SVG Open 2005

XML to SVG Transformation Mechanisms Comparison
The GraphML use case

cjolif@ilog.fr

XML to SVG Transformation Mechanisms

GraphML use case

- GraphML and SVG in a nutshell
- GraphML to SVG using XSLT
- GraphML to SVG using Java Transformation
- GraphML in SVG thanks to sXBL

GraphML (1/2)

- XML based file format for graphs (set of nodes connected by edges)

```
<graph edgedefault=”directed”>
  <desc>GraphML sample</desc>
  <node id=”root”/>
  <node id=”n1”/>
  <node id=”n2”/>
  <node id=”n11”/>
  <edge source=”root” target=”n1”/>
  <edge source=”root” target=”n2” directed=”false”/>
  <edge source=”n1” target=”n11”/>
</graph>
```

GraphML and SVG in a nutshell

GraphML (2/2)

- Can represent direct or undirect, hyper and nested graphs

```
<graph>
  <graph>
  </graph>
</graph>
```

- Can be extended to add additional data to the nodes and edges

```
<node id=”n1”>
  <data key=”color”>green</data>
</node>
```

GraphML and SVG in a nutshell

SVG

- Scalable Vector Graphics XML based format
- SVG can be used to represent a GraphML graph: graph.svg
- There are three main tasks to achieve to transform GraphML to SVG:
  - Transform node elements to g, rect & text, and edge elements to polyline elements
  - Position the created SVG elements
  - Apply a drawing style to the elements

XML to SVG Transformation Mechanisms

GraphML use case

- GraphML and SVG in a nutshell
- GraphML to SVG using XSLT
- GraphML to SVG using Java Transformation
- GraphML in SVG thanks to sXBL

GraphML to SVG using XSLT

Portable XSLT (1/3)

- We only experimented translating GraphML to SVG for the description of a single rooted tree. This already requires quite a big XSLT stylesheet for a limited result: graphml2svg.xslt
- We have to first iterate (using recursion) over the nodes to determine the root of the tree
- Once done, we then recurse over the tree (nodes and links) to create their SVG counterparts and layout them

Portable XSLT (2/3)

- Advantages:
  - Natural choice when translating an XML format to another one
  - Fits very well the first part of the transformation process
  - Can leverage CSS style to allow changing the style without modifying the XSLT stylesheet: default.css
  - Can be applied either on the client or on the server

Portable XSLT (3/3)

- XSLT limitations:
  - No iteration
  - No updateable variables
  - Lack of advanced mathematical library
  - The two last points lead to obscure code like this one:`<xsl:variable name="abs_x"select="2*(number($x > 0) -0.5)*$x"/>` to compute the absolute value of x
- These limitations makes the second task (nodes and edges positionning) difficult

GraphML to SVG using XSLT

Extended XSLT

- Using the ability of XSLT to be extended permits to overcome some of the XSLT limitations by building an external functions library which would reduce the stylesheet complexity
- However the resulting stylesheet is not anymore portable
- graphml2svg-ext.xslt

XML to SVG Transformation Mechanisms

GraphML use case

- GraphML and SVG in a nutshell
- GraphML to SVG using XSLT
- GraphML to SVG using Java Transformation
- GraphML in SVG thanks to sXBL

GraphML to SVG using Java Transformation

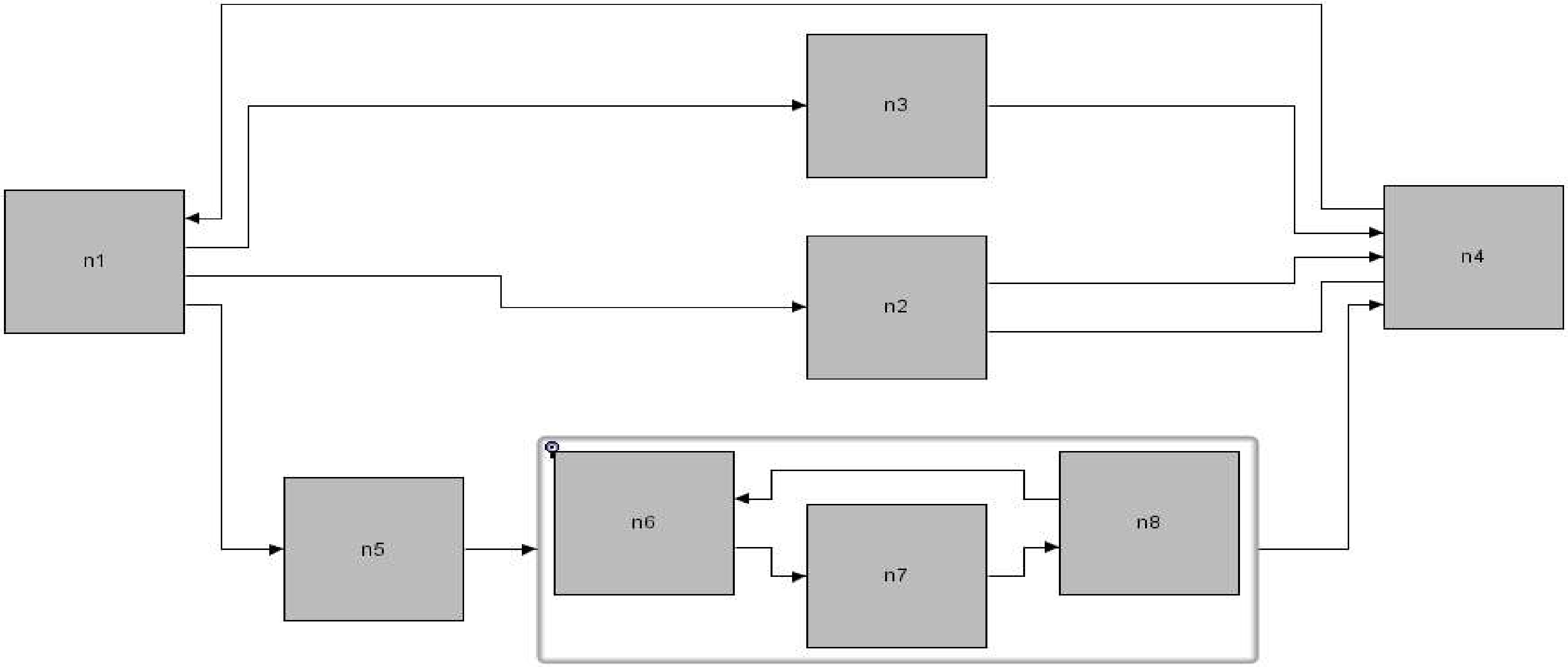
Leveraging ILOG Jviews Diagrammer

- Among ILOG JViews Component Diagrammer packages the following will be used:
  - The SDM (Stylable Data Mapper) package creates graphical representation from data models and a set of stylesheets
  - The Graph Layout package provides a set of algorithms that are able to layout nodes and/or edges of a graph and will be used internally by SDM
  - The Graphics Framework brings to SDM the ability to export its result to SVG

GraphML to SVG using Java Transformation

Advantages

- Allow to layout more complex graphs



- Allow to better parametrize the layout process

XML to SVG Transformation Mechanisms

GraphML use case

- GraphML and SVG in a nutshell
- GraphML to SVG using XSLT
- GraphML to SVG using Java Transformation
- GraphML in SVG thanks to sXBL

GraphML in SVG thanks to sXBL

Introduction

- SVG allows third party namespaces elements into its contents:

```
<svg
 xmlns:graphml=”http://graphml.graphdrawing.org/xmlns”>
   <graphml:graph edgedefault=”directed”>
      <graphml:desc>GraphML sample</graphml:desc>
      <graphml:node id=”root”/>
      <graphml:node id=”n1”/>
      <graphml:edge source=”root” dest=”n1”/>
   </graphml:graph>
 </svg>
```

- How to visualize it? sXBL (W3C Working Draft)

GraphML in SVG thanks to sXBL

SVG's XML Binding Language (1/2)

- Allows SVG user agents to automatically recognize elements in a third-party namespace and perform a transformation of these elements into SVG elements for rendering
- To each third-party element corresponds a definition element with a template sub-element which will be cloned an put into a shadow tree to be rendered by the SVG user agent
- The component definition receives events and can react to them by modifying the shadow tree

GraphML in SVG thanks to sXBL

SVG's XML Binding Language (2/2)

```
<xbl:definition element="grapml:graph">
  <xbl:template>
    <g><xbl:content/></g>
  </xbl:template>
  <xbl:handlerGroup>
    <handler ev:event="xbl:prebind" type="text/ecmascript">
       InitGraph(evt.target)
    </handler>
  </xbl:handlerGroup>
</xbl:definition>
<xbl:definition element="graphml:node">
  <xbl:template>
    <g class="node"><rect width="100" height="100"/><text/></g>
  </xbl:template>
  <xbl:handlerGroup>
    <handler ev:event="prebind" type="text/ecmascript">
   var text = evt.xblShadowTree.getElementsByTagNameNS(SVG_NS, "text").item(0)
   var label = document.createTextNode(evt.target.getAttributeNS(null, "id"))
   text.appendChild(label)
    </handler>
</...>
```

GraphML in SVG thanks to sXBL

Pros & Cons

- Pros:
  - Dynamic Transformation
  - Flexibility
  - Interoparable Component Model
  - Portable, Standart
- Cons:
  - Can't leverage other libraries than ECMAScript ones without breaking interoperability as SVG mandates only ECMAScript as supported language
  - No predefine integration with server side components

Conclusion

- The transformation process can follow a lot of different paths
- Developper has to choose among those paths depending on:
  - How far from the XSLT paradigms the required algorithm is?
  - Does he need a fully dynamic transformation?
  - Does he want to leverage existing libraries?
- The different alternatives can be mixed

Conclusion

References & QA

- http://jviews.ilog.com
- http://www.w3.org/TR/sXBL
- http://www.w3.org/TR/SVG11
- http://www.w3.org/TR/SVG12
- http://www.w3.org/TR/xslt
- http://www.w3.org/TR/xpath
- http://graphml.graphdrawing.org/primer/graphml-primer.html