Final Report for NAG1-02040

CAPRI:
A Geometric Foundation for Computational Analysis and Design

Robert Haimes
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
haimes@mit.edu

Introduction

The computational steps traditionally taken for most engineering analysis (CFD, structural analysis, and etc.) are:

- Surface Generation -- usually by employing a CAD system
- Grid Generation -- preparing the volume for the simulation
- Flow Solver -- producing the results at the specified operational point
- Post-processing Visualization -- interactively attempting to understand the results

For structural analysis, integrated systems can be obtained from a number of commercial vendors. These components couple directly to a number of CAD systems and are executed from within the CAD GUI. It should be noted that the structures problem is more tractable than CFD; there are fewer mesh topologies used and the grids are not as fine (this problem space does not have the length scaling issues of fluids).

For CFD, these steps have worked well in the past for simple steady-state simulations at the expense of much user interaction. The data was transmitted between phases via files. In most cases, the output from a CAD system could go IGES files. The output from Grid Generators and Solvers do not really have standards though there are a couple of file formats that can be used for a subset of the gridding (i.e. PLOT3D data formats and the upcoming CGNS). The user would have to patch up the data or translate from one format to another to move to the next step. Sometimes this could take days. Specifically the problems with this procedure are:

- File based. Information flows from one step to the next via data files with formats specified for that procedure. File standards, when they exist, are wholly inadequate. For example, geometry from CAD systems (transmitted via IGES files) is defined as disjoint surfaces and curves (as well as masses of other information of no interest for the Grid Generator). This is particularly onerous for modern CAD systems based on solid modeling. The part was a proper solid and in the translation to IGES has lost this important characteristic. STEP is another standard for CAD data that exists and supports the concept of a solid. The problem with STEP is that a solid modeling geometry kernel is required to do anything with this type of file.

- ‘Good’ Geometry. A bottleneck in getting results from a solver is the construction of proper geometry to be fed to the grid generator. With ‘good’ geometry a grid can be constructed in tens of minutes (even with a complex configuration) using unstructured techniques. Adroit multi-block methods are not far behind. This means that a multi-million node steady-state solution can be computed on the order of hours (using current high performance computers) starting from this ‘good’ geometry. Unfortunately, the geometry usually transmitted from the CAD system is not ‘good’ in the grid generator sense. The grid generator needs smooth closed solid geometry. It can take a week (or more) of interaction with the CAD output (sometimes by hand) before the process can begin.

- One-Way Communication. All information travels on from one phase to the next. This makes procedures like node adaptation difficult when attempting to add or move nodes that sit on bounding surfaces (when the actual surface data has been lost after the grid generation phase).

Until this process can be automated, more complex problems such as multi-disciplinary analysis or using the above procedure for design and optimization becomes prohibitive. There is also no way to easily deal with this system in a modular manner. One can only replace the grid generator, for example, if the software reads and writes the same files.

Instead of the serial approach to analysis as described above, **CAPRI** takes a geometry centric approach. This makes the actual geometry (not a just discretized version) accessible to all phases of the analysis. The connection to the geometry is made through an Application Programming Interface (API) and NOT a file system. This API isolates the top-level applications (grid generators, solvers and visualization components) from the geometry engine. Also this allows the replacement of one geometry kernel with another, without effecting the top-level applications. For example, if UniGraphics is used as the CAD package then Parasolid (UG’s own geometry engine) or UGOpen can be used for all geometric queries so that no solid geometry information is lost in a translation. This is much better than STEP because when the data is queried, the same software is executed as used in the CAD system. Therefore, one analyzes the exact part that is in the CAD system without translation (and the errors generated during that step).

CAPRI uses the same idea as the commercial structural analysis codes but does not specify control. Software components of the CAD system are used, but the control of the software session is specified by the analysis suite, not the CAD operator. This also means that the license issues (may be) minimized and individuals need not have to know how to operate a CAD system in order to run the suite.

The **CAPRI** API

CAPRI is a software building tool-kit that refers to two ideas; (1) A simplified, object-oriented, hierarchical view of a solid part integrating both geometry and topology definitions, and (2) programming access to this part or assembly and any attached data.

A complete definition of the geometry and application programming interface can be found in the document "**CAPRI**: **C**omputational **A**nalysis **PR**ogramming **I**nterface" appended to this report. In summary the interface is subdivided into the following functional components:

1. Utility routines -- These routines include the initialization of **CAPRI**, loading CAD parts and querying the operational status as well as closing the system down.

2. Geometry data-base queries -- This group of functions allow all top level applications to figure out and get detailed information on any geometric component in the Volume definition.

3. Point queries -- These calls allow grid generators, or solvers doing node adaptation, to snap points directly onto geometric entities.

4. Calculated or geometrically derived queries -- These entry points calculate data from the geometry to aid in grid generation.

5. Boundary data routines -- This part of **CAPRI** allows general data to be attached to Boundaries so that the boundary conditions can be specified and stored within **CAPRI**'s data-base.

6. Tag based routines -- This part of the API allows the specification of properties associated with either the Volume (material properties) or Boundary (surface properties) entities.

7. Geometry based interpolation routines -- This part of the API facilitates Multi-disciplinary coupling and allows zooming through Boundary Attachments.

8. Geometric creation and manipulation -- These calls facilitate constructing simple solid entities and perform the Boolean solid operations. Geometry constructed in this manner has the advantage that if the data is kept consistent with the CAD package, therefore a new design can be incorporated directly and is manufacturable.

9. Master Model access – This addition to the API allows for the querying of the parameters and dimensions of the model. The "feature tree" is also exposed so it is easy to see where the parameters are applied. Calls exist to allow for the modification of the parameters and the suppression/unsuppression of nodes in the tree. Part regeneration is performed by a single API call and a new part becomes available within **CAPRI** (if the regeneration was successful). This is described in a separate document.

Components 1-7 are considered the **CAPRI** base level reader.

CAPRI's Current Status

The CAD/Geometry Kernel coverage has been broadened under this contract. The tables below display the status of the various **CAPRI** components against all supported modelers. The reader component is the basic level of support allowing for loading and querying topology and geometry. The Solid Creation and Solid Boolean operation module supports the making of simple solids and intersecting, subtracting and fusing any solid loaded into or created from within **CAPRI**. The Master Model component allows for parametric control of parts as well as the control of the shapes of certain components. This component can also control defeaturing.

CAD System->	Catia V4	Catia V5	ComputerVision		I-DEAS
Interface	CAT GEO	CAA RADE	CADDS	Felisa (Native)	OpenIDEAS
Base Reader	X	X	X	X	X
Solid Creation & Boolean Ops	X	X	X	--	X
Master Model		X		N/A	X

CAD System->			Pro/ENGINEER	SolidWorks	UniGraphics
Interface	OpenCASCADE	Parasolid	Pro/TOOLKIT		UGOpen
Base Reader	X	X	X	X	X
Solid Creation & Boolean Ops	X	X	X	X	X
Master Model	N/A	N/A	X	X	X

It should be noted that the full test matrix has another direction. Each of these components can function against the complete suite of workstations:

Workstation->	LINUX	SGI	Windows NT/2000/XP	IBM RS/6000	Other UNIX
Catia V4		X		X	
Catia V5			X		
Computer Vision		X	X		
Felisa	X	X	X	X	Alpha, HP, Sun
I-DEAS		X	X		
OpenCASCADE	X	X	X		
Parasolid	X	X	X	X	Alpha, HP, Sun
Pro/ENGINEER	X	X	X		
SolidWorks			X		
UniGraphics	X^{64bit}	X	X		

Significant Accomplishments Under This Contract

Closely coupled integration of Computer Aided Engineering (CAE) applications with the CAD system presents a significant challenge for most simulation-based analyses and is often overlooked or more simply avoided. Clearly the kind of on-demand access to the CAD model from the native CAD system (as handled by **CAPRI**) must be available in order to streamline the downstream analysis and design applications. However, achieving this is arduous for at least two reasons and has major implications for using native CAD models directly in CAE. First, CAD APIs are extremely complex and significant resources have to be devoted to creating and maintaining the dedicated connection to a CAE application. Second, CAD access and licenses may be limited in an engineering enterprise particularly for parallel execution (in cluster computing based) multidisciplinary analyses or optimization. There are several examples of structural and fluid analysis packages with dedicated direct connections to CAD systems. However, a major drawback of these packages is that the user is forced to work within their GUI environment that, in general, requires manual intervention. This lack of generality for implementing and automating complex multidisciplinary problems is a fundamental problem. These difficulties in CAE tool integration has also contributed to the adoption of the translator approach which is not tied to any CAD system and is viewed as simpler to build into a software system.

The software implementation of **CAPRI** now illustrates these novel approaches to addressing these issues. These include:

- *CAD-neutral CAE-focused API.* The API is focused on simulation-based analysis and design requiring no expertise in Computational Geometry or specific CAD-vendor API programming. Writing and maintaining applications that interface with downstream mesh generators, solvers and optimizers in a tightly coupled manner can be done once through the API; all of the major CAD vendors are then automatically supported. Tracking various CAD revisions and releases is handled by CAPRI, simplifying the CAE software maintenance.

- *Lightweight Applications.* Building applications that tightly couple to the CAD system can require a complex build so that all of the software components of the system are linked together (including components of the CAD system itself). This makes the application specific to that CAD environment. This is now not the case for **CAPRI**; the build has been simplified by encapsulating portions of the API that are CAD-specific into software components using DLLs or share object libraries. The application link then only requires the **CAPRI** libraries. When executing the application the CAD-specific portions are loaded at runtime. Again, this can greatly simplify upgrading to new revisions of a CAD system where each version is handled by the appropriate dynamically loaded library. This software handling enables the application to remain lightweight and truly CAD-neutral, where each CAD system is handled by its own dynamically loaded back-end.

- *Client/Server Deployment*. CAE applications that need on-demand access to the CAD services do not have to be co-located with the CAD resources, and can be distributed among various disciplines in an enterprise. This enables the best use of an enterprise's CAD resources. This also allows for the use of CAPRI on systems that do not support CAD systems such as Apple's Macintosh running OS X and ITANIUM based servers (for example *Columbia* at NASA Ames Research Center).

  The client/server implementation in CAPRI is based upon **SOAP**, which uses RPC and XML to pass the data between client and server. The CAPRI port uses gSOAP, which provides a pre-compiler that generates the XML (from C structures/C++ class definitions) and the RPC stubs from the function definitions found in specific headers. The result has minimal impact as far as the network is concerned, this requires nothing more than what a Web browser does (at the client-side) and acts like a web server for the **CAPRI** server (requiring only an open port). The web-based technologies utilized in **CAPRI** are described below:

  - *SOAP*. Simple Object Application Protocol is used as the foundation for CAPRI client/server. SOAP is RPC based and data gets passed between client and server using XML. This is all hidden from the CAPRI application programmer.

  - *Attachments for moving files*. DIME is used to move the CAD/Geometry Kernel files from client to server, so that the server can have easy access. DIME is also used to move files back to the client after models have been saved. This provides a situation where access, ownership and permission are not an issue. The server owns the files (on the server machine) after being transmitted.

  - *Message Compression*. All XML and DIME communication passes through a compression phase. The data is uncompressed upon arrival. This was done with the notion that the bottleneck will always be network bandwidth and spending some processor time to (un)compress the data is a more optimal approach.

  - *Secure SSL Option*. HTTPS can be used for situations that require more security. There is a secure server and a matching secure client-side dynamic back-end. Additional steps need to be taken to generate protected certificates and keys.

  - *Access through proxy servers*. When the **CAPRI** server is not visible through the local network (but requires a proxy server to get to the machine), the **CAPRI** client-side back-end can be notified to go through the proxy server. This is done via environment variables so that no API changes are required.

Though this contract was neither responsible for **CAPRI**'s inception or was it the only funding source during the period, it was primarily responsible for the directions listed above.

Progress over the Last Year

• Maintenance and Support

General maintenance and support of the **CAPRI** API has been provided in order to assist continued development of derivative applications. This included assistance with problems encountered with building against all supported systems, additional I/O issues uncovered as part of application testing, enhancements to the existing API.

• Client/Server

The first state of the web-based client/server implementation in **CAPRI** (using **gSOAP**) had been conceptualized, written and evaluated at the beginning of this task. It worked and functioned efficiently for many applications but there are issues in the implementation. The problems addressed during this last phase of the grant were:

- *Multiple clients accessing the same model*. The first implementation segregated data from each client. This was done for security reasons. The server ends up owning all data (on the machine that the server is running upon) and therefore needs to keep it separate. This causes a problem in the case of a parallel job where all processes are looking at the same model. Each will load the model overloading the server with a duplication of data. A scheme was developed whereby the initial client to load the model (i.e. the owner) can request an encoded and encrypted *key* (a character string) that defines the load. If the owner passes this string to other clients wishing to gain access to the model then the non-owing clients can load the model by *key*. The owner is the only client that can change, mark-up or remove the model from the server. This required no changes to gain access at the API level. Clearly, this solution deals with the issue of security (ownership/privileges) and permission.

- *Collections of **CAPRI** calls/macros executing at the server.* Because of the network-based nature of the client/server implementation there are some situations where it was noticed that performance is a problem. This was seen during mesh generation where many thousands of queries of the geometry may be required in that each query will generate a request packet and then a response packet. This was mitigated by allowing for collections of certain **CAPRI** calls to be concatenated and treated as a single network request/response. The result provided much greater performance but uncovered other issues with both *Pro/ENGINEER* and *SolidWorks*. These CAD packages have APIs that also operate in a client/server fashion (specifically, these modes are required for **CAPRI**s dynamic back-end operation). This means that the collected group of packets needed to be passed on to each system as the aggregated whole. This was accomplished by dynamically loading a **CAPRI**-based *plug-in* to be resident directly with the CAD system.

- Continued Improvements in Surface Meshing

Over the last couple of years there has been great success in the use of quadrilateral methods to quickly generate surface triangulations. This is impressive for CAD Faces that are naturally trimmed on isoclines where the spacing of the mesh can then be driven by the Edge discretization alone. The resultant meshes are anisotropic, follow the surface curvature, and have minimal counts in comparison to the default (isotropic) scheme. Also, the maximum angles found in the triangulation are well bounded (since these tend to be right-triangles).

During last year grant there were attempts at taking current isotropic surface tessellator and supporting anisotropic triangular meshing by changing the triangle side swapper. One of the swapping techniques used in the triangulation scheme drives the tessellation toward isotropic (using a MinMax predicate). This is done in the underlying surface's parameter space *(u,v)*. Since the parameter space is artificial (i.e. not physical) any 2D mapping could be used. Therefore a transformation from a 2D space that could support an anisotropic stretching to the surfaces parameter space would be all that is required to achieve the anisotropy found in the quadrilateral patch method. There are a number of approaches that allow for anisotropic triangular meshing. In general these schemes locally remap the space and triangulate against some isotropic predicate in the stretched space. They require a background grid or some way to get local curvature throughout the surface being meshed. This is difficult in our procedure in that we are attempting to generate the surface triangulation for the first time (and with no original reference). It is also not clear what degree of control these approaches offer in regards to mesh orthogonality.

Changing the swapping scheme did not work well, but what did work was an insertion scheme that placed new vertices at intersections of a *(u,v)* grid. From preliminary tests this new alignment scheme provides nice anisotropic meshes but needs further testing. Also, there are now 3 surface triangulation methods. The selection of the appropriate scheme based on the underling surface and its trimming must be automated. Part of this task is to look at techniques to automatically select the appropriate method (either isotropic, quadrilateral templates or alignment). Not much progress was made. Time and effort was diverted in support of a *Quilting* package that has been added to **CAPRI**. This allows for sliver Faces to be integrated into neighboring (larger) Faces. The triangles associated with the Face are then merged so less geometry will be presented to the application.

In a real sense *Quilting* changes the solid part's topology without affecting the underlying geometry. This is an important characteristic; the physical object remains unchanged but the topology is being modified to potentially reflect the engineering analysis (and not the method of CAD construction).

The effort here was is in support of work done at Syracuse University by Prof. John Dannenhoffer and has resulted in the following papers:

- Dannenhoffer, J. and Haimes, R., "Surface Parameterization of 3D Configurations via Quilts", AIAA Paper 2005-5238, June 2005.

- Dannenhoffer, J. and Haimes, R., "Using Quilts to Generate Grids from Imperfect CAD Assemblies", AIAA Paper 2006-0943, Jan. 2006.

Deliverables

Throughout the life of this contract access to **CAPRI** has been given to NASA Langley Research Center. This access is the most concrete deliverable in that it embodies all of the research and development (except for the *Quilting*) described above.

Access to **CAPRI** has facilitated in the construction of the NASA Langley Research Center's grid generation suite **GridEx**, which contains both an interactive component and a non-interactive mode (**BatchEx**). Being able to query actual geometry (in an easy to build and maintain manner) has also allowed for the construction of the LaRC software package **REFINE** that allows for tetrahedral mesh adaptation. In this solver driven grid modification application **CAPRI** is used to move vertices directly on the surfaces of the bounding volume of interest.