# Interoperable geometry and mesh components for SciDAC applications

Institute of Physics Publishing Journal of Physics: Conference Series **16** (2005) 486–490
doi:10.1088/1742-6596/16/1/067 SciDAC 2005

# Interoperable geometry and mesh components for SciDAC applications

**T. J. Tautges[*+], P. Knupp[*], J. A. Kraftcheck[+], H. J. Kim[+]**

[*1]Sandia National Laboratories, Albuquerque, NM, 87185
[+]University of Wisconsin-Madison, Madison, WI, 53706

tjtautg@sandia.gov

**Abstract**. Software components for representing and evaluating geometry (TSTTG/CGM) and finite element mesh (TSTTM/MOAB), and a higher-level component for relations between the two (TSTTR/LASSO), have been combined with electromagnetic modelling and optimization techniques, to form a SciDAC shape optimization application. The TSTT data model described in this paper allows components involved in the shape optimization application to be coupled at a variety of levels, from coarse black-box coupling (e.g. to generate a model accelerator cavity using TSTTG) to very fine-grained coupling (e.g. smoothing individual mesh elements based in part on geometric surface normals at mesh vertices). Despite this flexibility, the TSTT data model uses only four fundamental data types (entities, sets, tags, and the interface object itself). We elaborate on the design and implementation of effective components in the context of this application, and show how our simple but flexible data model facilitates these efforts.

## 1. Introduction

PDE-based computations rely heavily on discrete (mesh) and continuous (geometry) domain representations, increasingly by importing this capability in the form of software components rather than "re-inventing the wheel" locally. A recent example of a component-based approach is a collaboration between SLAC, Sandia and Carnegie Mellon University on accelerator shape (design) optimization where the electromagnetic (EM) performance of accelerator cavities is optimized through adjustment of geometric design parameters [5]. The model problem for this effort is a Low-Loss cavity model for the International Linear Collider (ILC) shown in . This application requires components for geometry and mesh which interact with EM modeling and optimization parts of the calculation at a variety of levels. The geometry and mesh parts of this application also present an interesting study in components themselves, because of their interactions together (amongst themselves and with other components in the calculations) in typical simulations, the often central role of mesh data in various types of simulations, and the variety of data from other applications (so-called "metadata") often accompanying and accessed through the mesh data.

The Terascale Tools & Technologies (TSTT) center [1] was established as part of the DOE SciDAC program [2] to develop interoperable components for geometry, mesh, and other enabling technology tools, for applications like the one described above. One of the principal outcomes of this

[1] SANDIA IS A MULTIPROGRAM LABORATORY OPERATED BY SANDIA CORPORATION, A LOCKHEED MARTIN COMPANY, FOR THE UNITED STATES DEPARTMENT OF ENERGY UNDER CONTRACT DE-AC04-94AL85000.

 

effort has been the definition of common interfaces for geometry (TSTTG), mesh (TSTTM), and data relations (TSTTR). These interfaces, implemented by our CGM, MOAB and LASSO components, respectively, play a central role in the construction of the shape optimization application described above. This paper describes the design used for these components, and shows how this design simplifies the construction of the shape optimization application.

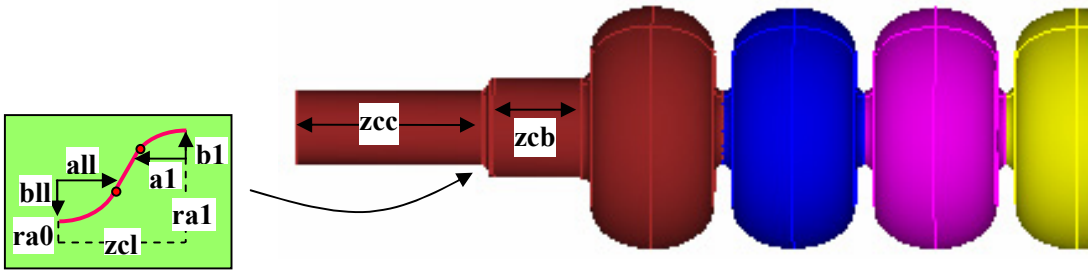


**Figure 1:** Parameterized geometric model for International Linear Collider Low-Loss superconducting cavity. Not all of the 20 design parameters are shown.

## 2. Geometry, Mesh and Relations Interface Implementations

Our components share the same basic data model, which enables both simplicity and flexibility of the components. This data model, or ontology, is described first, followed by descriptions of the actual components.

### 2.1. A Simple Ontology for Geometry and Mesh

This ontology consists of four basic data types:

**Interface Instance:** The instance of a component which serves as the overall "container" of a component's data and as the point of reference for those data.

**Entity Handle:** The data type used to reference topological entities in the geometry or mesh ("vertex", "geometric face", etc.) in calls to the component functions.

**Entity Set:** Arbitrary combinations of entities and of other sets; supports directed (i.e. parent-child) relationships between sets, which are distinct from a set containing another set.

**Tag:** An arbitrary piece of data stored on entities, sets, or the interface itself.

This ontology is used to interface to the various types of geometry, mesh, and other application data found in many computational simulations; specific examples of this are given later in this paper.

### 2.2. CGM: A Component For Geometry Representation [3]

The Common Geometry Module (CGM) is a code library providing CAD geometry functionality used for mesh generation and other simulation applications. CGM provides functions for geometry creation, query and modification, tools for geometry decomposition and non-manifold topology representation, and support for loading solid models on parallel computers. CGM uses the ACIS solid modeling engine, and has support for facet-based and "virtual" geometry. CGM implements TSTTG, which provides a core set of commonly needed functions, as well as functionality for the tags and sets described in Section 2.1. CAD-based geometric modeling is crucial for high-fidelity accelerator modeling applications of the type described later in this paper and in Ref. [5]. For example, the model shown in  is constructed by a C++ function which calls CGM through the TSTTG interface.

### 2.3. MOAB: A Component For Mesh Representation [4]

MOAB is a component for representing and evaluating finite element mesh data. MOAB represents elements in the finite element "zoo" as well as polygonal and polyhedral elements, and can store hexahedral meshes in structured or unstructured formats. MOAB is optimized for efficiency in memory and cpu time (in that order). Figure 2 shows that mesh access times for MOAB are comparable to the C++ object-based representation like those in CUBIT, while memory costs are substantially less in MOAB. Figure 2 also shows the cost savings of using MOAB's TSTT-like C-language interface compared to calling through the standard SIDL-based TSTTM interface.

2.4. LASSO: A Component For Relating Geometry & Mesh Domain Data

Although modern finite element meshing tools require access to both geometry and mesh, we keep the representations of mesh and geometry separate and independent, recovering when necessary the relations between mesh and geometry which existed when the mesh was generated. This separation results in finer-grained components for geometry and mesh, which increases potential for re-use without duplicating application-native functionality. The TSTTR interface, implemented in our LASSO component, performs the tasks of finding, storing, and retrieving these relations between geometry and mesh data. Relations are stored in terms of tags on the geometry (pointing to associated mesh) and the mesh (vica versa); since tags can store arbitrary data, this does not introduce dependencies between the geometry and mesh components themselves. In the future, this functionality will also be useful for relating geometry and mesh data to those in other components (e.g. fields and discretizations).

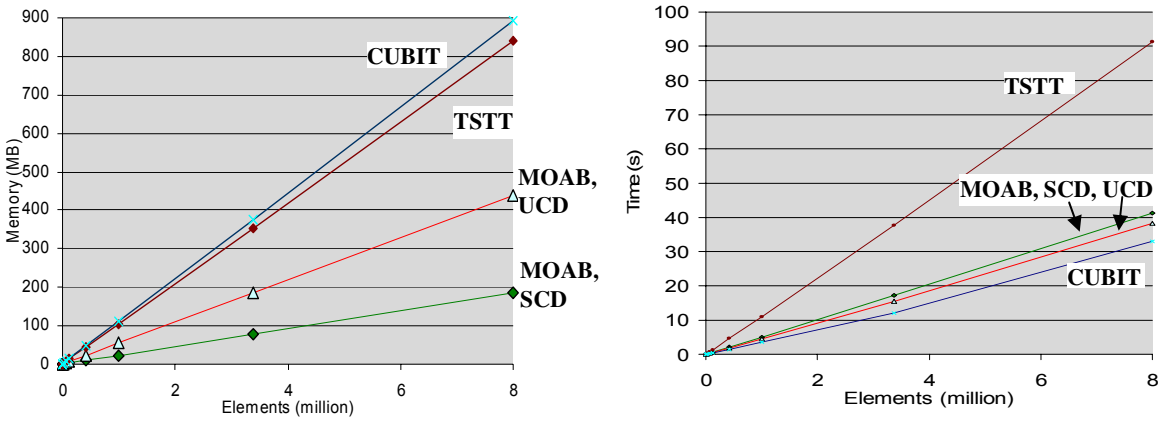


**Figure 2:** Memory usage (left) and access time (right) for MOAB unstructured (UCD) and structured (SCD) representations. MOAB's native interface is compared to CUBIT datastructures and MOAB's SIDL/Babel-based TSTTM interface implementation. Performance of MOAB's C-based TSTT interface is comparable to MOAB's native interface in most cases.

2.5. Mesquite: A Mesh Quality Improvement Toolkit

Mesh quality can affect both robustness and accuracy of finite element analyses. The Mesquite toolkit uses optimization-based techniques, in concert with mathematically rigorous quality objective functions, to achieve mesh improvement in critical regions without significant quality degradation in other regions. See Ref [6] for more details.

## 3. Shape Optimization Application

In this collaboration between Sandia/TSTT, Stanford Linear Accelerator Center, and Carnegie Mellon University, components from the various institutions are being integrated into an overall shape optimization capability, where design parameters determining the shape of the accelerator cavity shown in are being tuned such that electromagnetic performance of the cavity is optimized [5]. The overall procedure used for this effort is shown in Figure 3. This procedure makes calls to the CGM, MOAB, LASSO, and Mesquite components, as well as incorporating optimization techniques from CMU into the SLAC analysis code Omega3P. Focusing on the geometry and mesh, implemented in the "DDRIV" section of Figure 3, this procedure a) generates a geometry *G(p)* based on a new shape parameter vector *p* using a procedure which calls TSTTG/CGM (the model shown in is generated by this procedure, for example); b) relates a pre-computed mesh $M_o$ to the new geometry and projects the mesh to that model using TSTTG/CGM and TSTTR/LASSO; c) Untangles & smoothes the mesh onto the new model using Mesquite; and d) repeats steps a-c for small perturbations δp of each parameter, computing the sensitivity of boundary mesh vertices with respect to those changes (so-called "design velocities").

Data is communicated between the various components, inside and outside DDRIV, using tags and sets provided in the TSTT data model. For example, tags are used to mark geometric boundary mesh vertices to tell Mesquite not to move these vertices; sets are used to group mesh faces for assignment of boundary conditions and communication of those to Omega3P; and design velocities are stored as tags on mesh vertices for communication to the optimization procedure. Communication of these diverse types of data using the same tag mechanism in TSTTM and TSTTG shows the flexibility of the data model. This example also shows coupling between components at a variety of levels, from high-level access (e.g. calling TSTTG as a black box from a function to generate the ILC geometry) to very low-level, fine-grained access (e.g. optimizing mesh vertex positions in part by evaluating the geometric surface normal at each boundary vertex). The ability to couple at these very different levels is also an example of the flexibility of the data model.

Many applications in addition to the SLAC application exhibit this behavior of accessing geometry and mesh data at both coarse- and fine-grained levels. The flexibilities in data types and level of access is crucial to component-based applications, since they facilitate changing interactions between components and tools (something which is quite likely during application development) without modifying the component interfaces or the components themselves (something which should be quite rare). This flexibility will also be crucial for extending this shape optimization approach to different accelerator modes and to different applications altogether.

The benefit of using software components, for shape optimization and other applications, comes not only from relieving the application from having to develop the components' capabilities locally, but, just as importantly, by providing a vehicle for delivering future improvements to applications through standard interfaces. This has been the case during the development of the DDRIV driver application, where new capabilities in Mesquite were accessed simply by adjusting startup options and calling through the same Mesquite interface used previously. Modifications to the overall process flow were also made simpler because of the flexibility inherent in the data model; for example, the geometric model used in a given iteration of the optimization can either be read from disk, or generated by DDRIV using a dynamically-assigned design parameter vector; subsequent steps of the iteration (relating mesh, smoothing, etc.), proceed from there without depending on the method used to obtain the geometric model. In fact, standard interfaces go one step further, by allowing wholesale substitution of components to gain access to new techniques.

## 4. Component Design and Implementation Issues

As part of this development effort, we have observed several characteristics which strongly influence the usability and efficiency of our components, and which are important elements of the design of these components. Briefly, these characteristics are:

**Component Scope:** Defining the proper scope of a component involves tradeoffs. In essence, a component should be just large enough in scope to cover some logical grouping of functionality, while being small enough that its scope does not needlessly overlap with that of other components or with code already implemented in most applications. In general, making finer-grained components gives applications more options for which pieces of functionality to use. It is for this reason we choose to package geometry and mesh as separate and independent components.

**Ontology:** As with component scope, there are tradeoffs in defining a component's ontology: having too few semantic types makes it difficult to verify semantic correctness, while having too many or too specific semantic types makes it difficult to extend the functionality of a given component (something that is often needed relatively frequently) without also changing the component's ontology (something that *should be* quite infrequent). The ontology for the TSTT components is effective, as demonstrated by its ability to communicate both low- and high-level data between related and unrelated components, even though only four basic data types are used.

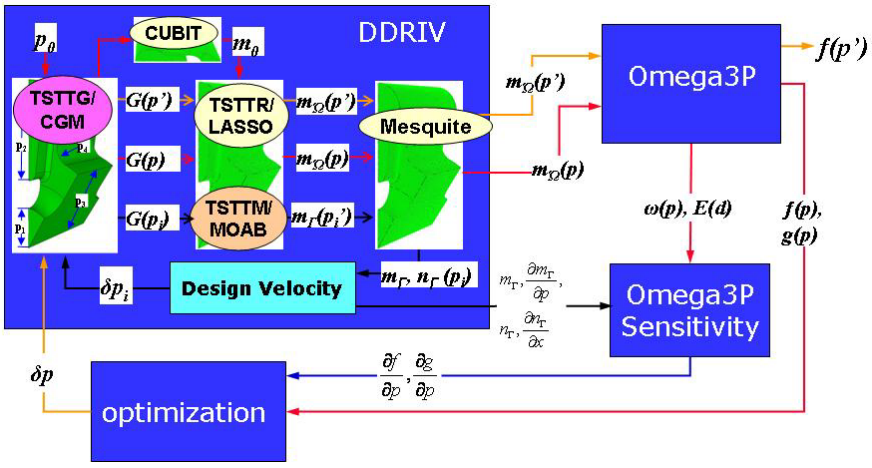


**Figure 3:** Overall shape optimization process flow; ddriv tool is box on upper left.

**Implementation & Efficiency**: Good component designs balance the opposing goals of a) efficient access to component data and calls across component interfaces, and b) support for rapid assembly of components to support new applications. This balance has proven quite challenging in practice, particularly during the definition of the data model used in and functional interfaces exposed by the TSTT components. The following capabilities are relevant to this issue and must be considered carefully:

- Native storage & minimized data copying
- Aggregate access to fine-grained data
- Use of the right component framework (if any)

Overall we feel we have struck the right balance between these goals, at least as shown by the rapid construction of the shape optimization application; real performance numbers will be useful for evaluating this assertion quantitatively.

## 5. Conclusions

This paper describes our design and implementation of components for geometry, mesh, and data relations between the two. A shape optimization application is described which shows the simplicity and flexibility of the data model implemented by these components, and how the components can be used to store and retrieve data commonly encountered in simulations of this type. Designing effective geometry and mesh components (measured by the simplicity and flexibility with which they can be used in real applications) has required a careful balance of component scope, design of the component ontology, and implementation of that design.

## References

[1] The Terascale Simulation Tools and Technology (TSTT) Center, http://www.tstt-scidac.org/.
[2] SCIDAC: Scientific Discovery Through Advanced Computing, http://www.csm.ornl.gov/scidac/.
[3] Timothy J. Tautges, "CGM: a Geometry Interface for Mesh Generation, Analysis and Other Applications", Engineering with Computers, 17:299-314 (2001).
[4] Timothy J. Tautges, Ray E. Meyers, Karl Merkley, Clint Stimpson, Corey Ernst, "MOAB: A Mesh-Oriented Data Base", Sandia National Laboratories Report SAND2004-1592, Sandia National Laboratories, Albuquerque, New Mexico, 2004.
[5] K. Ko, "Impact of SciDAC on Office of Science Accelerators through Electromagnetic Modeling", SciDAC2005, June, 2005.
[6] M. Brewer, L. Diachin, P. Knupp, T. Leurent, D. Melander, "The Mesquite Mesh Quality Improvement Toolkit", Proceedings, 12th International Meshing Roundtable, Sandia National Laboratories report SAND 2003-3030P, Sept. 2003.