

Position Statement on Remote and Distributed Visualization Frameworks

(For Spring 2003 Workshop on Visualization Frameworks, April 14-15, 2003.)

Jim Kohl ~ ORNL

My expertise on this topic stems from crossover research between visualization and high-performance computing, including scientific software component technology (CCA), so I will emphasize the challenges relating to these areas in my position statement.

Given this context, the greatest challenges to constructing a unified visualization framework, specifically amenable to remote and distributed scenarios, are as follows:

- Common Interfaces for Visualization Functions
 - Scientific Application Integration
 - Common Distributed Data Model
 - Generalized Parallel Data Redistribution (“MxN”)
- Flexibility and Minimalism of Encapsulating Framework
- Practical Concerns
 - Packaging, Deployment and Support...

Common Interfaces for Visualization Functions

To collaborate in the development of next-generation systems for remote and distributed visualization, we must adopt a common foundation for developing software modules and integrating them into unified solutions. This will enable the merging of expertise from a variety of research areas, allowing each organization to share their software tools and technology, interoperating with the rest of the community.

The SciDAC “Center for Component Technology for Terascale Simulation Software” (CCTSS), otherwise known as the “Common Component Architecture” (CCA), is a grass roots effort to apply business component models (CORBA, Enterprise Java Beans, DCOM) to high-performance scientific software. Engaged since 1998 as an open “CCA Forum”, it has developed a fundamental specification for scientific software components and the frameworks that encapsulate them.

CCA is not a “silver bullet” for magically wrapping up scientific code as components, thereby providing instant plug-and-play interoperability and universal software re-use. CCA only provides the *mechanism* for designing such interoperable components, including the basic frameworks for composing them into full applications (for a variety of programming models). The real burden of constructing software components lies in *defining common interfaces* for these components. Without agreement on the basic methods, syntax and semantics for a given interface, we cannot truly work together and share software as a community. While object-oriented techniques are *sufficient* for achieving such standardized interfaces, often this has not been the case – component technology enforces the boundaries among software components to expedite the development of common interfaces.

The visualization community must take the initiative to come together and define a suite of “standard” interfaces for the wide spectrum of data analysis, transmission, rendering and presentation operations. Each operation must be generalized to produce an interface specification that adequately covers or extends existing implementations. Given this (likely informal) initial standardization effort, the wrapping up of existing tools and technology to support the generalized interfaces is relatively straightforward using CCA infrastructure. The result will be a collection of component implementations for basic visualization functions, which can be combined and swapped to compose new visualization programs.

Such an effort requires compromise and cooperation among expert teams of visualization researchers. The central roadblock is not likely technical but rather social or political; the competitive environment sowed by sparse pockets of visualization funding does not encourage “team thinking”. Yet the new SciDAC mentality has evoked a significant phase shift within other scientific communities, leading to a collaborative and cooperative spirit that should “infect” the visualization community as well. The new age of “Collaborate or Die” is upon us...

Scientific Application Integration

Adoption of the CCA component model and frameworks would expedite integration with the growing number of scientific applications being built using CCA technology. Application and visualization components could coexist in the same framework, or could communicate between frameworks via well-defined bridges. The CCA would also provide a mechanism for combining parallel rendering computations into remote or distributed visualization pipelines; current CCA frameworks support a variety of parallel, multithreaded and distributed programming models.

Common Distributed Data Model

A fundamental need for component-based software, in many scientific domains including visualization, is a common, unified data abstraction. Work is underway in the SciDAC “Terascale Simulation Tools and Technology” (TSTT) Center to define basic interfaces for structured and unstructured data objects and meshes. This work builds on preliminary work in the CCA Forum to describe dense structured rectangular arrays and their distributed data decompositions. The goal is not to replace existing data wrangling technology, but instead to *describe* the organization of data objects and wrap them with a generalized data access interface. The visualization community should collaborate with the TSTT and CCA to ensure that visualization-specific data access requirements are integrated into this ongoing development.

Generalized Parallel Data Redistribution (“MxN”)

In conjunction with a generalized data model, a crucial need for any remote and distributed visualization framework is the ability to efficiently transfer and exchange parallel data. Especially when manipulating very large datasets, it will be necessary to pipeline data operations across collections of high-performance computing resources. Parallel data elements must be efficiently mapped between each pair of pipelined stages to maintain bandwidth and reduce latency in analyzing and rendering ultrascale data, especially for interactive environments.

ORNL leads the CCA research thrust into “MxN” parallel data redistribution technology, where decomposed data is shared between parallel components of different cardinality and topology (i.e. “M” processors versus “N”). This technology can be applied to transfer massive data through the pipeline, from the simulation program or data archive through analysis and reduction to final rendering. (An exploratory “Scalable Visualization Cache” architecture, that applies MxN technology in this manner, has been proposed to DOE MICS.)

Flexibility and Minimalism of Encapsulating Framework

A framework should not overly dictate the nature of the software components that will be assimilated into it. A specific programming model or source language should not be required, and minimal demands should be made by internal framework services. The framework should provide only those services necessary for the containment and composition of software components; everything else should be a component. This is the CCA mentality for framework development, and care has been taken to minimize the degree of code modification for “CCA compliance”. Any visualization framework design should keep this perspective, to ease the adoption of the framework by external collaborators (and scientific users). It must be easy to integrate existing toolkits, as well as modularly extend the framework capabilities.

Practical Concerns

Packaging, deployment and support take serious time and effort, and therefore require additional special funding. We must have a consistent deployment scheme with repositories (SourceForge or whatever) and easy install and build systems (autoconf, automake, RPM or InstallShield, etc). (We need to decide whether we try to support both Linux/Irix *and* Windows environments.)

Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.