

## "Reverse Engineering" Your Pro/ENGINEER Models for Top-Down Design

By Jason Clark, OceanWorks International, Inc.

[Table of Contents](#)

Pa

Why would you want to "reverse engineer" your Pro/ENGINEER model? Simply because the first model you make isn't the final one. After all, there's always something to tweak whether it be for clearance, machining allowance or some other refinement.

The process of adding top-down design functionality to existing models is actually part of the top-down design (TDD) process. Quite often you will brainstorm an idea with little or no constraints. Once you are happy with the overall concept, you will then add intelligence to the model by implementing the TDD functionality. You can't think of all the variables at the start of the design, so you take an educated guess and progress. The idea of reverse engineering your models is to apply what you learn through the design phase to add intelligence to your models for final production.

You can download the models referenced in this article from our web site.

[Click here to begin your download...](#)

When people ask why develop models that adhere to the TDD philosophy, the easiest answer is "because Pro/ENGINEER is designed for it." In truth, Pro/ENGINEER has very powerful tools to administer and control your designs. Utilizing TDD practices offers many advantages including the ability to:

- Administer changes to many models from just one "document" (called a layout)
- Control the flow of changes, and the effect of the changes, over many models
- Rough out designs and create concepts without designing any detailed parts
- Easily generate spin-offs from an existing design
- Empower your Pro/ENGINEER assemblies to adapt to change, including part replacement.

### Our Objective

The example used throughout this article is a hydraulic/pneumatic cylinder that begins as a typical Pro/ENGINEER assembly and is reverse-engineered to leverage the TDD principles (see Figures 1 and 2). By "typical" assembly, I am referring to the bottom-up approach in which a Pro/ENGINEER user models each piece of an assembly and manually compares values to ensure their validity. Models of this type contain minimal intelligence and thus do not automatically adjust to dimensional changes or even model replacement.

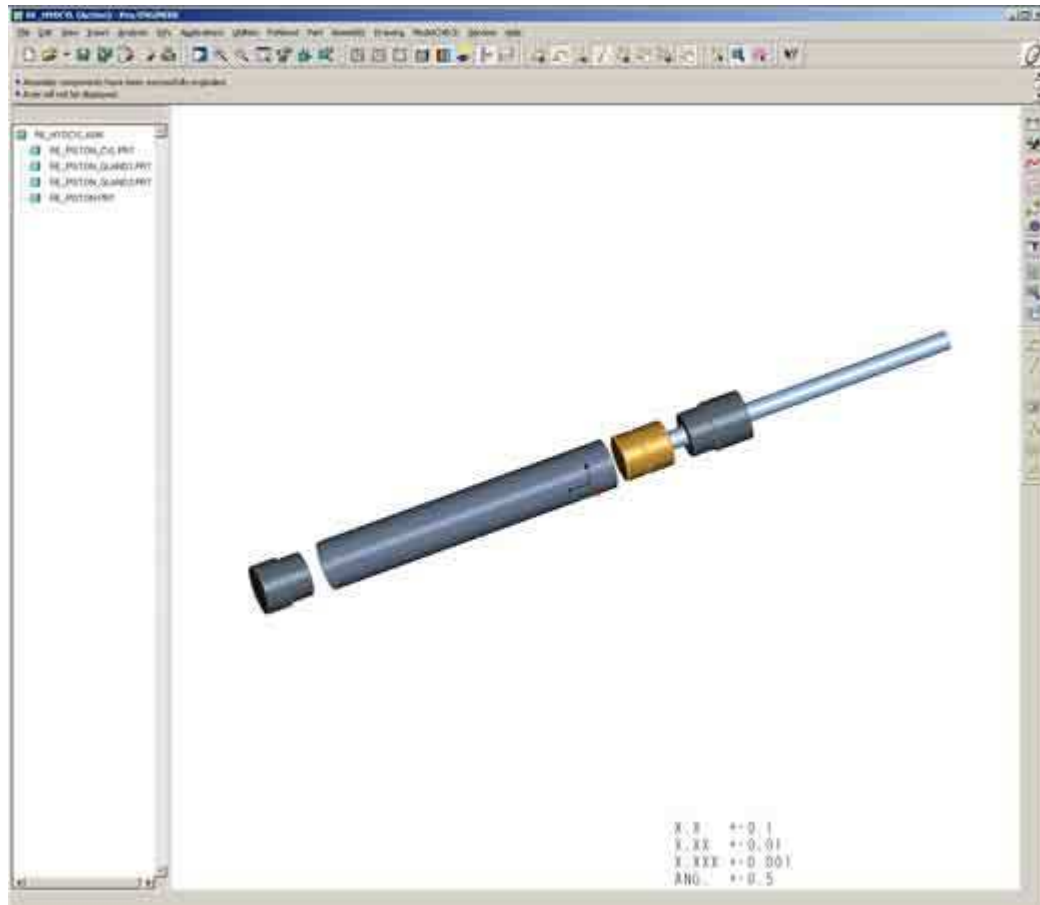


Figure 1. Exploded assembly

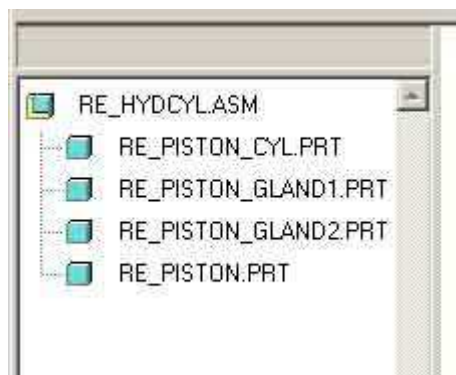


Figure 2: Assembly model tree

After reading this article, you should be able to add TDD features in one form or another to enhance legacy Pro/ENGINEER models. The following discussion assumes you have a fairly good understanding of the Pro/ENGINEER software interface, as well as a basic concept of what TDD is and how it functions within Pro/ENGINEER.

### 1. Start with a Blueprint

Before we progress too far, we need to ask a few questions to ensure we attain the goals we want our Pro/ENGINEER model to achieve. Applying TDD does require some forethought and planning. I say this because once you start, you can get carried away and create too many parametric links. This makes your models hard to work with in the future—especially if you do not document them carefully. Even though top-down design does take upfront effort, you will find that as you develop best practices applying TDD will become second nature. If you ever have to modify a TDD drastically, you will find that you can complete changes faster than if you were working with an assembly with no TDD methodology.

So let's create some assumptions and goals for our cylinder assembly. It must be able to:

- a. Adapt to the piston bore size.
- b. Adapt with a change in stroke length.
- c. Adapt with a change in shaft diameter.

Albeit a small step for this particular exercise, this kind of planning is crucial for any design you try to accomplish in Pro/ENGINEER.

## 2. Sherlock Time!

Now assume we have no prior knowledge of the Pro/ENGINEER models we are to work with. Perhaps you are a contractor continuing someone else's design or perhaps you are tweaking your model for production. Since we are dealing with an existing design, a lot of work has already been done and we do not want to contradict the design intent.

The easiest way to learn the assembly intent is to browse the assembly and its components to make note of constraints. Investigate the assembly and become familiar with it. Later in the process, the goal will be to reroute the assembly constraints to a skeleton.

Browse the constraints in Figures 3-6 using Info\_Component:



Figure 3.



Figure 4.

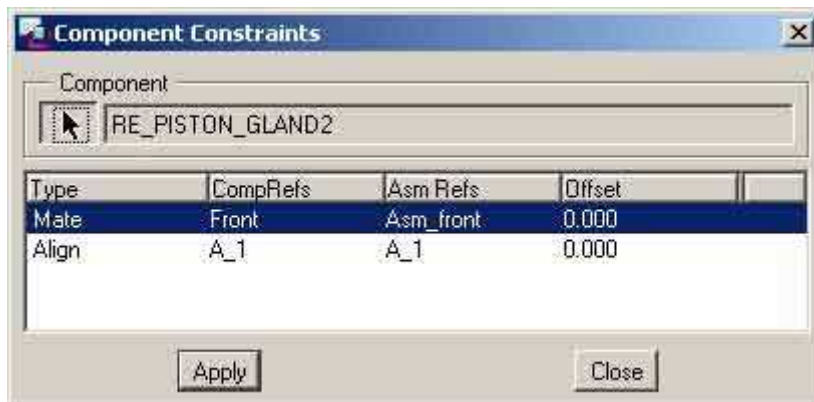


Figure 5.



Figure 6.

### 3. Lay It Out!

At this point, we know what our goals are for controlling the assembly and have determined the method of assembly for each component. The next step is to document the master plan—often a weakness in parametric modeling. Documentation is the key with TDD since it is almost as powerful as actually implementing the parameters and relations to control the design.

To document the intent of the design, we use a layout. A layout is a two-dimensional representation of our design that may contain static sketches and tables containing parameter data. While sketch data can be made within layout mode, I recommend importing 2D views whenever possible. Sketching in Pro/ENGINEER's layout mode is weak and can be time-consuming.

If you are dealing with a design that is already documented, export the drawing as an iges file and import it into your layout. Once you have imported the data, you can delete what is not needed. Other options are to import dxf, dwg and Pro/ENGINEER overlays. (Imported iges, dxf, and dwg data can be edited within layout, but Pro/ENGINEER overlays cannot.)

Figure 7 shows our layout, which is an imported IGES file (from an existing drawing). Refer to the notes below the figure.

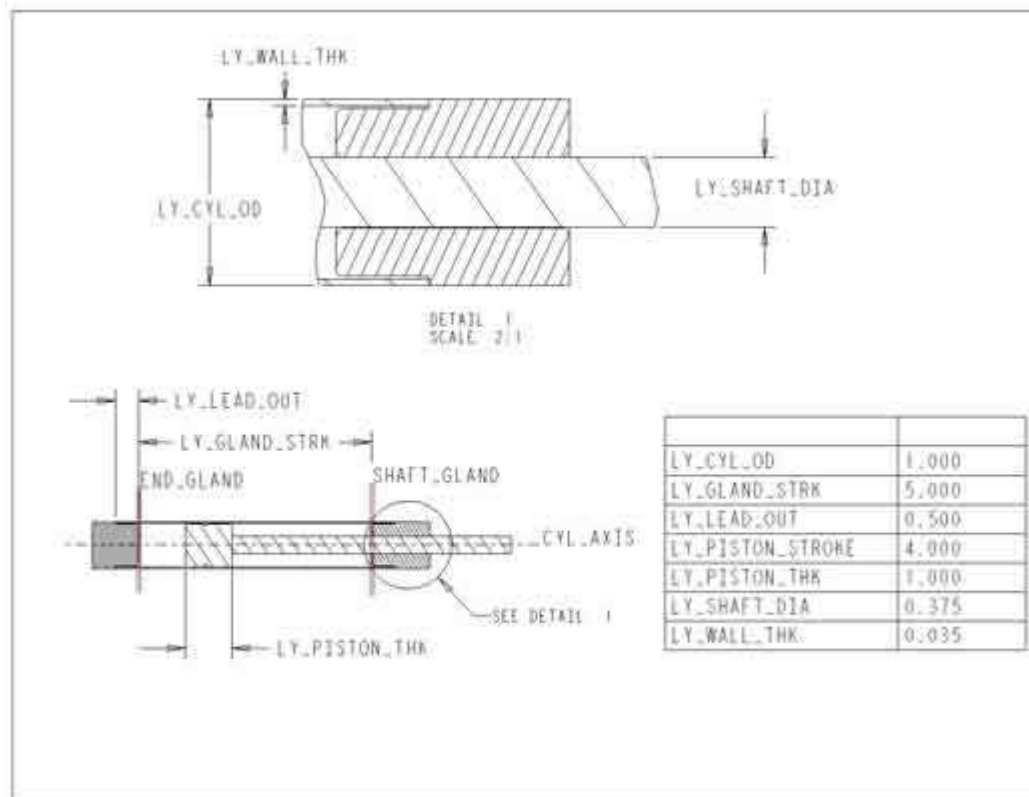


Figure 7. Layout view (click to enlarge)

Bear in mind there are many ways to go about controlling a design, and what I have illustrated here is a simplified interpretation. What you see in the layout is a sketch and enough dimensions so that we can control the key aspects of the

cylinder. By creating a dimension in layout (Edit\_Insert Dimension), you in fact create a parameter because, when prompted, you need to enter a parameter name and give a value

Once the dimensions are done, I recommend creating a table using a repeat region. For our example, I created a two-column table and used the report parameters lay.param.name and lay.param.value in the repeat region. I then added a filter to show only layout design parameters (i.e., the LY\_ prefixed ones).

Now it's time to add relations. For now, the parameter LY\_PISTON\_STROKE is controlled by a relation LY\_PISTON\_STROKE=LY\_GLAND\_STRK – LY\_PISTON\_THK. Alternatively, you could write a relation to control the stroke with LY\_PISTON\_STROKE value and a relation would update LY\_GLAND\_STRK and LY\_PISTON\_THK. There are many ways to go about this, and ultimately the choice is yours.

The important areas of a layout are:

- Parameters to contain numerical data or textual data (e.g., notes, specifications)
- Relations to create intelligence and to generate rules for the dimensions (e.g., shaft\_bore=<cylinder\_bore, shaft\_dia=piston\_dia-4)
- 2D Sketch to provide a simple two-dimensional representation
- Draft datums to show assembly intent.

A few key things about layouts to keep in mind:

- a. The sketches are not parametric and are best imported from an existing file such as dwg, dxf or iges. I recommend the import if you've done some preliminary sketching (e.g., using AutoCAD), or the export of detail drawings you've done using the iges format. The sketching tools in layout are weak, but you can get by with them if necessary.
- b. Naming convention is a definite consideration that requires some thought. I suggest:
  - Prefixing layout parameter names so they are easy to distinguish in your parts and assemblies. This practice protects you from parameter conflicts upon declaring your layout. These conflicts generally happen when you use a PDM tool like Pro/INTRALINK or if you use generic names that may appear in parts. You need to be careful here because you can overwrite parameters in parts and assemblies that have the same name in a layout.
  - Defining your datum name scheme since the names will be used throughout your parts and assemblies.
- c. Add datums to your layout (Edit\_Draft Datums). This step is very important because it documents the design assembly constraint intent and allows you to use layouts to drive automatic component replacement if desired.

#### 4. Skeletons Out of the Closet

For TDD, skeletons are a fabulous—and essential—tool. Skeletons are the three-dimensional representations of our layouts. The links from layouts to skeletons are the parameters, relations and datums we define in the layout. For visual purposes, I find it handy to add surface geometry. To illustrate, imagine a large assembly. Adding surfaces can be a lifesaver because you can define work area boundaries that represent a physical volume of a subassembly (for example, a surfaced rectangle could represent a drivetrain). The surfaced rectangle acts as the physical boundary of a subassembly and provides a visual indicator for other designers to recognize items that infringe on the boundary. This helps to resolve interferences immediately.

To prepare the skeleton, open the assembly and select `Component_Create_Skeleton` from the side menu. You should specify a start part if you utilize start parts (i.e., templates). Be sure to name the skeleton `<assembly name>_SKEL`. Once your skeleton is defined in the assembly, save the assembly and open the skeleton in a new window.

The next step is to lay down your foundation. In Figure 8, note the naming of the datums since those datums within the layout will need to be represented within the skeleton. I recommend labeling key skeleton datums with a prefix (in this case, `SK_`). Naming datums and prefixing them can be invaluable when you have many datums. Naming datums also helps when using `Sel by Menu` operations.

Feel free to add motion controllers, such as the datum `SK_MOVE_PISTON`. We will add a relation to limit movement of `SK_MOVE_PISTON` based on values in our layout.

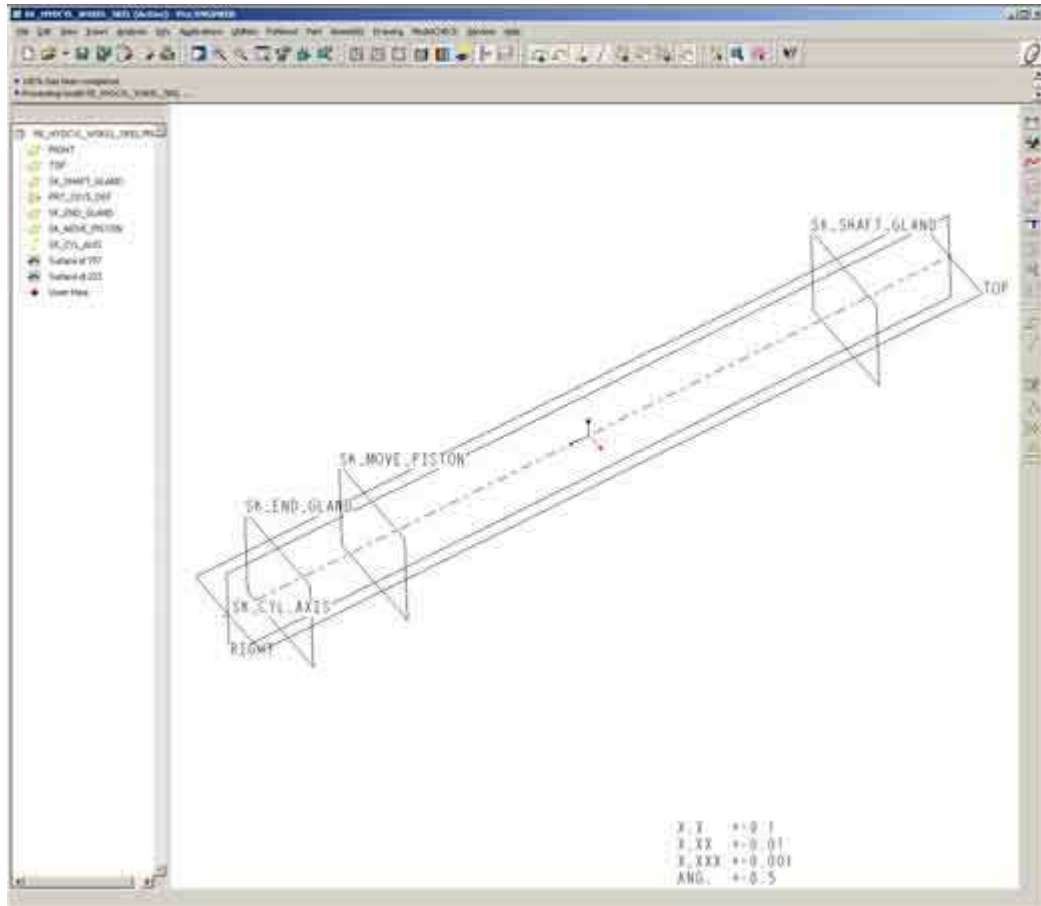


Figure 8. (click to enlarge)

To fill things in a bit, add some surface geometry to help with visualization. Note in Figure 9 that the piston is roughed in.

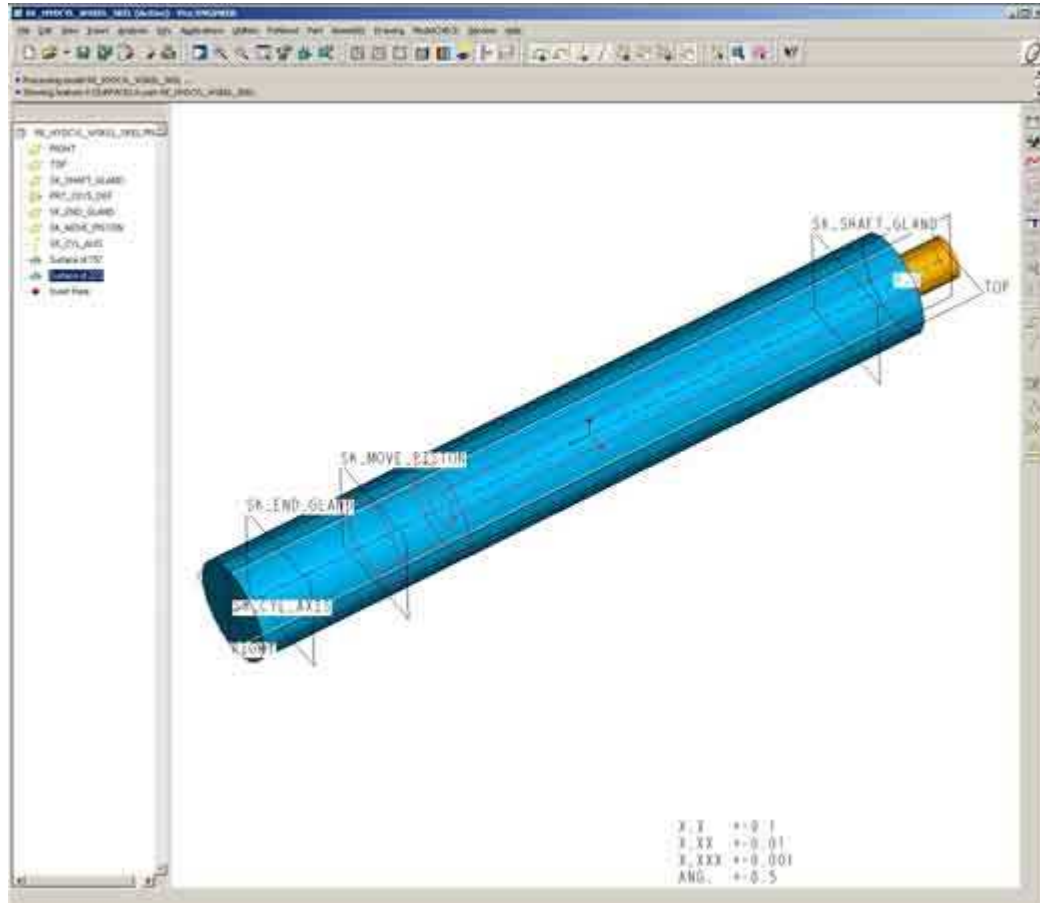


Figure 9. (click to enlarge)

Now that we have geometry, we need to associate the values from layout to skeleton. First off, we have to declare the layout to our skeleton. Open the layout created earlier into session and then activate the skeleton window. Select Set Up\_Declare\_Declare Lay and then select the layout name from the menu. You can only declare layouts that are in session.

Open up your relation window and add the relations to control the skeleton's geometry. Adding relations depends heavily on the design intent and is one tool to capture that intent. Be sure to comment your relations! (See Figure 10 for an example.)

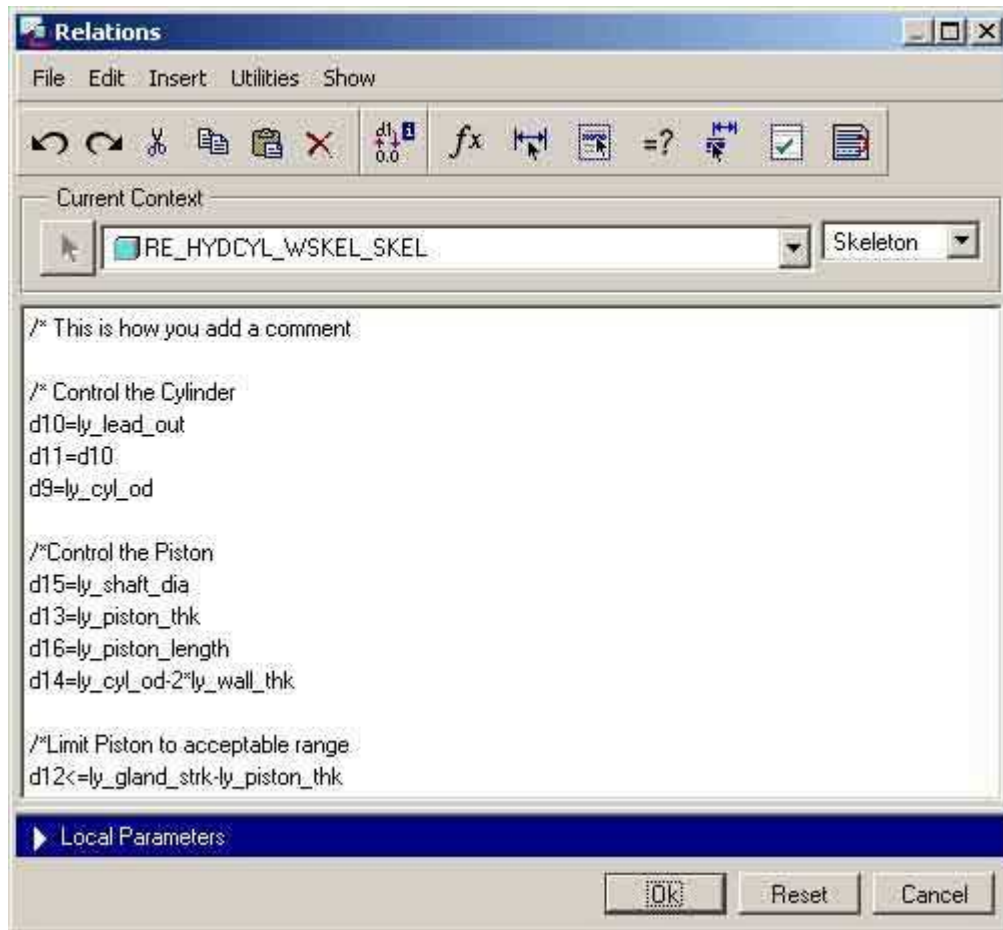


Figure 10.

At this point, we can use the skeleton to look at how the geometry can be manipulated and in doing so we can do case studies, test geometry movement, etc. For example, modify the datum SK\_MOVE\_PISTON and give it a value like 20. Regenerate the model and see what happens.

### Putting It All Together

Let's open up our assembly that contains our super-smart skeleton.



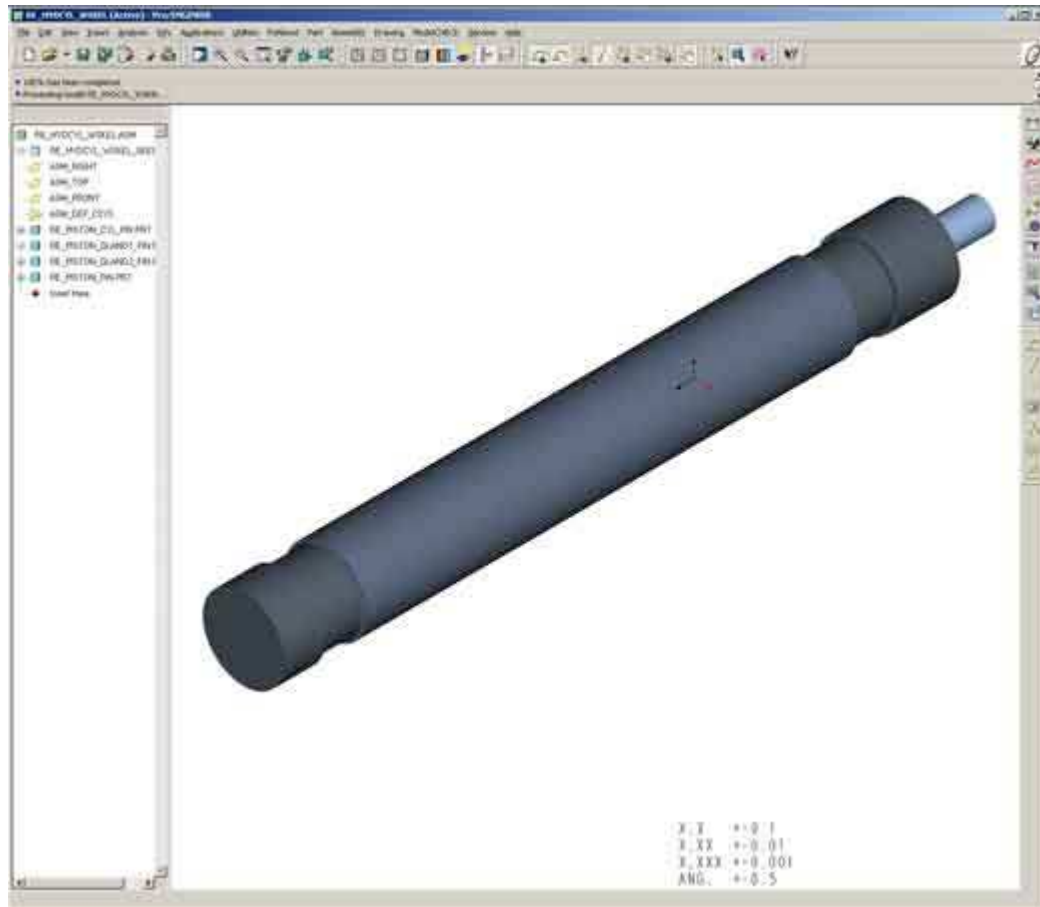


Figure 12. (click to enlarge)

What is going on here? In Figure 12, I redefined the components to match the constraints defined in my layout so it should look like my layout! Well, not quite. The last step is to pass intelligence to the parts that make up the assembly. We can now either use copy geometry from the skeleton to define boundaries and use those features to fix the parts, or we can apply our layout to the parts.

I will illustrate the second approach. Using copy geometry is a quick fix, but it will build a dependency to the assembly. If your assembly is large, making a change that results in a copy geometry feature requiring regeneration means you have to load the entire assembly into session to complete the regeneration. Declaring a layout to our parts eliminates this step. By using a layout, you can work on a part with only it and a layout open in session, saving all that regeneration time for other things!

For example, to fix the short cylinder tube, we can modify the tube to contain the extra dimensional information and add a relation to keep things in order. When complete, our cylinder should look like Figure 13.

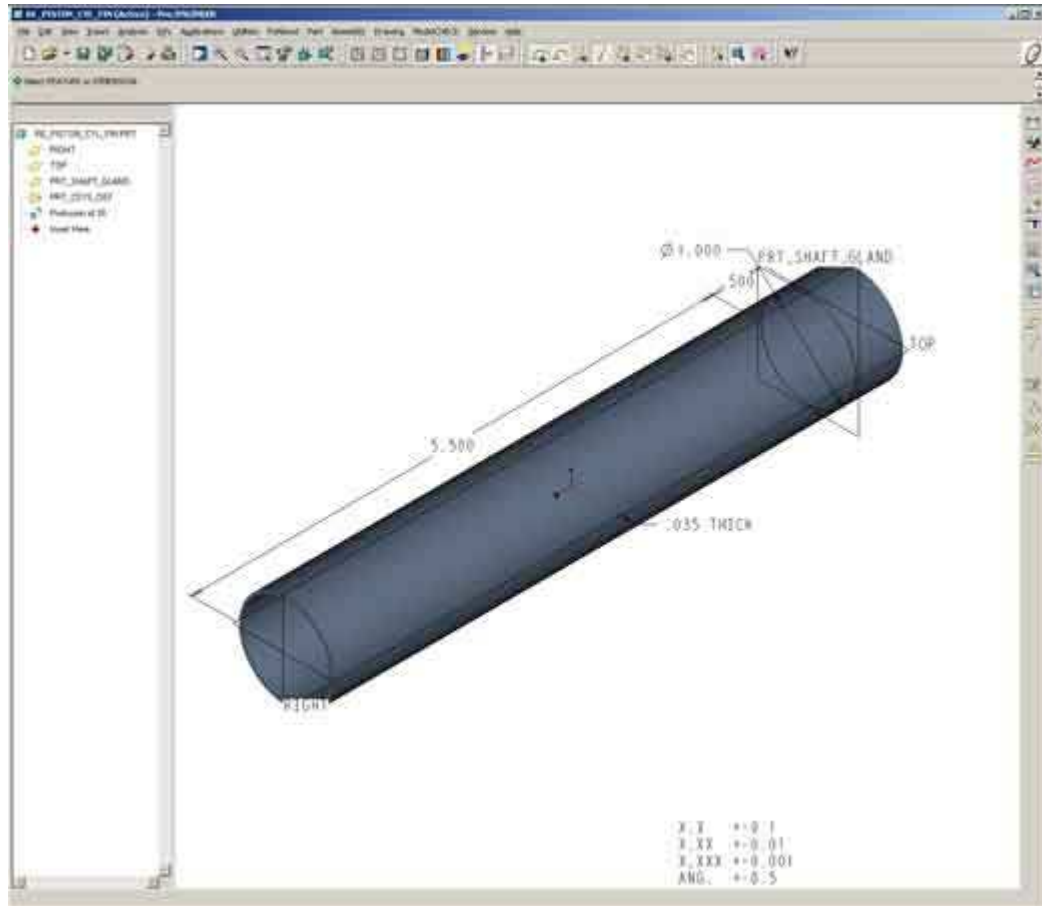


Figure 13. (click to enlarge)

To accomplish the geometry, I declared the layout (Declare\_Declare Lay, in part mode) and added the following relations (Figure 14).

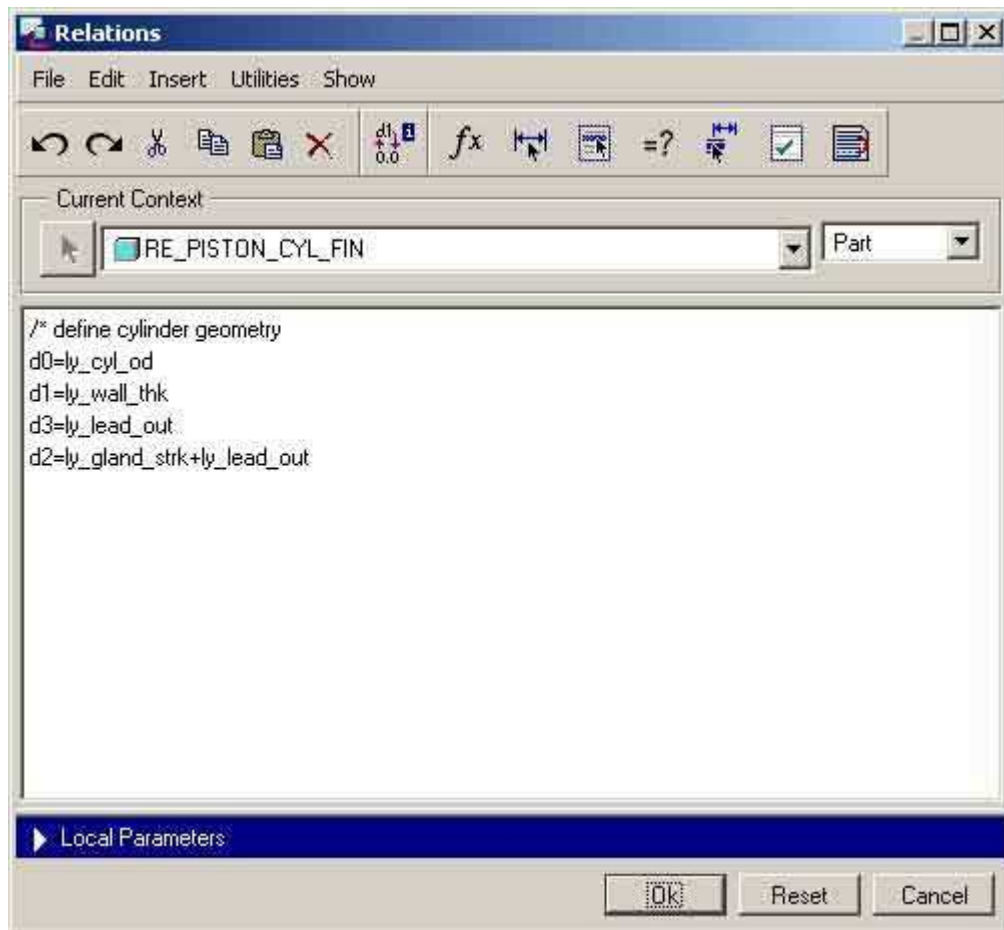


Figure 14.

I then modified the piston part by declaring the layout and adding some relations to ensure clearances (Figure 15).

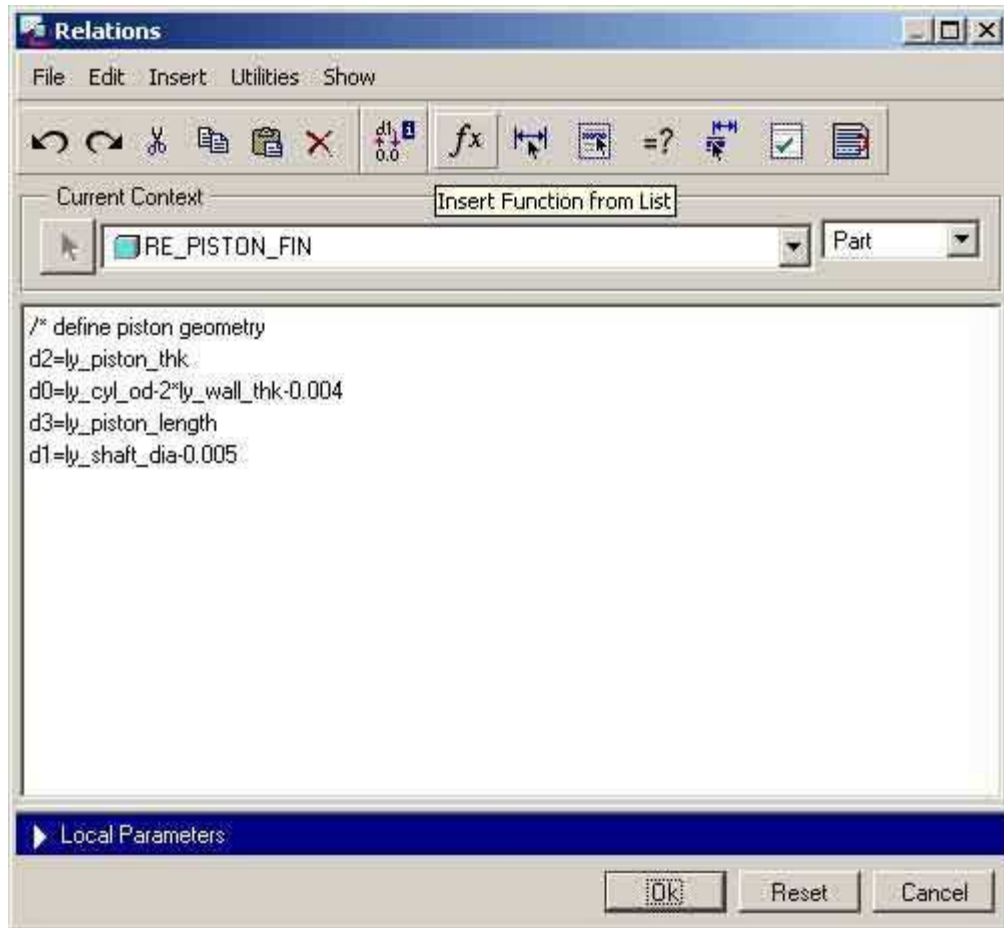


Figure 15.

Just to confirm we're going in the right direction, we can compare our cylinder to our skeleton with a simple shade test. The blue and yellow represent the cylinder tube and piston in the skeleton. As you can see, we have a perfect fit (Figure 16).

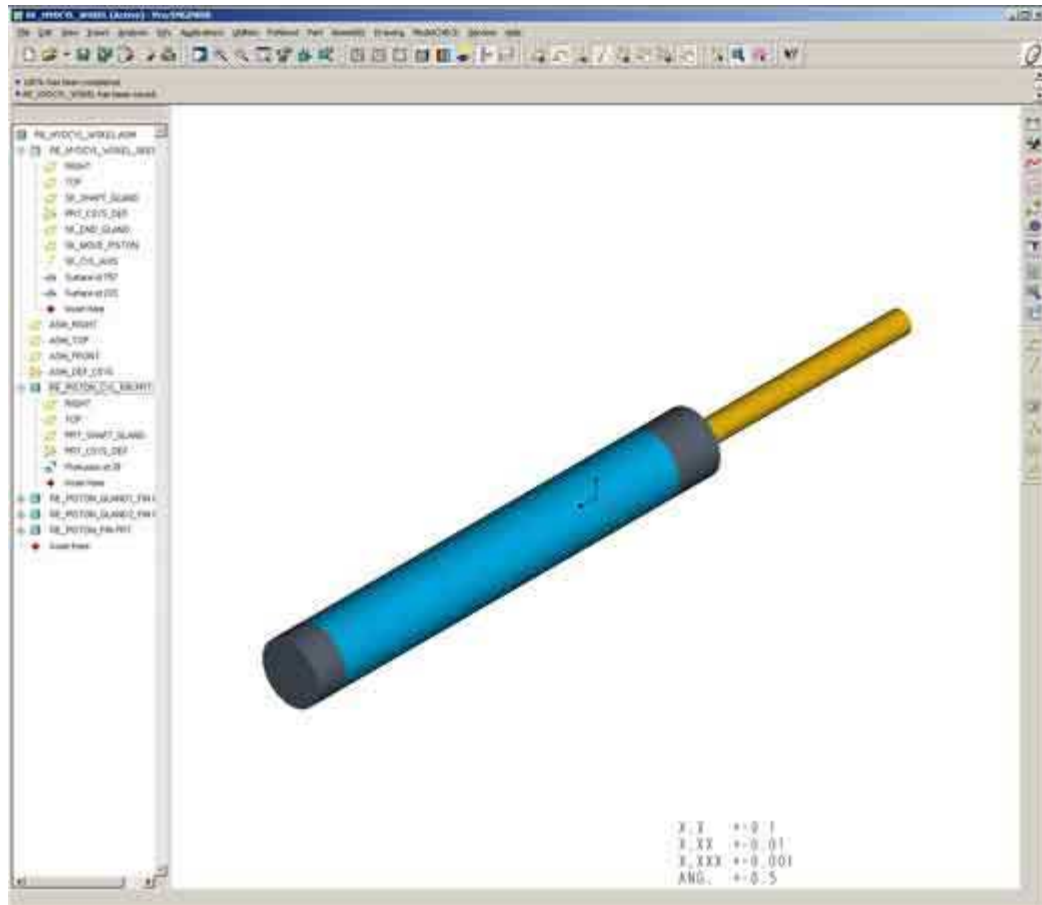


Figure 16. (click to enlarge)

From here, you can declare the layout to the other parts and associate the values from the layout to their respective dimensions in each part. Once we have completed the associations, we then fulfill our plan to:

- a. **Adapt to bore size.** By varying the LY\_WALL\_THK parameter, we can change the cylinder internal bore, which in turn automatically adjusts the piston diameter.
- b. **Adapt to stroke length change.** By varying the LY\_GLAND\_STRK parameter, we can lengthen the cylinder, which in turn automatically updates the parts placement since the skeleton will update its datum offsets. If we were smart, we would have added a relation to our piston part to associate its length to the stroke.
- c. **Adapt to shaft diameter.** By varying the LY\_SHAFT\_DIA parameter, we can adapt the shaft end gland bore to accommodate a shaft size change.

## Conclusion

I have covered a lot of basics to help you start to integrate some top-down design methodologies into your older models. By taking two simple steps with a layout and a skeleton, you almost immediately add greater control and stability to your Pro/ENGINEER assemblies. Many other tools complement the top-down design

methodology—such as copy geometry, replace by layout, shrinkwraps and envelopes—but are outside the scope of this article. I encourage you to review these and other options for leveraging all that Pro/ENGINEER has to offer. ♦

Jason Clark is a senior designer and PTC application administrator at OceanWorks International. Jason also chairs the Vancouver PTC/USER group. He can be reached by e-mail at [jclark@oceanworks.cc](mailto:jclark@oceanworks.cc).