Common API: Using Visual Basic to Communicate between Engineering Design and Analytical Software Tools

Prince, Stewart P., Ryan, Robert G., and Mincer, Tom

California State University, Northridge

Submitted for the 2005 ASEE Annual Conference

Abstract

Mechanical Engineering students at California State University, Northridge currently use the SolidWorks© suite of design and analysis tools to complement classroom learning. In particular, SolidWorks, CosmosWorks©, CosmosMotion©, and FloWorks©, plus Microsoft Excel© are used to solve different types of engineering problems. Communication between the different packages can be simplified and enhanced by the use of the Visual Basic Application Programmer Interface (API). Using the API, model files created in Solidworks can be manipulated directly inside of a control program such as Excel and the resulting altered parameters can be returned for further review. This paper describes how the API functions and how data can be transferred between the previously mentioned software tools. A machine design example is presented showing how a simply supported shaft can be designed in Excel, modeled in Solidworks, and analyzed in CosmosWorks for stress and deflection. The results, including stress, deflection, and mass properties are then returned to Excel for review.

Introduction

The Mechanical Engineering Department at California State University, Northridge has been using the Microsoft Excel computing environment as its primary programming "language" for more than a decade. The familiar capabilities of the worksheet environment, coupled with the ability to perform complex calculations with Visual Basic for Applications (VBA), provide an effective platform for analyzing, designing, and optimizing engineering systems.

Since 2000, the Department has been using SolidWorks and the associated COSMOS analysis packages as the tool for designing and analyzing parts and assemblies. The continuing development of the Visual Basic language, and its ability to communicate with the SolidWorks "design suite" via the Applications Programmer Interface (API), has opened up new possibilities for creating an integrated computing environment for design and optimization.

The Visual Basic programming language is based on the BASIC (Beginners All-Purpose Symbolic Instruction Code) language, a language used by more programmers than any other in the history of computing. The "Visual" part of the name refers to the method

used to create the graphical user interface (GUI). Rather than writing numerous lines of code to describe the appearance and location of interface elements, the user puts pre-built graphical objects onto the screen thus significantly reducing programming time [1].

Visual Basic has evolved from the original BASIC language. It now contains many new statements, plus keywords which relate directly to those found in the Microsoft Windows GUI. Students beginning to learn Visual Basic can create powerful applications using a few keywords yet the language lends itself to powerful advanced programming and fits well into the engineering curriculum.

Visual Basic supports the Integrated Development Environment (IDE), a working environment that integrates program design, editing, compiling, and also debugging into a common environment. Other programs would have each function in a separate program, making development more difficult.

The API allows the student to program using Visual Basic while *inside* of another application. Many applications, such as Microsoft Windows based ones, as well as the Solidworks suite, DP Technologies Espirit, and countless others, support Visual Basic and the API. Using the API, students can write their own custom programs while inside of another application, and can also communicate directly between them. This makes Visual Basic and the API an important tool for students as they are required to learn many different applications while pursuing an engineering degree.

An application's API may support more than one language (Visual Basic, C++) thus there is a distinct difference between the API and a programming language: an application can contain an API which supports different languages while a programming language such as Visual Basic can be used to create a program inside the application.

Understanding Visual Basic

Because Visual Basic is so easy to learn and so very powerful, it is often used by engineering students to create custom programs and solve problems. The most fundamental part of a Visual Basic program is the *form*. The form object is the most fundamental building block of the interface upon which controls can be placed and behind which code can be created. A program can contain many forms.

Controls are objects to be contained within a form. There are virtually thousands of controls available and include: *textboxes, labels, timers, dialog boxes, buttons*, and many more. Most students will use controls that allow for entering and displaying data, accessing other applications, and the processing of data.

Each *object* in Visual Basic has its own set of: *properties*, *methods*, and *events*. For example, consider a telephone (not a Visual Basic object, but simply used to demonstrate the concept). The telephone's properties might include attributes such as size, type, and

color. The telephone's methods, or actions might include dialing, or hanging up. There can also be responses to events, such as responding to dialing. A *class* differs from an object, in that it defines the structure, or behavior of an object that will be created during execution [2].

Visual Basic code can be added to a program in two ways: (1) behind a form (2) into a module. A form is graphical but behind it resides a coding area. This area is available for general coding of subroutines as well as code that responds to an object's events. A *module* is another coding area available for subroutines, but cannot be used for event code.

A typical Visual Basic program might be created as follows: Assuming the purpose of the program is to calculate an output value (stress, for example) based on a set of inputs (say, force, area), a form can be created with textboxes, labels, and buttons (e.g. calculate button) on it (see Figure 1) Underneath the calculate button resides a subroutine that will calculate stress based on the values in the force and area textboxes. Clicking on the calculate button creates an event which activates the calculation subroutine. The subroutine will then send the value to the output stress textbox. If the program in which the macro resides is Microsoft Excel, the output can be written to a sheet cell as well, but how the data is manipulated depends on where the macro resides.

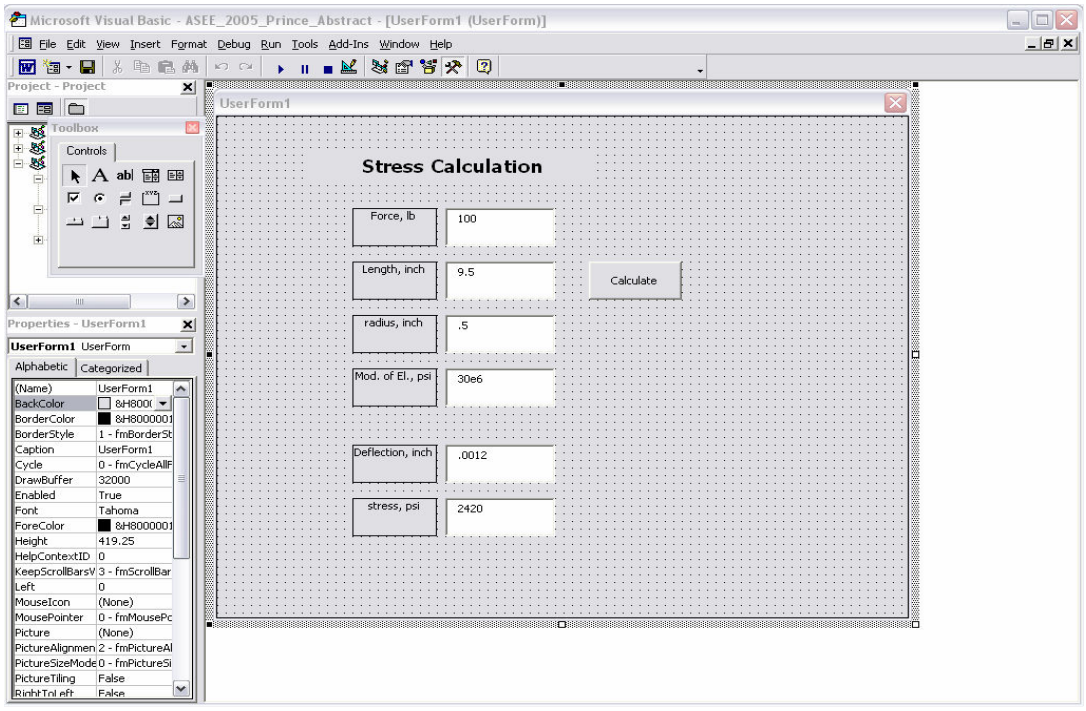

Figure 1. Visual Basic API layout inside of Microsoft Word, showing a form with textboxes, labels, and a button.

Once the program has been created, execution can be started by either running it directly from the IDE as an interpreted program (macro), or by compiling it and running it as a standalone program (executable file). In most cases the student will embed the macro in

an application and run it directly from the IDE, or create a button somewhere in the application that executes it.

SolidWorks and the Design Suite

The SolidWorks design suite is a set of applications created for Computer Aided Design (CAD). Like many other programs, SolidWorks supports the API, and offers a huge number of unique objects that can be used to perform simple tasks such as data manipulation yet are also powerful enough to build an entire assembly from a set of guide equations. This approach is significantly more powerful than the "design table" option in SolidWorks, which is typically used to control part geometry via an Excel worksheet which has cells linked to key dimensions.

The heart of the suite is a modeling program where the solid is first created. Other programs within the suite are accessed from the same SolidWorks GUI making the access virtually seamless and simple. This reduces the time required by the student to learn the Cosmos analysis packages, since the menus and terminology used are similar. Note the COSMOSWorks menu heading in the SolidWorks screen shown in Figure 2.

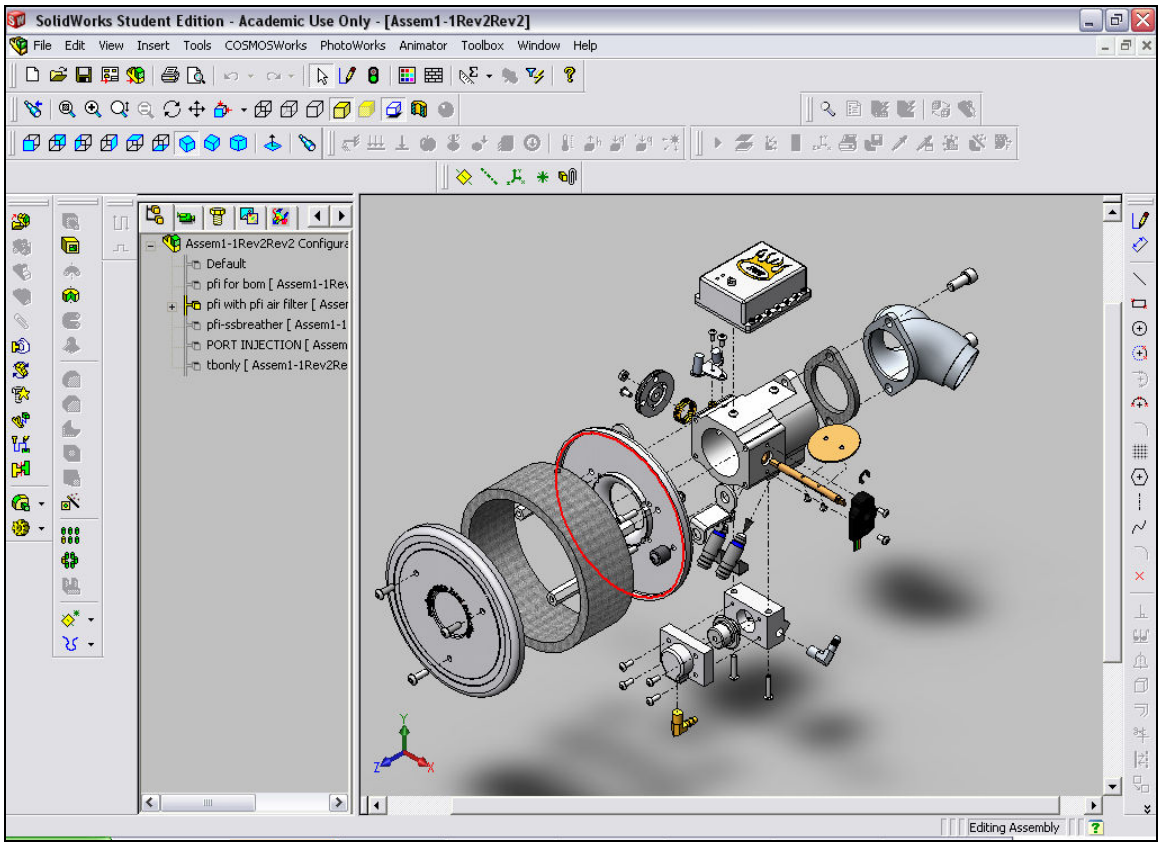

Figure 2. Example assembly created using Solidworks, exploded view shown.

Although the main SolidWorks program can perform other tasks such as the creation of assemblies of parts and drawings, it is the integration of other programs within

SolidWorks that makes the suite so powerful. The suite contains design tools that perform tasks such as: kinematic, finite element stress, fluid flow, manufacturing, and heat transfer analysis (and much more). There is complete associativity between the suite meaning changes in one program are reflected in another[3].

Using the suite, the student begins by modeling a part. A series of default features (three default datum planes and a coordinate system) are automatically created and are visible in the feature manager window. The student begins the creation of the solid by creating a simple sketch onto any plane available. This sketch is then either extruded or revolved to create the base solid (also a feature). Material can then be added or subtracted using this technique until the model (single model is a part) is complete.

With the part complete, a number of types of analysis can be performed . For instance, if the student desires to perform a stress analysis on the part (or assembly) the COSMOSWorks Stress option is then selected, exposing the stress interface containing all menus associated with finite element stress analysis. After applying the necessary loading and constraints, the model is meshed and the type of analysis is selected (static, dynamic). Still within SolidWorks, the analysis is performed and results returned. All important information such as stress, deformation, strain, and factor of safety is available both graphically and in numeric format. Since complete associativity exists, the geometry of the model can be changed and all analysis parameters will be updated directly, making iteration a very simple task.

SolidWorks API

SolidWorks and many of the programs within the suite support Visual Basic through the API. This allows the student to (1) create programs to simplify and automate operations within the suite (2) create programs to perform preliminary analysis prior to or during model creation (3) share data with programs outside the suite that support the API.

SolidWorks has created a rich array of objects which themselves contain methods, properties, and events. The list is too long to discuss in detail here thus only a few will be considered. Usually, a Visual Basic program, or macro, will begin with the "getobject" method (function), or the "createobject" method. These methods get, or create the instance of the needed application. If inside of SolidWorks, the getobject function would be used, and, for example, if inside of Microsoft Excel, either could be used depending on whether or not the application is already open [4].

For example, consider the following. Assuming a part has been created in SolidWorks, and has a feature with a dimension named "D1," a snippet of code taken from a macro written inside of the SolidWorks application might look like:

```
Sub example
*
```

```
*
*
Set swapp = GetObject(, "sldworks.application")'

Set part = swapp.ActiveDoc'

Set feature = part.FirstFeature'
*
*
*
end sub
```

The first line obtains the application object (note that the *set* command must be used in the creation of objects). Once a new instance of an external application has been created and a reference to the application object obtained, the various objects beneath the top-level application object as well as the various properties and methods may be accessed, such as the active part, or document, and the features. Thus, the first feature available is now exposed as well as all of its methods, properties, and events:

- The swap object, or solidworks_application object, is created.
- The part object, or solidworks part object, is created from the active part in solidworks.
- The feature object, or solidworks feature object, is created from the first feature found in the part.

Now consider the remainder of the subroutine. As long as features are available in the current part, the program begins cycling through them, exposing whatever information is required. In this case, it is desired to find a dimension named "D1" and change its value.

```
*
*
*
While Not feature Is Nothing' as long as feature is not nothing, continue while loop

 featurename = feature.Name 'expose the name property of each feature

Set thedispdimen = feature.GetFirstDisplayDimension' get the dimension object of the
feature object

While (Not thedispdimen Is Nothing)' as long as the dimension is not nothing, continue
while loop

Set thedimen = thedispdimen.GetDimension' create and get the dimension object of the
feature

thevalue = thedimen.Value' get the value property of the dimension object
```

```
thename = thedimen.Name'get the name property of the dimension object

If thedimen.Name = "D1" Then'keep cycling through the features until the d1 dimension
if located

 thedimen.Value = 25'change the value (property) of the d1 dimension to 25

 End If

Set thedispdimen = feature.GetNextDisplayDimension(thedispdimen)'get the next
dimension object

Wend

Set feature = feature.GetNextFeature()'get the next feature object

Wend

*
*
*
```

Thus, the dimension (object) named "D1" (property) had its value (property) changed. It is also possible to obtain information as well from the model:

```
*
*
*
Set swmodelext = part.extension' get extension info
Set mass = swmodelext.createmassproperty' create mass array object
mass.usesystemunits = False
vol = mass.volume'get the volume property
mprop = swmodelext.getmassproperties(1, nstatus)'get the mass properties (array)

*
*
*
```

Here the mass and volume properties of the part have been exposed. Although not completely clear from the code snippet, the result is a volume and an array of mass properties such as mass and inertia, however any information available could be returned.

Microsoft Excel API

As mentioned above, many other applications, including Microsoft Excel, utilize the API thus making it very easy to open other applications and pass data back and forth. To make it much easier to communicate with the application's object library, the user should first make a reference to the application's object library using the

Tools/References dialog [5]. In this case, this would mean activating the SolidWorks references. This will enable the object browser thus showing the student all the available objects, methods, and properties available in an application.

The same macro created inside of SolidWorks can be used inside of Excel because of the common API. In fact, the external application need not even be activated before running the macro. This means a program such as Excel can be used as a "command and control" center while other programs can be used as analytical tools, simply performing analysis and passing data to Excel while running in the background.
Continuing with the previous example, consider the macro being embedded into Excel and executed from its API. Nothing in the macro needs to be changed as Visual Basic is completely supported in both programs. To show the use of some Excel objects, it is possible to select a workbook sheet (object), then access the SolidWorks information, and write it to cells in the sheet:

```
*
*
*
Sheets("Sheet2").Select 'use the select method to select sheet 2 object

Cells(2,2)="Volume"' write the text string to cell array location 2,2

Cells(2, 3) = vol 'write the value of vol, passed from solidworks to excel macro

Cells(3,2)="Mass"'write the text string to cell array location 3,2

Cells(3, 3) = mprop(1)'write the value of mass, passes from solidworks as an array, to
excel macro to location 3,3

*
*
*
```

The block diagram below shows the flow of data to and from applications. As long as the application supports the API, data can flow back and forth seamlessly. The macro can reside in any or all the applications at the same time, or many macros can reside in the applications.

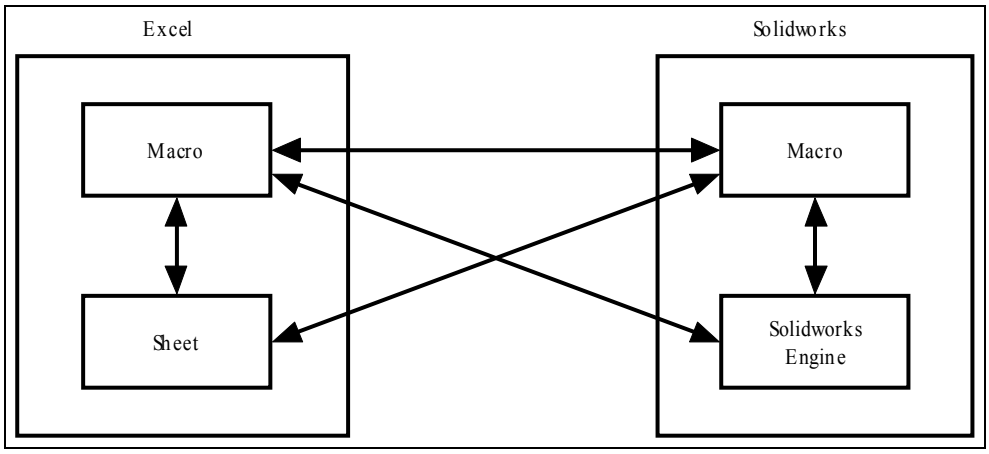

Figure 3. Block diagram showing data/command flow between various parts of different programs. Macros can reside in many different programs allowing multi-level communication.

Example

Consider the case of the simply supported beam[6], center loaded. As shown in Figure 4, a beam of length L loaded at L/2 with load F will have reactions R_1 and R_2 where

$$R_1 = R_2 = \frac{F}{2} \tag{1}$$

and will have a maximum deflection Y_{max} where

$$Y_{\max} = \frac{-FL^3}{48EI} \tag{2}$$

and a maximum bending moment M_{max} where

$$M_{\max} = \frac{FL}{4} \tag{3}$$

The maximum normal stress due to the bending moment is

$$\sigma = \frac{M_{\max} C}{I_{zz}} \tag{4}$$

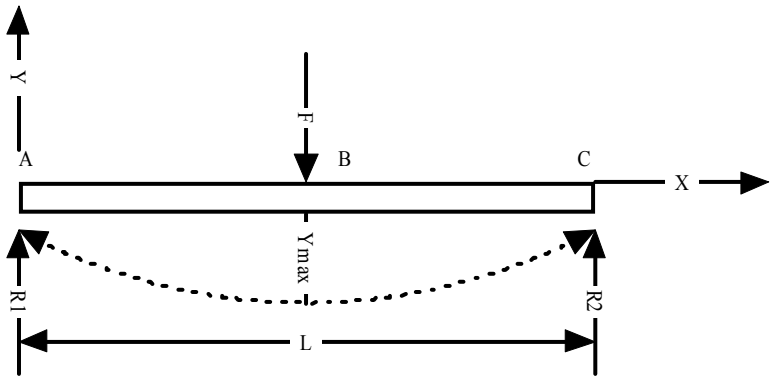

Figure 4. Free body diagram showing loads, reactions for simply supported beam.

If the beam is of circular cross section and is solid, then C is the radius of the beam. Since, for such a beam, the area moment of inertia around the Z axis is

$$I_{zz} = \frac{\pi R^4}{4} \tag{5}$$

By combining equations (3), (4), and (5)

$$\sigma = \frac{FL}{\pi R^3} \tag{6}$$

These equations seem quite simple but the point here is to demonstrate that the analytical solutions for stress and other part properties such as mass and volume compare well with the same quantities calculated by SolidWorks and COSMOSWorks for this straightforward geometry.

An Excel macro containing the above equations for stress and displacement was created. For the shaft, the following parameters were used:

- Material: 1020 Steel, E=30x10^6 psi, ρ=.29 lb_m/in^3
- Length: L=9.5 inch
- Radius: R=.5 inch
- Force: F=100 lb_f

The values for stress and maximum deflection, plus the volume and mass of the shaft, are shown as the "Exact Solution" values in Table 1.

Technique Used	**Exact Solution**	**SolidWorks**
Max Stress, ksi	2420.382	2414

Max Deflection, inch	.001214	.001294
Volume, inch^3	7.4575	7.46
Mass, lbm	2.1626	2.13

Table 1. Comparison between exact solution and SolidWorks.

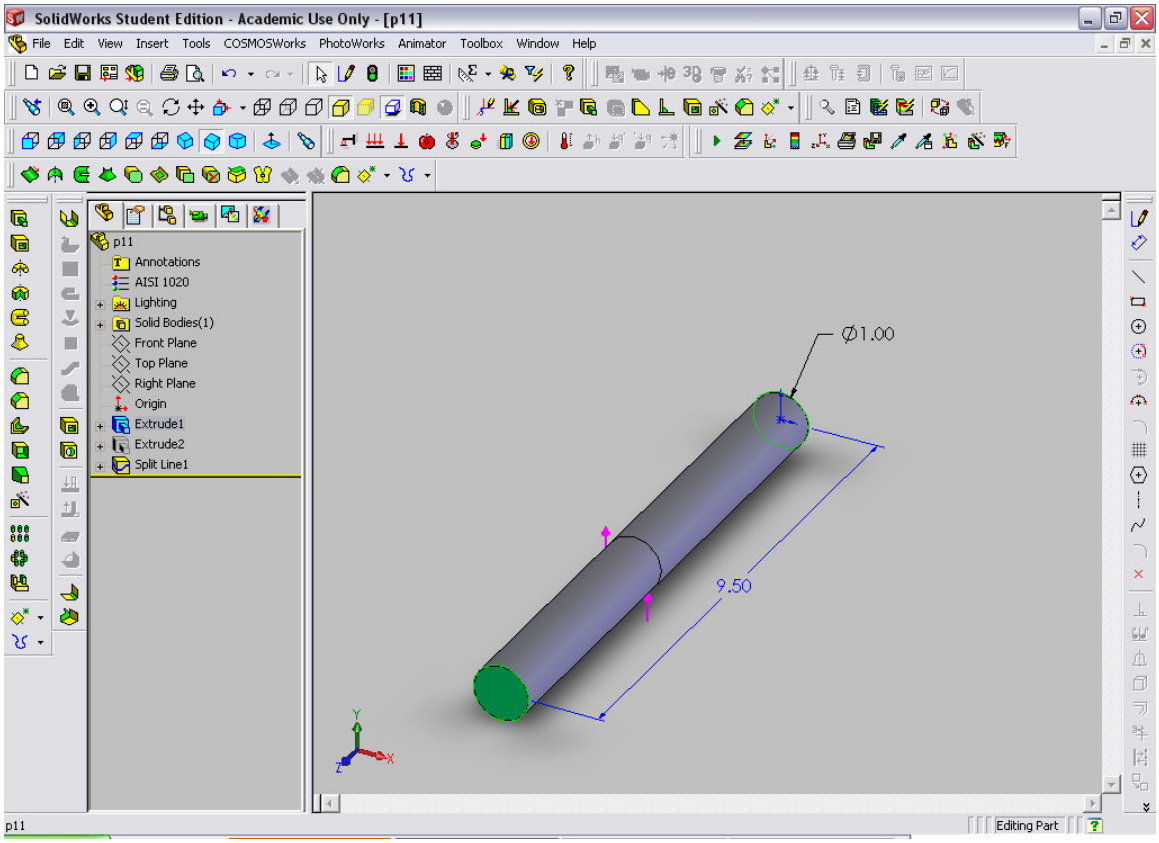

Figure 5. SolidWorks model showing GUI and solid.

The solid model of the desired shaft was also created in SolidWorks as an individual part. As shown in Figure 5, the model, as well as the feature manager and various toolbars are visible. The previously created macro also contains code that filters through the features looking for the shaft diameter, then modifies it, and passes the SolidWorks data for mass and volume back to the Excel macro. Then, a study using COSMOSWorks Stress was created using static analysis and solid tetrahedron elements . The appropriate restraints plus a 100 lb_f load were imposed on the model, the mesh was created, then analyzed[7]. The standard COSMOS graphical output is shown in Figure 6.. Code was included in the macro to pass desired output data from SolidWorks and COSMOS to the Excel worksheet. The macro was then executed, returning the appropriate parameters of volume, mass, maximum stress, and maximum displacement. As can be seen in Table 1, the results are very similar to the analytical values Data from many of the SolidWorks suite programs can be returned, including stresses and deflections from Cosmos Stress (professional version only, at this time)

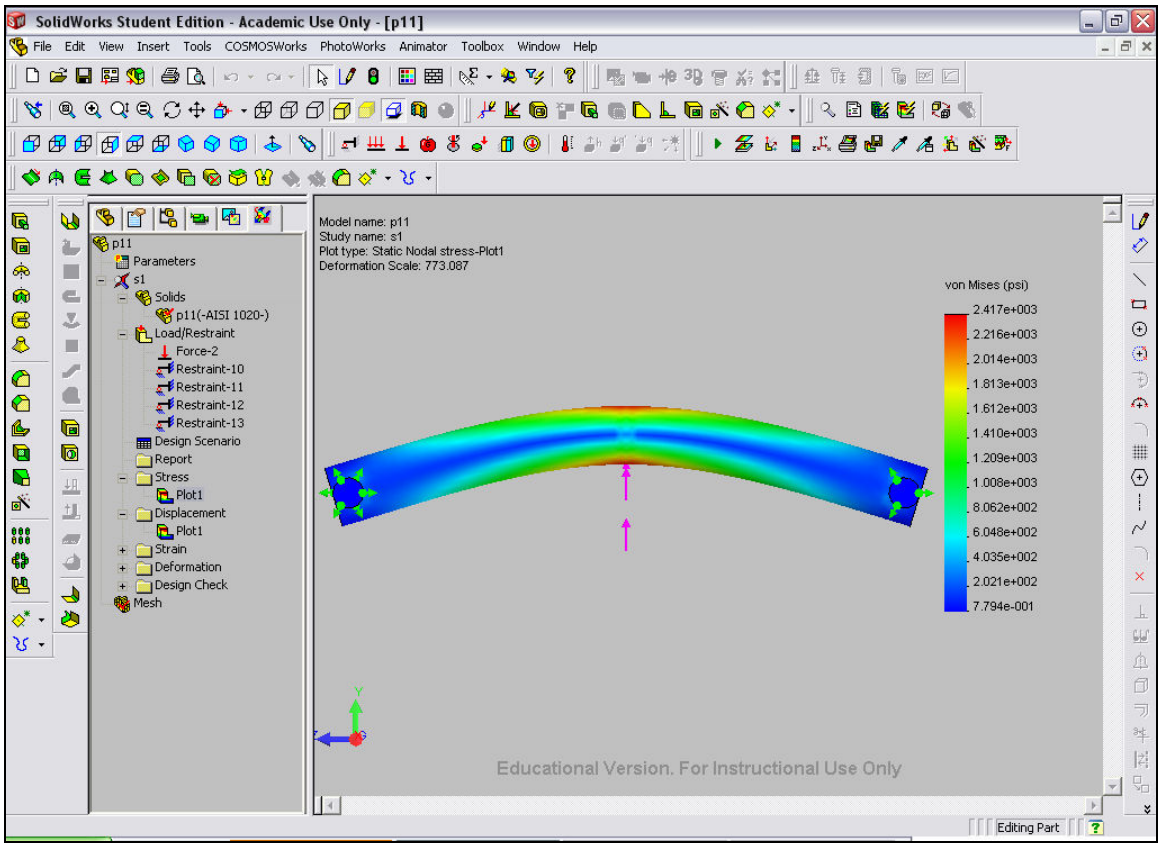

Figure 6. Solidworks stress plot. Maximum stress (Von Mises) shown on right hand bar.

The power of this approach for designing and optimizing complex parts and assemblies is clear. An Excel worksheet can be used as the design command and control center, and geometry, material properties, etc. can be changed until desired limits on weight, stress, and deflection (or other quantities, depending on the COSMOS package being used). These values for the assembly can then be incorporated into a larger system design optimization. Additional code development to fully automate this process for various types of analyses is currently being created by faculty and graduate students at CSUN.

Conclusions

Visual Basic and applications that support the API provide a valuable tool for students by (1) minimizing the number of software interfaces the student must master (2) providing a conduit by which data can be passed from application to application (3) providing an integrated computing environment for design and optimization.. By supporting the API and by providing a suite of programs with a common interface and associativity, SolidWorks has created a CAD environment which streamlines the design and analysis process. By introducing these types of tools into the curriculum early in the student's career, more effective integration between design and the theoretical fundamentals of engineering can be achieved. Although the main focus here was on interfacing SolidWorks with Excel, by no means is this the only option. Because Excel contains a rich assortment of plotting/charting/logging tools, it is often used as the command and

control center from which data is passed to and from other programs that support the API. Other programs may be better suited for specific applications. Matlab is excellent for matrix manipulation and has many add-ins that allow for easy time simulation and control analysis. Espirit is a CAM program used for wire EDM, milling, and turning. Data such as tool sheets, cutting time, and the number of cutting operations can be returned to the calling macro when embedded in Excel. This, of course, defines the beginning for complete engineering integration and optimization.

More and more application programs are beginning to support the API. A sample, although not complete, includes the following packages:

- Microsoft Excel General spreadsheet
- Microsoft Access© database
- Microsoft Word© word processing
- Solidworks Solid modeling
- Cosmos Stress Finite element analysis
- Cosmos Motion Kinematic analysis
- Espirit© Manufacturing analysis

Bibliography

Microsoft Visual Basic Programmer's Guide. Redmond, Washington. Microsoft Press, 1998.
Balena, Francesco. Programming Microsift Visual Basic 6.0. Redmond, Washington, Microsoft Press, 1999.
Solidworks Description. Online. http://www.solidworks.com/pages/products. Accessed 2004.
Mincer, Tom. Mechanical Engineering 686 Optimization Class Notes, 2004.
Mincer, Tom. Mechanical Engineering 686 ppt presentation, 2004.
Beer, F, and Johnston, R. Mechanics of Materials. McGraw-Hill, 1991.
Cosmos Training Manual, Vol. 1. Los Angeles, CA. Structural Research and Analysis Corporation, 2002.