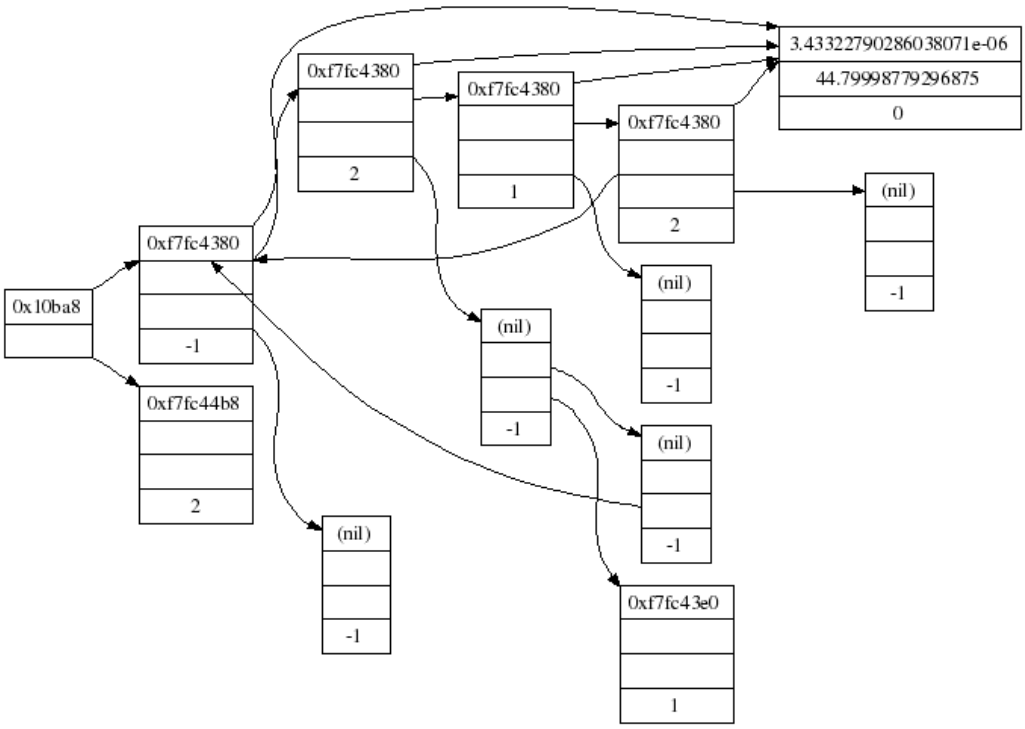


Figura 2.7

Imagen de una estructura de datos representada por Graphviz.

Los ejemplos anteriores solo son algunos de la gran variedad de maneras de visualizar diagramas con Graphviz. Se puede entender la importancia de la representación de información por medio de imágenes al poder ver lo que estos ejemplos representan, por esta razón se justifica la gran variedad de aplicaciones de esta herramienta en diferentes campos.

2.2.2.2 Lenguaje estructurado para grafos Dot

El despliegue final de los grafos por parte de Graphviz es gracias al lenguaje Dot ya que es muy sencillo crear modelos gráficos con éste y los visualizadores de Graphviz pueden procesar estos modelos para generar las imágenes e interactuar con ellas.

Dot es un lenguaje estructurado que describe grafos en formato de texto, es una manera simple e intuitiva para representarlos [Gransner, 2002]. Este lenguaje describe los tres objetos básicos de los grafos: subgrafos, vértices y arcos. Los primeros elementos que se crean siempre son los vértices después de esta acción es posible crear los arcos que conectan los vértices.

El formato de *Dot* representa los vértices empezando con una línea de texto con la palabra *node* de la siguiente manera

node [shape=box];.

Un arco se representa cuando dos vértices son unidos con el operador ->, como se muestra en la siguiente línea:

1983 -> 1985

Como podemos observar se crea un arco que parte del vértice *1983* al vértice *1985* creados con anterioridad. A estos elementos al igual que a los vértices es posible agregarles más características como color, estilo de letra, estilo de fondo, entre otras para poder darle tanto presentación visual, como aportación informativa al grafo final. Un ejemplo del código de un grafo representado por este lenguaje se muestra a continuación:

```
digraph a_log0_f_1_1 {
graph [fontsize=12, compound=true, labelloc=top,
       labeljust=r, label="a_log0_f Iter: 1 Substructure: 1
            Graph(2v,1e) Instancias: 12654"];
node [label="\N", fontsize=12];
edge [fontsize=12];
graph [lp="160,133",
      bb="0,0,320,144"];
1 [label=" ACCESO", pos="133,104", width="1.06", height="0.50"];
2 [label=" FROM", pos="133,18", width="0.86", height="0.50"];
      1 -> 2 [label=" SITIO_REMOTO", pos="e,133,36 133,86 133,74
      133,59 133,46", lp="179,61"];
}
```

Figura 2.8

Grafo representado en formato del lenguaje Dot.

Con el lenguaje Dot podemos agregar los elementos al grafo así como información visual o descriptiva adicional. Para poder visualizar los elementos de los grafos de manera agradable se debe de agregar también posiciones a los a cada elemento [Gransner, 2002]. Debido a que esta tarea es un tanto complicada y tardada Graphviz cuenta con una aplicación llamada *Dot* que coloca las posiciones automáticamente de cada elemento del grafo. Esto hace el trabajo más fácil ya que solo se debe preocupar por crear los elementos sin tener que acomodar los elementos.

2.2.2.1 Aplicación *Dot*

La aplicación *Dot* es una herramienta de Graphviz que se ejecuta desde consola por medio de una línea de comando, una aplicación web o a través de herramienta interfaz de visualización de grafos [Graphviz,2005]. Entre sus características están la generación de grafos estructurados, es decir, crea las posiciones o coordenadas para cada elemento creado con anterioridad. Gracias a esta aplicación es posible generar los diferentes tipos de salidas con que trabaja Graphviz.

La manera en que *Dot* dibuja o crea los conjuntos de grafos es analizando las entradas de grafos y sus atributos desde un archivo de texto (estos archivos están creados con la gramática de Dot mostrado en el Apéndice A) y después genera los diagramas, ya sea en un archivo que almacene los diagramas en diferentes formatos como son GIF, PNG, SVG o PostScript (que se puede convertir a PDF) [Dot, 2005].

El proceso de generación de grafos consta de cuatro fases. Lo primero es romper todos los ciclos dentro del diagrama de entrada, para esto se cambia la dirección de los arcos que representen un ciclo, esto debido a que la salida de *Dot* permite estos casos. El siguiente paso es asignar niveles a los arcos en orden descendente para determinar las coordenadas de "Y" del grafo final. El siguiente paso es ordenar los vértices por niveles para evitar que se encimen. Finalmente en último paso es establecer las coordenadas de los vértices en *X*.

2.2.2.2 Ejecución de la aplicación Dot

Para ejecutar la aplicación Dot se siguen los siguientes pasos.

1. Primero se escribe el comando *Dot* que manda llamar la aplicación.

2. Después se escribe el nombre del archivo escrito con la gramática del lenguaje Dot. Cuenta con varios comandos extras opcionales; uno de ellos genera el formato de salida colocado antes del nombre del archivo en la línea de comando y finalmente el comando para archivo de salida con el comando *–o* y el nombre del archivo final.

Un ejemplo de la estructura de ejecución se muestra en la Figura 2.9 que genera la imagen de la Figura 3.8 en formato de PostScript y se almacena en un archivo llamado *graph1.ps,* el cual puede ser manipulado de diferentes maneras ya sea para su impresión o su conversión a PDF [Gransner, 2002].

```
$ dot -Tps graph1.dot -o graph1.ps
```

Figura 2.9

Línea de comando para ejecutar la aplicación Dot.

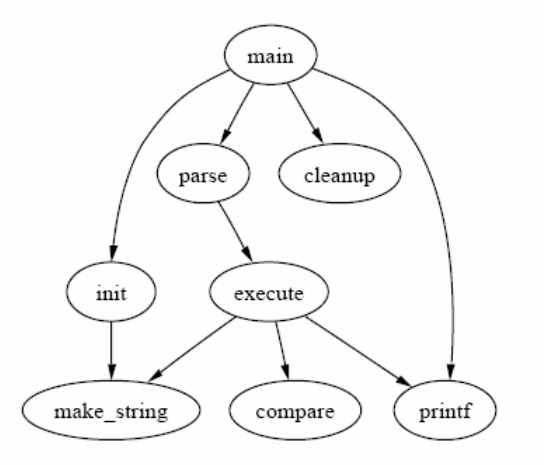


Figura 2.10

Grafo generado por la aplicación Dot

La imagen de la Figura 2.10 es generada a partir de un archivo ejemplo en lenguaje Dot presentado en la Figura 3.9, en la figura podemos ver en la línea numero uno el nombre y el tipo de grafo, las siguientes líneas crean vértices, arcos o subgrafos y le son agregados atributos a cada elemento.

```
1:  digraph G {
2:      main -> parse -> execute;
3:      main -> init;
4:      main -> cleanup;
5:      execute -> make_string;
6:      execute -> printf
7:      init -> make_string;
8:      main -> printf;
9:      execute -> compare;
10: }
```

Figura 2.11

Grafo en formato del leguaje Dot.

Esta aplicación funciona gracias a archivos establecidos con la gramática de Dot, como el caso de la figura anterior. Con el lenguaje Dot podemos agregar mucha mas información a los elementos los cuales la aplicación Dot se dedica a plasmarlos en los diagramas. Cada elemento grafo, arco o subgrafo tienen un conjunto de atributos que los hace mas versátiles. Se pueden hacer los elementos de cierta forma, color, tamaño así como integrarles texto con diferentes tamaños, colores, estilo y textos que indique información relevante extra [Gransner, 2002].

La razón por la cual se uso este elemento es importante ya que el software a implementar no tiene que verse involucrado en la asignación de coordenadas de los diferentes elementos del grafo permitiendo concentrarse en la creación de los grafos, vértices y arcos así como sus atributos.

2.2.3 Visualizadores de Graphviz

Como se mencionó anteriormente, Graphviz consta de un conjunto de visualizadores para visualizar, crear y manipular grafos [Grsphviz, 2005], la variedad de estas herramientas nos permite poder escoger entre varias de ellas la que mejor se acople al objetivo de GeoMVisio. A continuación se presentan las herramientas de visualización que se tomaron en cuenta.

2.2.3.1 Grappa

Hasta ahora hemos comentado las herramientas que nos permiten implementar, organizar y crear los grafos pero no se cuenta con un visualizador. Graphviz cuenta con varios visualizadores, pero Grappa es la herramienta que más se acopla a las necesidades de GeoMVisio ya que esta implementado en java, mismo lenguaje de programación para el desarrollo de este sistema. De hecho, Grappa se considera como una extensión de Graphviz en Java.

Grappa fue un proyecto desarrollado en los laboratorios de AT&T dirigido por el Dr. John Mocenigo en los Estados Unidos de Norteamérica. Grappa obtiene su nombre a partir de la unión de las palabras en ingles *Java Graph Package* que quiere decir Paquete Java para Grafos [Grappa, 2005].

Grappa más que ser un visualizador es un paquete de clases de java que permiten realizar esta tarea, así como la capacidad de manipular los grafos dentro de aplicaciones o *applets* desarrollados en Java. Este paquete cuenta con una gran variedad de utilidades, sin embargo se tiene la posibilidad de modificar las clases. Este conjunto de clases, al igual que Graphviz son de

acceso libre en la red. Para poder obtener una copia de este paquete se adquiere en la página de Graphviz.

La última versión de Grappa se implementó en la versión 1.2 del JDK (*Java Development Kit*) y hace uso de recientes implementaciones de Java tales como *Java 2D* y *Javax Swing* [Grappa, 2005]. Estas nuevas integraciones brindan un gran apoyo para la realización de la tesis ya que tiene una gran cantidad de clases del paquete *Javax Swing* que permiten interactuar al usuario con las imágenes mostradas.

Grappa cuenta con varias características para manipular los grafos, algunas de ellas son las siguientes.

- Cuenta con métodos que permite construir, manipular y desplegar los elementos vértices, arcos o subgrafos.
- Cada grafo contiene una cantidad de atributos que incumben tanto al nombre como a sus características.
- Grappa incluye métodos para leer y dibujar grafos realizados en formato de texto Dot.
- La interacción que ofrece Grappa permite la creación y eliminación de elementos del grafo según la necesidad del programador por medios como métodos internos de clases de la paquetería o por interacción con el mouse.
- Cuenta con una gran variedad de formas para la creación de vértices o arcos.

- Esta herramienta permite desplegar un grafo en diferentes ventanas al mismo tiempo sin presentar problemas.
- Permite agregar una etiqueta que aparezca al momento de interactuar con un elemento de alguna manera para poder agregar algún tipo de información.

Estas son solo algunas de las funciones que tiene Grappa, sin embargo cuenta con una variedad más amplia de utilidades que ayudarán a que la implementación del software sea más sencilla, en especial para la creación de los hipergrafos.

2.2.3.2 Web –Dot

Web-Dot es un programa CGI (Common Gateway Interface que es programa dinámico que recibe e interprete datos enviados a través de un servidor Web por medio de Internet) desarrollado por John Ellson que convierte el grafo en un archivo .Dot a una imagen que puede ser incluida a una pagina Web [Web-Dot, 2005].

Web-Dot esta escrito en Tcl (que viene del ingles Tool Command Language, es un lenguaje de programación interpretado y multiplataforma que permite una gran facilidad de implementar funciones en C/C++) usando el paquete TclDot de Graphviz

Para poder ejecutar Web-Dot se necesita de graphviz-tcl y un servidor Web como Apache. Una aplicación interesante de esta herramienta es que permite

crear los grafos en formato *svg* o *svgz* (Scalable Vector Graphics es un lenguaje basado en XML para describir gráficos vectoriales bidimensionales estáticos o animados) y en formato *pdf*. [Web-Dot, 2005].

Esta herramienta proporciona varias características interesantes en aplicaciones Web, desde una simple presentación de un grafo sencillo, elementos del grafo interactivos, hasta características propias del svg.

2.2.3.1 Como usar Web-Dot

Para explicar el uso de Web-Dot se presentan algunas funciones que pueden realizarse a los grafos, para cada caso siempre debe de existir un archivo en formato Dot [Web-Dot, 2005] creado con anterioridad que contenga las características del grafo a mostrar. Así también se requiere de conocimientos del lenguaje Html (Hyper Text Markup Language) el cual es un lenguaje de marcas diseñado para estructurar textos y presentarlo en hipertexto el cual es el formato estándar de las páginas Web [w3, 2005].

Para comenzar se muestra el caso más simple, un grafo normal descrito en un archivo en formato .Dot el cual pueda ser integrado en un servidor para su distribución por la red. Para que pueda ser agregado a una página Web se necesita la siguiente línea en un archivo en lenguaje Html:

```
<img src="/cgi-bin/webDot/ webDot/basic.Dot .neato.png">
```

Figura 2.12

Marca de html para integrar webDot a un documento web.

El URL de esta marca de html img esta estructurada de la siguiente manera:

- cgi-bin/webDot : es el servidor CGI de Web-Dot.
- webDot/basic.Dot: es el archivo del grafo con el que se va a trabajar.
- neato: Es la máquina de despliegue de datos de graphviz. Puede ser ésta o la aplicación Dot con la que trabajamos en esta tesis.
- .png: Es el formato de salida, es decir el formato en el que será almacenado el grafo.

En el caso de que una persona quiera observar un grafo sin tener que instalar Web-Dot en su maquina, es posible llevarlo a cabo. Es posible visualizar el grafo de muestra desde cualquier navegador de Web por medio de la dirección Web “http://www.graphviz.org/demo.Dot”, la cual es la dirección de una página de un servidor que cuenta con Web-Dot. Para este caso se colocan la siguiente línea de hipertexto:

```
<img src="http://webDot.graphviz.org/cgi-bin/webDot
          /http://www.graphviz.org/demo.Dot.Dot.png"
      alt="graph from public webDot server">
```

Figura 2.12

Marca de Html para integrar webDot a un documento web.

Siguiendo la misma estructura redirige a una dirección web donde nos muestra un archivo ejemplo Dot usando la maquina de despliegue de grafos Dot de Graphviz en un formato de salida de imagen.

Cuando se requiere que un elemento de algún grafo sea interactivo, es decir, algún vértice o arco, primero debe de ser implementado por medio del lenguaje Dot en las especificaciones del grafo contenidos en el archivo .Dot de prueba como se muestra a continuación en la siguiente imagen:

```
digraph G {
        graph [URL="default.html"]
        a [URL="a.html"]
        b [URL="b.html"]
        c [URL="c.html"]
        a -> b -> c
        a -> c
}
```

Figura 2.14

Integración de un URL por medio del formato del lenguaje Dot.

Como podemos observar en la imagen anterior por medio del lenguaje Dot una de las características que se le pueden asignar a los vértices o arcos es el URL que permite al vértice actuar como una liga redirigiendo al usuario a la página que esta refiere. Sin embargo esta información sola no realiza nada, Web-Dot debe de activar esta acción de con la siguiente línea:

```
<a href="/cgi-bin/webdot/webdot/hnodes.dot.dot.map">
<img src="/cgi-bin/webdot/webdot/hnodes.dot.dot.png" ismap>
</a>
```

Figura 2.15

Marca de Html para integrar webDot a un documento web.

Para poder realizar esta acción se tiene que crear un elemento *map* del lenguaje de hipertexto (elemento que permite crear regiones en una imagen y les asigna acciones especificas). Web-Dot soporta map-server y map-client.

Como se comentó Web-Dot es capaz de mostrar las imágenes en formato *svg* el cual presenta varias ventajas propias de este lenguaje. Para poder trabajar con este formato Web-Dot hace uso de una marca de hipertexto para poder crearlo:

```
<webdot src="/cgi-bin/webdot/webdot/svginline.dot" engine="neato" type="svg">
```

Figura 2.16

Marca de Html para integrar webDot a un documento web.

Como podemos ver en la Figura 2.16 el parámetro que restringe el formato de salida ahora es “type” que en este caso la genera en formato svg.

Estos son solo algunos ejemplos de lo que puede hacer Web-Dot. Existe aún un variado número de aplicaciones para los grafos [Web-Dot, 2005]. Como podemos ver esta herramienta es bastante interesante, podemos trabajar con grafos y distribuirlos a través del Internet de manera sencilla en diferentes formatos. Esto puede ser de gran interés si en un futuro se desea transportar al

Internet las contribuciones de GeoMVisio y de esta manera el usuario no necesariamente tendría que instalar en su maquina de trabajo el software diseñado para esta tesis.

En este capítulo se presentaron los términos con los que se trabajara a través de este documento. Así también se presentaron las tecnologías que se pretende usar en GeoMVisio y también se presentaron algunos visualizadores desarrollados especialmente para trabajar con Graphviz para poder ver los diferentes campos en los que se puede desarrollar esta herramienta.