vgraph

Victor Volle

28. Feb. 2010

1 Introduction

Did you ever need to create a Visio (TM) document from some graph model and keep the Visio document in synch with the model? Even if you completely changed the layout? *vgraph* has been created to achieve that.

vgraph creates a Visio document with a very simplistic layout: it just distributes the nodes evenly on the page. I haven't yet found a layout algorithm that would really do what I want, so it is up to the user to do the layout. *vgraph* just helps you to keep your Visio diagram and your model in synch. The workflow is shown as depicted in the image below:

1. *vgraph* is used to create an initial Visio document from the first version of the model. In the example, the diagram contains four nodes (A, B, C, and D) and a single connection between B and C.

2. Someone then manually changes the layout, moving the nodes around, adding some colour. (Perhaps even adding some additional shapes)

3. Now the model has changed (a connection from A to B is added and node D has been removed). *vgraph* is used to merge these changes into the Visio diagram. All manual changes are kept, only new nodes and connections are added and deleted elements are marked.

2 Using *vgraph*

vgraph is a simple command line utility which only needs a Java (TM) runtime environment.

If you simply execute

```
java -jar vgraph-one-jar.jar
```

on the command line you should get a helpful description like this:

```
Usage: java -jar vgraph.jar [OPTIONS] <source> <target>
OPTIONS:
 -c (--conf) <configuration string> : a configuration for the source loader
                                      (depends on the loader implementation)
 -f (--force)                       : overwrite the target instead of merging
```

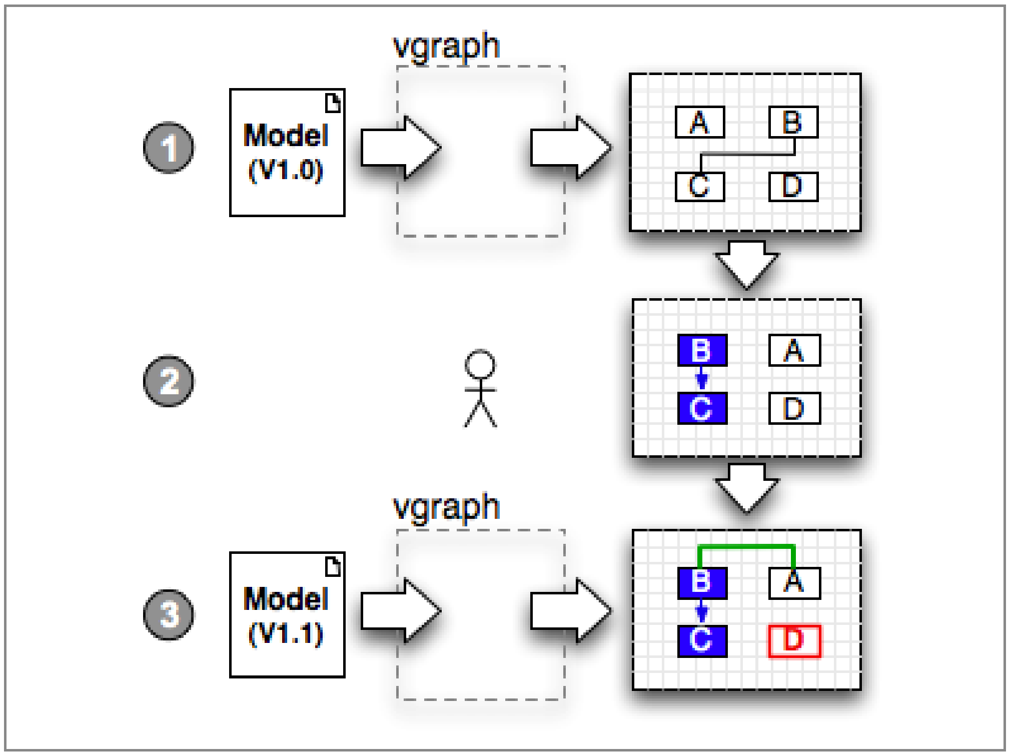

Figure 1: Workflow

```
-h (--help)                          : display help
-l (--loader) <loader>               : the class of the source loader
-t (--template) <template>           : the Visio (TM) template to be used
-v (--version)                       : display the version
```

So you have at least to set the 'source' and the 'target', where the source is your model and the target ist the name of the Visio document you want to create or merge into. But you must tell *vgraph* how to load your model. Currently only models in XML are supported and you must provide some XPath expressions that allow *vgrah* to extract the necessary information. Let's look at a simple example:

```
<graph>
    <systems>
        <system id="A">Node A</system>
        <system id="B">Node B</system>
        <system id="C">Node C</system>
        <system id="D">Node D</system>
    </systems>
    <connections>
```

```
        <connect id="cAB" source="B" target="C"/>
    </connections>
</graph>
```

Here the nodes are called 'system' and the edges of the graph are called 'connect'. The corresponding property file would look like:

```
nodes=/graph/systems/system
edges=/graph/connections/connect

node.extId=@id
node.text=.

edge.extId=@id
edge.source.extId=@source
edge.target.extId=@target
edge.text=.
```

The properties 'nodes' and 'edges' should reference all nodes and edges in your graph[1]. These XPath expressions must be absolute, i.e. start from the root node of the XML document. In the example `/graph/systems/system` references all `system` elements. *vgraph* then iterates over all these nodes and extracts the 'id' and the 'text' of the node (using the XPath expressions 'node.extId' (`@Id`) and 'node.text (`.`)). The 'id' must be unique for the whole graph (neither another node nor a connection may have the same 'id'). Because it is this 'id' that allows *vgraph* to find existing diagram elements, when it merges changes from the model into an existing Visio document. The edges are handled the same way, but since they connect two nodes, these nodes must be given as well (the properties 'edge.source.extId' and 'edge.target.extId' are used to extract that information). The XML example above contains only a single connection between the node with the 'id' B and the node with 'id' C.

If the properties are in a file called 'myXPath.properties' and your model is in a file called 'myModel.xml', you could execute *vgraph* with:

```
java -jar vgraph-one-jar.jar -c myXPath.properties myModel.xml visio.vdx
```

If the simple XPath extraction mechanism does not suffice, you can even write your own loader, which only have to implement a very simple Java interface:

```
public interface GraphLoader {
  public void load(String source, String configuration, Graph graph);
}
```

where the 'source' may be a file name (or a database table, or a URL, etc.) and the configuration might be anything. The latter is just the String given as command line parameter '–conf'. The loader must add all nodes and edges to the 'graph' object given as

[1]If you cannot extract all graph elements with a single XPath expression or if you have multiple source models, you can simply run *vgraph* multiple times and merge everything.

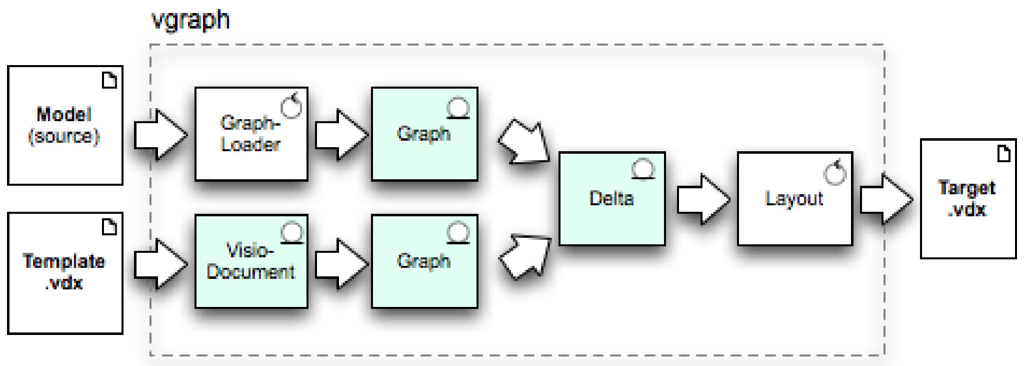

Figure 2: Processing

last parameter. The class `SimpleXmlSourceLoader` is a good starting point, if you need to create your own loader.

3 Internals

Internally *vgraph* simply loads the source model into a Graph object. It then loads the content of the Visio document into another Graph object and compares these two. Any graph element from the model that is not already in the Visio-'graph' is added. *vgraph* then looks for any Visio shape that has a custom property named 'external_id', but whose value could not be found in the source model. These shapes will then be put on a Visio layer called "Deleted" (if the layer does not exist, it will be created). All elements on this layer are coloured red, so they should be rather visible. If you have coloured some elements red yourself, you could just make all layers besides the "Deleted" layer invisible.

If you do a merge, all new elements will be put on a layer "New" as well (all elements on that layer are coloured green). You should then manually move these elements onto another layer (or simply remove it from the "New" layer).