

gViz – A Reference Model and Application of XML for Visualization on the Grid

David Duce and Musbah Sagar

Department of Computing, Oxford Brookes University, Wheatley Campus Oxford OX33 1HX

Abstract

Shalf and Bethel in a recent special issue of IEEE *Computer Graphics and Applications* (25(2), pp.6-9, 2003) write: “the promise of Grid computing, particularly Grid-enabled visualization, is a transparent, interconnected fabric to link data sources, computing (visualization) resources, and users into widely distributed virtual organizations”. This paper explores one theme within the gViz project, the potential of XML for visualization in Grid computing. A model for Grid-enabled distributed collaborative visualization and an XML language to describe a visualization application are presented.

1. Introduction

Visualization is a key component for understanding large-scale simulations and observations. The gViz project arose from the observation that the move to a Grid model of distributed computing and data management called for a review of the effectiveness of current visualization systems in a Grid context. The modern era of visualization stems from the influential NSF Report “Visualization in Scientific Computing” [McCormick:87]. Development of visualization systems was stimulated by this and a number of general purpose systems, notably the family of modular visualization environments (based on a dataflow paradigm) emerged. A number of these are in common use nowadays, including IRIS Explorer, IBM Data Explorer (now OpenDX) and AVS. Other systems are aimed at specific applications or computing environments such as pV3 (for distributed memory parallel computing environments).

In many visualization applications, particularly those involving simulation, the computational requirement is significant. Many visualization systems therefore allow part of the processing to be done remotely. Computational modelling based on simulation followed by large scale data analysis aided by visualization is often involved. Typically though the mechanisms for supporting this are proprietary and insecure. Much modern scientific research requires the collaborative efforts of teams of scientists with complementary skills. Thus visualization systems which include support for multi-user operation have emerged. The combination of these two areas: distributed visualization and collaborative visualization is known as *distributed collaborative visualization (DCV)*.

The aim of distributed collaborative visualization is to harness the processing power of many humans and many computers and stems from the observation that visualization is a medium of communication between members of geographically distributed groups of people each making their individual contribution to some joint endeavour. Such groups may be, for example, formally constituted project teams or informal and transient gatherings of experts. Distributed collaborative visualization calls for collaboration at the system level to enable collaboration at the human level.

There is a growing interest in the use of DCV applications within an Access Grid, for example [Brodlie:02] and a demonstration of a range of systems that took place in May 2003 across at least 14 UK Access Grid sites led by Lakshmi Sastry at Rutherford Appleton Laboratory.

For a detailed review of the state of the art in distributed and collaborative visualization see [Brodlie:03].

In this paper we describe a layered model for distributed collaborative visualization and work in progress to develop an XML application (language) to capture a description of a distributed collaborative visualization application and its resource requirements.

2. The 3-Tier Model

A layered model of distributed collaborative visualization has emerged through thinking about the organization of the human participants in a distributed collaborative visualization session and the organization of the visualization software that supports the activity. The classical reference model for visualization systems is the model of Haber and McNabb [Haber:90] which describes visualization in terms of the sequential composition of three types of processes which are now referred to as

Filter, Map and Render (using later terminology due to Upson) with extensions to collaborative visualization due to Wood *et al.* [Wood:97].

The gViz model arises from a number of observations.

- Different participants within a DCV session may have different roles and hence there may be different tasks that each may need to perform.
- In principle at least, different participants may use different software.
- A DCV session is certainly likely to be heterogeneous with respect to the physical computing environment: display capability, processing capability, networking connectivity.
- Different parts of the overall computation may impose different resource constraints, for example for security reasons it might be necessary to co-locate a data source and Filter process and deny general access to anything other than the filtered data.

The gViz model is described in terms of three layers:

- The *conceptual layer* which describes the visualization (i.e. the transformation from data to image) to be performed, independently of the visualization software and physical resources with which it is to be realized.
- The *logical layer* in which the visualization is expressed as a particular configuration of software entities, but independently of the physical resources with which the configuration will be realized.
- The *physical layer* in which software entities are associated with physical resources.

The idea of the conceptual layer is to abstract away the details of the visualization software and physical resources. It captures the participants' perspective: the role(s) and tasks that the participants are to perform, the nature of the data sources, the visualization itself, and the control and viewing environment. The data source might be a simulation or data repository, the visualization might be an isosurface of a data field from a simulation overlaid with a corresponding isosurface from experimental data, the control and viewing environment might be a remote seminar environment in

which the speaker alone can control the isosurfaces displayed, but individual listeners have local control of the view.

The logical layer then introduces the software entities with which the conceptual description is to be realized. It is useful to consider this in two parts, the *logical visualization design* and the *core software*. If a procedure library is used to realize the visualization, the logical design is essentially the structure of the user program and the core software is the procedure library. If a modular visualization environment is used the logical design is the composition of module network realizing the design and the core software is the set of modules provided by the environment. The logical layer can thus be thought of as a binding of the conceptual visualization design to a software architecture.

The logical layer also includes constraints on the resources required by the logical visualization design, for example, particular processor characteristics, operating environment (display resources etc.), quality of service for network links and requirements for co-location of components to ensure performance and other criteria are met. There is the possibility that resource constraints are not static and will evolve dynamically as a collaborative visualization session progresses. New data sources might be brought into play requiring computation to migrate to satisfy performance criteria or a participant might switch from a desktop to a mobile device.

The physical layer is a binding of the logical visualization design, data sources and core software to particular physical resources. The binding includes allocation to particular processing, networking and display devices. Again there is the possibility that the binding is not static and may evolve as the logical layer evolves.

3. skML

We are developing an XML application (language) to capture a description of the visualization application, or transformation, that the visualization system performs and a proof-of-concept implementation for use with NAG's IRIS Explorer visualization system. IRIS Explorer already has a collaborative visualization capability and extensions to support remote visualization and computational steering in a Grid environment are being developed by other partners in the gViz project [Wood:03]. As part of the gViz project NAG is

contributing IRIS Explorer licences to UK e-Science projects. For these reasons, IRIS Explorer is an appropriate vehicle for experimentation.

A visualization application is constructed in IRIS Explorer by assembling modules into a dataflow pipeline using a visual editor. Such an assembly of modules is termed a *map*. IRIS Explorer contains a library of visualization modules. In addition to the collection of modules provided with the system developers may add their own modules and there is an international repository of public modules. IRIS Explorer enables maps to be saved and loaded from external files. The system also provides a scripting interface called *skm script*. A *skm* script is essentially an audit trail of the operations by which a map can be created or modified. The *skm* language evolved from a language called *Scheme* which is a dialect of Lisp. The basis of our current work is an XML application (language), skML, which can be translated into *skm* script.

The basic components of a map are modules and links. These are represented in skML by the module and link elements, for example:

```
<?xml version="1.0"?>
<skml>
<map>
<link>
  <module name="ReadLat"
    style="left:20;top:170"
    out-port="Output">
    <param name="Filename">
      testVol.lat
    </param>
  </module>
<module id="iso"
  name="IsosurfaceLat"
  style="left:220;top:120"
  in-port="Input">
  <param name="Threshold"
    min="0" max="27">
    1.8</param>
  </module>
</link>

<link>
  <module id="met" name="Metal"
    style="left:420;top:220;
      controlPanel:show
      (100,400,400,300)"
    out-port="Output">
    <param ...>...</param>
  </module>
```

```
<module name="Render"
  style="..." in-port="Input" />
</link>
<link>
  <module ref="met"
    in-port="Input" />
  <module ref="iso"
    out-port="Surface" />
</link>
</map>
</skml>
```

The parameter element is used to represent a module's initial parameter values. The position of a module within the visual editor's display space and the position and size of the module's control panel on the display when the module is launched are regarded as style and are controlled through the style attribute as shown above.

We allow a skML document to contain more than one map. Roles at the conceptual level can thus be represented as different maps within the same skML document. The map element may be decorated with a role attribute to indicate the role with which the map is associated.

Constraints on resources may be described and associated with skML module and link elements using RDF (Resource Description Framework). The idea is to associate RDF descriptions with elements in the skML document, for example the visualization system to which a module belongs, constraints on the physical location of the resource.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/
    1999/02/22-rdf-syntax-ns#"
  xmlns:v="http://www.gviz.org/
    skML/">
  <rdf:Description about="RImg">
    <v:Type>IrisExplorer</v:Type >
    <v:PhysicalLocation
      rdf:resource="http://
        www.gviz.org/Mars101" />
  </rdf:Description>
</rdf:RDF>
```

The expression of resource constraints requires a common vocabulary for describing resources and a language for expressing constraints. It is not the purpose of the gViz project to develop new resource description vocabularies or constraint expression languages, we posit the existence of open standards in these areas from other work. For experimental purposes the GLUE (Grid Laboratory Open Environment) schema and vocabulary is being used [GLUE].

This schema provides a vocabulary for describing computing, storage and network elements. GLUE does not encompass display capabilities, which are important for visualization applications. The Composite Capability/ Preference Profiles CC/PP [CC/PP] vocabulary being developed by W3C goes some way towards providing a vocabulary for display environments, but more work is likely to be needed in this area.

For constraint specification a simple constraint language based on Globus RSL [RSL] but expressed in XML is being used.

At the time of writing an IRIS Explorer module has been written that will take a skML document and launch a selected map within the document. Functionality has also been provided to generate skML documents from maps in the IRIS Explorer map editor. The idea is that a skML document will represent the visualization required within a DCV session and each user may choose from this skML document the IRIS Explorer map corresponding to their role in the session, recognising that different roles may be chosen as the session progresses. Implementation and demonstration of the resource description aspect of skML is in hand.

4. Related Work

The ICENI middleware [Mayer:02] has an underlying flow-based programming model. Applications are constructed by composing components. The middleware uses application related meta data to support scheduling and the generation of component bindings. The meta data is represented as XML annotations and is based on a separation of concerns between meaning, behaviour and implementation. The 3-tier model presented here emphasises different concerns (user role and task, resource constraints).

5. Future Directions

The use of XML annotations of the kind described here is extensible. One potentially interesting direction is to add more semantic annotation, indicating the function of modules in a network in some way, which would open up possibilities of instantiating a network for a different core software package or interfacing to different core software.

Acknowledgements

Discussions with other partners in the gViz project: the Universities of Leeds and Oxford, IBM UK Ltd, NAG Ltd, Rutherford Appleton Laboratory and Streamline Computing Ltd are gratefully acknowledged, as is financial support from EPSRC.

References

[Brodli:02] Brodli, K.W., Wood, J.D., Boyd, D.S., Sastry, L., Gallop, J.R., Osland, C., Bunn, S.E., Collaborative Visualization Using Access Grid, 2002. http://www.comp.leeds.ac.uk/kwb/all_hands/BOF.pdf

[Brodli:03] Brodli, K.W., Duce, D.A., Gallop, J.R., Walton, J.P.R.B. and Wood, J.D., Distributed and Collaborative Visualization, Eurographics 2003 conference, State of the Art Report, 2003.

[CC/PP] CC/PP web link, visited 27 July 2003. <http://www.w3.org/TR/CCPP-struct-vocab/>

[GLUE] GLUE weblink, visited 27 July 2003. <http://www.globus.org/mds/glueschemalink.html>

[Haber:90] Haber, R.B. and McNabb, D.A., Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In B. Shriver, G.M. Neilson and L.J. Rosenblum, editors, *Visualization In Scientific Computing*, pages 74-93, IEEE Computer Science Press, 1990.

[McCormick:87] McCormick, B.H., DeFanti, T.A., Brown, M.D., Visualization in Scientific Computing, *Computer Graphics*, 21(6), 1987.

[Mayer:02] Mayer, A., McGough, S., Gulamali, M., Young, L., Stanton, J. Newhouse, S., and Darlington, J., Meaning and Behaviour of Grid Oriented Components. In *3rd International Workshop on Grid Computing, Grid 2002*, volume 2536 of *Lecture Notes in Computer Science*, Baltimore, USA, Nov. 2002

[RSL] The Globus Resource Specification Language (RSL), web site visited 27 July 2003. http://www.globus.org/gram/rsl_spec1.html

[Wood:97] Wood, J.D., Wright, H. and Brodli, K.W., Collaborative Visualization. In R. Yael and H. Hagen, editors, *Proceedings of IEEE Visualization '97*, pages 253-259, 1997.

[Wood:03] Wood, J.D. *et al.*, gViz – Visualization and Steering on the Grid, EPSRC All Hands Meeting, September 2003.