

## Top-Down Design—A Method for Pro/ENGINEER Users

by Jason Clark, OceanWorks International

[Table of Contents](#)

Pa

A lot of people assume that top-down design (TDD) is too cumbersome and time-consuming to use. But these same people spend a lot of time tweaking models and blame the computer for faults. At the end of a design cycle, though, you can bet that systems built with top-down design will have required less time and end up being more robust.

The following tips build upon the information presented in my previous Pro/files article, "Reverse Engineering." For those of you who didn't read that story, I've included a summary of Pro/ENGINEER's TDD toolset in addition to guidelines for planning and implementing a top-down approach.

### Parametric Autonomy

Pro/ENGINEER is a parametric modeler at its core. The price you pay for this power, however, is that it creates parent/child relationships. With the toolset Pro/ENGINEER provides, we can easily weave ourselves into a mess of circular references that overload our hardware as well as our own patience.

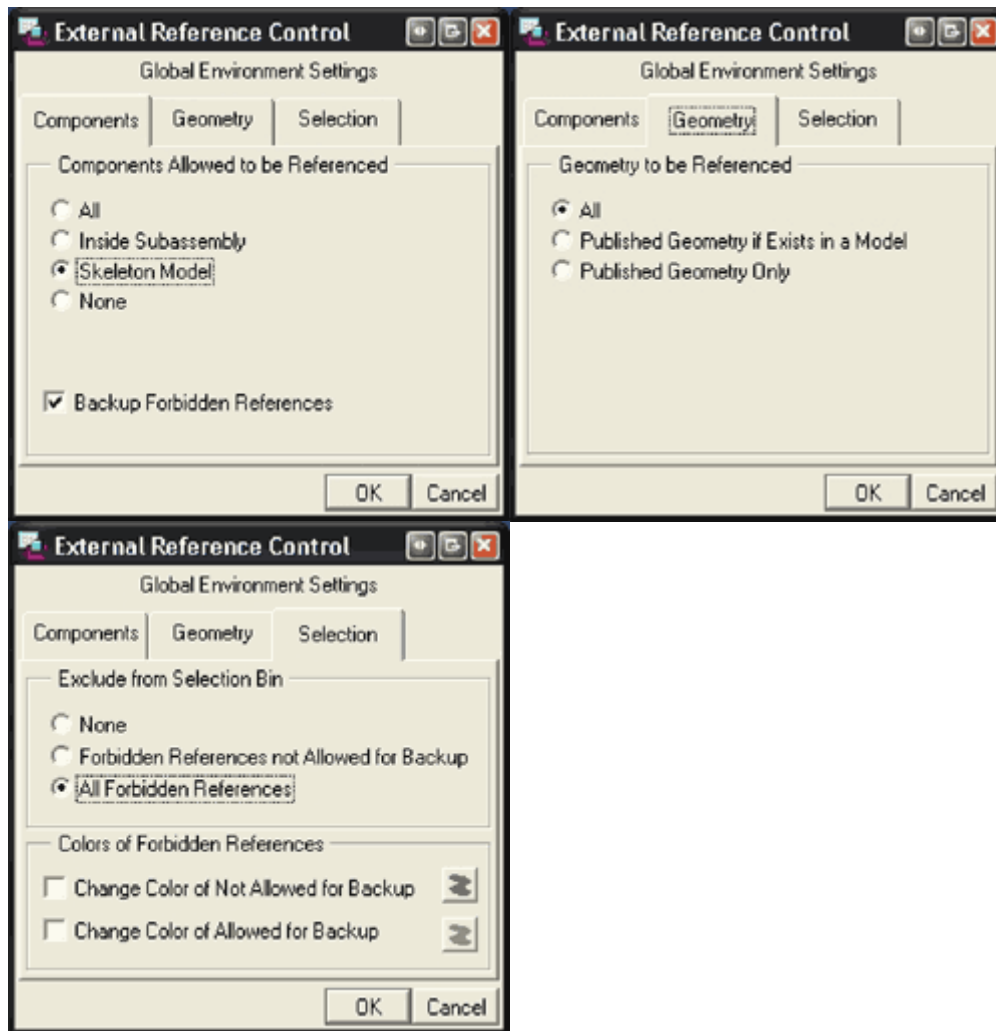
Parametric autonomy—perhaps the Holy Grail of top-down design—is a method to gain parametrics across far-reaching models from a single interface while at the same time minimizing parent/child relationships. Although it requires a little upfront planning, and sometimes duplicate work of creating geometry and relations, parametric autonomy offers some important advantages:

- huge reductions in time spent tweaking data in several models
- rapid alteration of models and design intent from a minimum of input points
- documentation of design intent, and
- more efficient use of computer hardware.

### Overview of Top-Down Design Tools

Here's a quick summary of the TDD tools that Pro/ENGINEER currently offers.

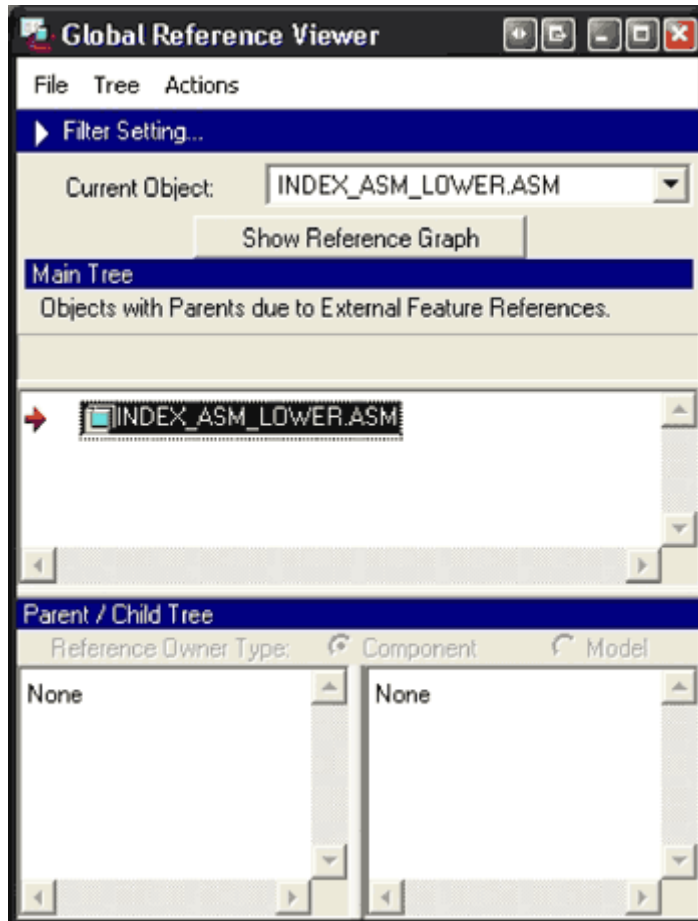
**External Reference Control.** Available through the Utilities menu, this tool is really handy for protecting your models from inheriting references from unauthorized locations. Referencing from skeletons is best because the skeleton models are the foundation of your design and are controlled against change. You can also choose settings that allow only published geometry to be selected, and color or hide forbidden references.



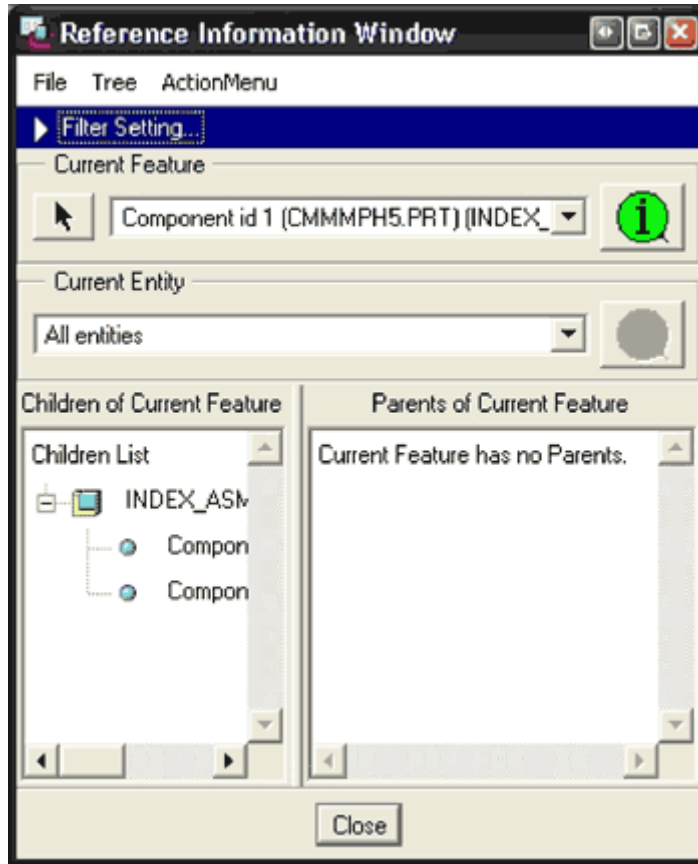
Most favorable settings for External Reference Control.

If you're using "mirror" to create parts or assemblies, you must change your scope setting from *skeleton\_model* to *all*.

**Global Reference Viewer.** The global reference viewer (GRV), the granddaddy of the top-down design tools, tells you where your *external* references reside. If you've made copy geometry features, the GRV will let you see them.



**Info: Parent/Child.** Similar to the global reference viewer, this tool shows you any parent/child relationships. You can access Info:Parent/Child using either the right-click option on a feature or the *Info* top-menu.



***Skeleton Models.*** Skeleton models provide the foundation for our parametric autonomy paradigm. The function of the skeleton model is to:

- contain *key* assembly constraints
- act as a conduit for “sketching” a design by adding geometry to complement a layout
- serve as the hub of data sharing
- allow assembled components to interact with the parent assembly, insulating it from failure of component placement.

***Layouts.*** Layouts generate the key input data for skeletons, models, assemblies, and relations. Using a static 2D sketch along with parameter tables, we can communicate design intent and produce a history of a design’s evolution. A layout immediately reveals what the important datums are and what the design’s key dimensional constraints are. You typically use layouts from large assemblies down to a medium-size subassembly. Occasionally, interface layouts are used to control bolt circle information or connection information that may not be suitable to document on a subassembly layout. (More about layouts later.)

***Data Sharing.*** Pro/ENGINEER’s copy geometry toolset is ultimately its Achilles heel. Sloppy use of copy geometry yields circular references and a terrible mess of parent/child relationships that produce a model that’s slow to regenerate. As a result, it’s better to use published geometry, which lets you “gather” references that can then be available for use outside the host model.

Using published geometry forces you to manage and control your copy geometry references. Think about a team environment. If a team leader is managing the assembly, it is his or her responsibility to control the references and the general constraints of the assembly. By using published geometry, a team leader can therefore decide which references are valid and publish them for use, thus minimizing the problem of bad references. (Recall the External Reference Control option of “published geometry only.”)

**Important!** If you make copy geometry “independent,” do *not* use copy geometry. This rule is paramount. Once you make copy geometry independent, the management and tracing of errors becomes extremely time-consuming.

**Packaging Components.** Packaging is a great way to rough in component placement as you sort out your design. A good technique is to rough in your component placement (using packaged components), modify your skeleton, sketch in datum geometry (without referencing the components), and then redefine your packaged components to reference the skeleton.

**Relations.** Relations are the way to link the data input values from your layouts to your skeleton, part and assembly features. Using logic operations and condition statements, you can create geometry that can adapt under certain conditions.

**ModelCHECK.** While configuring ModelCHECK is outside the scope of this article, I do want to mention that this tool goes a long way toward helping to develop and maintain consistency in your model information. ModelCHECK will warn you about copy geometry references, external references, layer configurations and parameters. If you can set up a sound configuration of models, parameters, and such, ModelCHECK helps you maintain the consistency as well as future enhancements you want to incorporate (e.g., new parameters or renaming parameters).

### Planning and Conceptualization

During the planning and conceptualization phase, we rough in our assembly and create our skeletons and layouts. Unfortunately, this is the stage that overwhelms most users new to TDD and the one skeptics complain takes too long. Once you have a procedure everyone follows and then practices, however, I don’t find that using TDD upfront adds much time to the finished design. In fact, planning is the most important step because we want to form—and document—our design intent. While all CAD systems lack good tools for this documentation process, Pro/ENGINEER makes up for this deficiency somewhat by integrating layouts and skeletons.

**Home Coordinate System.** Your “home” coordinate system is the one that’s common to all of the assemblies and subassemblies in your drawing tree. For example, if you were designing a car, you’d have assemblies for body, engine, and interior. These are major assemblies that relate to one another, so they would share a common coordinate system.

**Layouts.** Layouts let you sketch out your 2D representation. I prefer to use imported AutoCAD data or IGES data, but in a pinch you can do some rudimentary sketching in Pro/ENGINEER. In general, you should have one layout for every major assembly and possibly another one for each level below. Remember, the idea is to capture layout data that is specific to the assembly you're working on.

#### *Tips for Developing Layouts*

- *Place all interface datums in the layout using proper naming conventions.* Be aware that the yellow side of the datum gets created by the direction in which you pick your points when placing draft datums, so practice and get used to it. The datum direction is only important if you plan to use layouts for *automatic* placement. Identify your home coordinate system if applicable.
- *Keep a table showing parameters pertaining only to the design and not to the layout itself* (e.g., naming/part numbers for layout tracking). To facilitate this, I use a naming convention to filter out layout-related parameters, typically prefixed by "LY\_". Be sure to include a comment column to describe the parameters' usage. You may want to create a standard table that you can retrieve, thus creating a standard look.
- **FOCUS!** I cannot overemphasize the importance of keeping your layouts focused. This is particularly vital because you can declare multiple layouts to assemblies and parts. For example, don't put parameters for bolt circles on a layout that is for placing subsystems for the top-level assembly. To keep your layouts simple, you may have to make more than one.
- *Avoid declaring layouts to each other.* Otherwise, you can end up with a mess of parameters declared to parts that are not required.
- *Use meaningful names for your key datums.* The terms you use for your base datums will be the same as those for your parts, skeletons, and sometimes assemblies.

And remember: the layout is an excellent place to store design notes and document your design intent.

**Start Parts and Layering.** Before creating any geometry, it's best to plan out what information will be required from the parts—typically in the form of parameters. If you already have start parts or templates, you're probably all set but you might read through this section to be sure you've covered your bases.

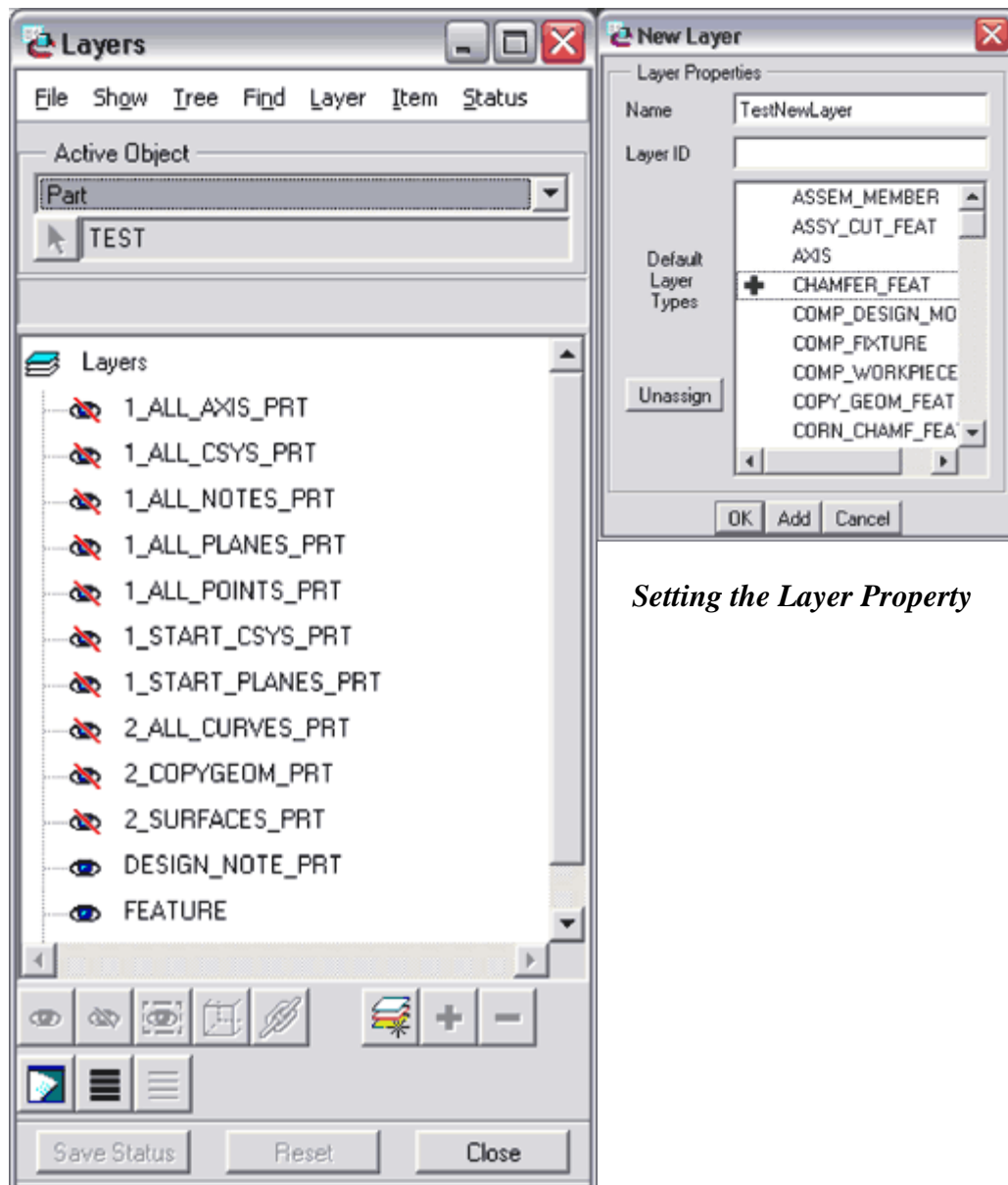
Start parts are great simply because you can define a layering rule, default parameters, default relations, and so forth. Creating your defaults in your start parts also eliminates the need to do this with config.pro options, which never really worked well. When you create a start part, it's a good idea to use a base start part with features common to all parts. I keep a start part for skeletons,

pipng parts, sheetmetal parts, and machined parts.

### *Tips for Using Start Parts*

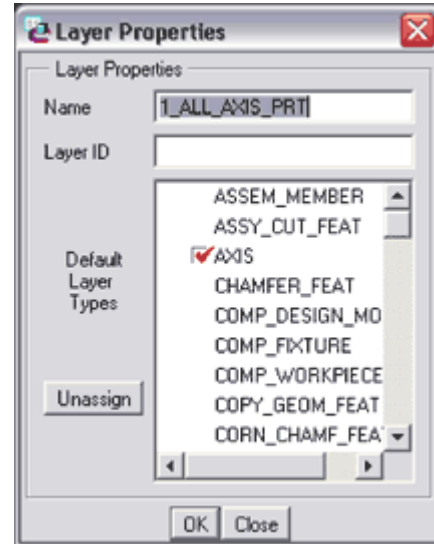
Default layer items are best served by associating entity items to be placed automatically on layers. You can set your layer properties either during creation by using the “new layer” button or by right-clicking the layer and selecting Layer Properties. When you name your layers, include a digit in the prefix so you can organize the layers in sequence.

The “isolate” function in the layer box is ideal for putting your assembly datums on a layer. Isolating a layer in effect tells Pro/ENGINEER to ignore the status of any items in the layer. This means that even if the items are on other blanked layers, Pro/ENGINEER will still show them.



***Setting the Layer Property***

*Sample Layer Scheme*



*Modifying Layer Properties*

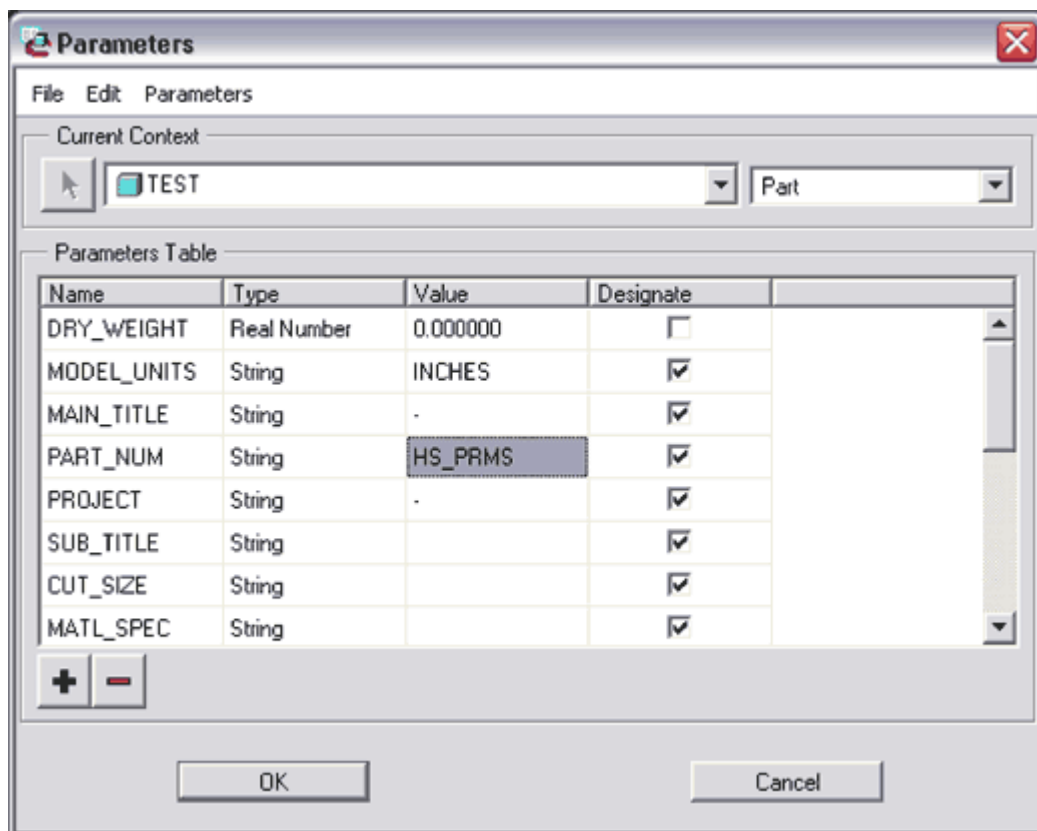
**Recommended Configuration Options**

Below I have listed the recommended options for your config.pro and/or config.sup files. Use the config.sup to prevent user independent config.pro settings. (You will need to consult the PTC documentation because some config.pro options do not work properly when placed in the config.sup.)

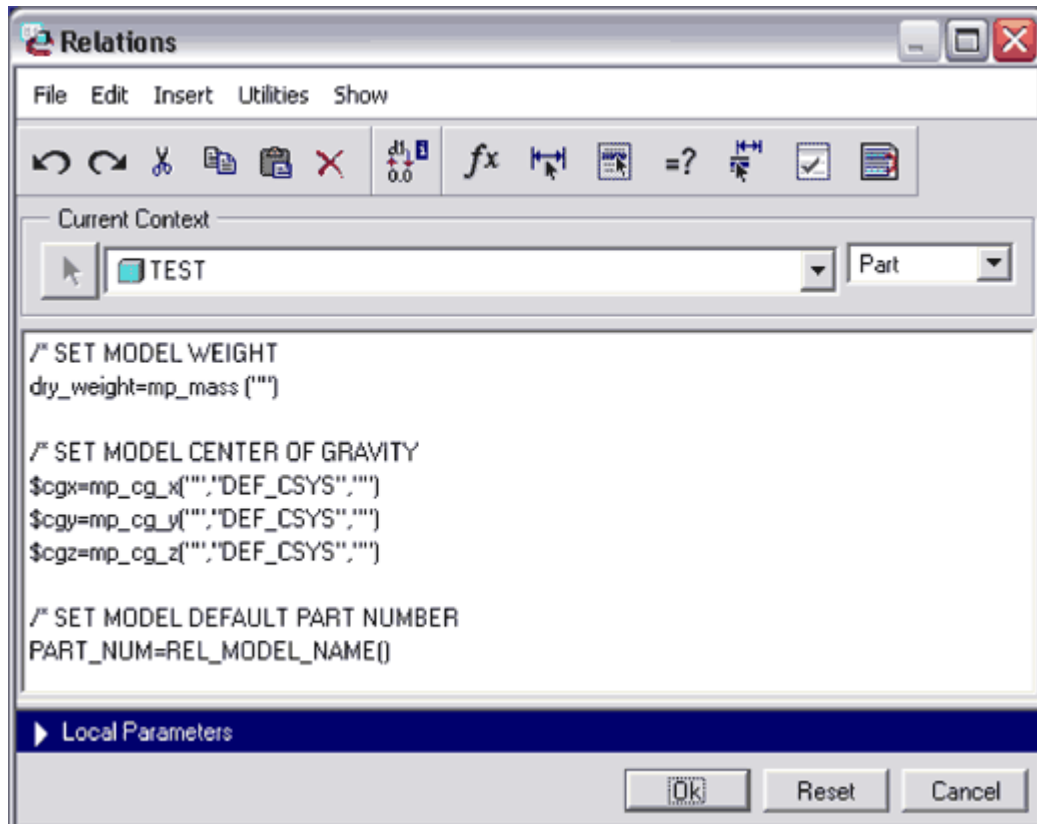
<b>RECOMMENDED CONFIGURATION OPTIONS</b>	
<b>OPTION</b>	<b>SETTING</b>
allow_ref_scope_change	yes
default_ext_ref_scope	skeleton_model
default_object_scope_setting	skeleton_model
enable_component_interfaces	yes
force_new_file_options_dialog	yes
maintain_limit_tol_nominal	yes
modelcheck_enabled	yes
multiple_skeletons_allowed	yes
new_parameter_ui	yes
new_relation_ui	yes
open_simplified_rep_by_default	yes
sketcher_intent_manager	yes
scope_invalid_refs	prohibit

**Default Datums.** The default 3 planes and coordinate system usually requires an axis. Create any default datums you feel are necessary for most parts.

**Parameters and Relations.** Default parameters are used for consistency and compatibility with any formats you've defined. You can add parameters used for describing the part, bill of material information, and weight calculations. Pro/ENGINEER 2001 introduced a new parameter user interface as a hidden option, which you can use by setting the configuration option *new\_parameter\_ui* to *yes*.



Adding relations to your start parts is also quite handy. For example, you can create relations to store the mass of a part as a value or automatically add a prefix to your part number. In Pro/ENGINEER 2001, you can use the new relation user interface by setting the configuration option *new\_relation\_ui* to *yes*.



*Settings to store last known value for weight and center of gravity.*

**Skeletons.** Skeletons form the geometry prescribed by the layout. Since they are very lightweight and very fast to regenerate, I recommend having a skeleton for each assembly. If I'm working on an assembly that only has a few components, I'll sometimes use assembly datums rather than a skeleton. But if your assembly has three or more components, skeletons are a better choice.

Multiple skeletons in an assembly are highly useful. When dealing with subassemblies that affect each other, I add the skeleton from one assembly to the other as a *secondary* skeleton. Remember that the first skeleton is placed automatically, so you want that skeleton to be native to your subassembly. When you add other skeletons, you can then place them.

#### *Tips for Using Skeletons*

- *Name your interface datums.* I find that using a prefix like "SK\_" followed by the datum name used in the layout allows quick identification during query select.
- *Use a layer scheme, typically defined by your start parts.* Don't be afraid to add more layers to group key datums. (By using the *isolate* layer option, you will see only the datums required for assembly.)

**Component Interface.** Component interfaces allow you to predefine your assembly datums in each part so that, at the time of assembly, there is no question about which datums to use. This option is particularly useful when it's time to pass designs off to others. Component interfaces are a hidden option in Pro/ENGINEER 2001, which you must turn on using the configuration option *enable\_component\_interfaces* and selecting *yes*.

**Copy Geometry.** Ideally, you should limit the use of copy geometry as much as possible. The nasty thing about copy geometry is the parent/child relationships that it can create. Imagine working in a large assembly consisting of a few hundred parts. If you make a copy geometry to a part in a subassembly, Pro/ENGINEER stores the reference required to regenerate the copy geometry in the part. To redefine the copy geometry, Pro/ENGINEER will now have to retrieve that assembly with those few hundred parts into session. The retrieval scenario applies in the Pro/INTRALINK environment as well, compounding the problem because Pro/INTRALINK will manage any changes to the large assembly across any versions saved.

I suggest using copy geometry for visual references only. Since we drive our key dimensional attributes from a layout, there is minimal need for copy geometry. In fact, minimizing copy geometry helps to fulfill parametric autonomy.

**Feature/Datum Naming Conventions.** As mentioned in the discussion of layouts, datum naming is key to documenting your design intent. Naming features can be just as important as naming datums. For example, rather than having "hole 123" show up in the model tree, naming it to "piston\_bore" is much more helpful. I'm not suggesting you name everything, but simply that you name and document your design intent.

## Implementing the Plan

Another place that users get overwhelmed in top-down design is at the "where-to-begin" dilemma. I find it best to start with the upper-level assembly and create its skeleton model and develop some geometry using datums. At the same time, I also create a temporary drawing of the skeleton to export as a .dxf. I then import the .dxf as groundwork to insert dimensions (parameters) in layout once the skeleton is well roughed in, or to sketch a 2D geometry reference in another CAD program. Note: Wildfire 2.0 has great AutoCAD support (from release 12 to 2000), offering a host of options for importing your AutoCAD drawings cleanly.

### 1. Create the Skeleton Model

Sketching with curves is excellent because it provides

- easy references to use to define datum planes and axes
- robust geometric references for copy geometry, which you can then use to create part features (although I *strongly* caution against using copy

geometry for part features), and

- a great way to control your datum and surface features in your skeletons.

When using a skeleton to “sketch” your idea, you will also be trying to get a reasonable start on usable dimension values. To define your design, you’ll find that you do a little bit of skeleton work and then layout work. You don’t necessarily do one before the other, and quite often the process is a parallel effort.

## **2. Associate Values from the Skeleton to the Layout**

Once you have a working skeleton, it’s a good time to populate a layout that contains parameters required to drive the key areas of the skeleton. Declare the layout to your skeleton and start associating your values to your skeleton and parts.

Another complaint I hear about TDD is that it’s “too restrictive” and “I can’t edit dimensions on the fly, since I have to set values in my layout.” To disprove these claims, let’s assume you’ve created datums and curves, and perhaps even some surfaces in your skeleton. You’ve also made key parameters in your layout with no particular value since you don’t really know what’s required.

At this time, declare your layout (Declare option in the part menu). To link the values in your layout to your skeleton, you have to create relations. Don’t do this quite yet since you have different values in two places. You should therefore set the parameter values from your skeleton into your layout parameters. Now you may create your relations and regenerate your part. Of course, nothing will happen since the values are identical.

Making edits to many values using the layout can be tedious. At the same time, layout does not allow you to use the dynamic drag functionality to “eyeball” geometry changes. So what you do is simply “comment out” the relations in the relation editor to let Pro/ENGINEER temporarily assume these relations are not required. To comment out a relation, put a “/\*” (slash-asterisk combination without quotes) on the leftmost side of the relation. Now start modifying your feature. When you find the dimensions you like, simply edit your layout with the values, uncomment the relations, and regenerate.

Creating copy geometry is a step in the process whether you use it to drive geometry (not preferred) or as general visual feedback geometry (preferred). At this point, we have a working skeleton and layout. You can now start creating some published geometry to contain the working references other designers (or yourself) are allowed to use for referencing your skeleton datums.

## **3. Populate Your Assembly**

We’re in the homestretch now. By this time, you’ve put a lot of effort into creating skeletons and assembly structure by way of subassemblies and

subassembly skeletons. You can now integrate parts you're designing—whether they already exist or are being modeled for the first time—into your TDD assembly structure.

For existing parts, take the opportunity to update your models to utilize the layouts you've created. Since TDD can be implemented into existing designs (see my previous article, "Reverse Engineering"), you can slowly update your legacy models to take advantage of a TDD practice.

For parts you are now creating, you should assemble the part with its default datums so that they are the first features of the model and less likely to change. If you cannot use the default datums, be sure to make your assembly datums in the part as early as possible. As you design your part, keep in mind which features are going to be driven by the layout.

This is when you get the payoff from parametric autonomy. While copy geometry may be faster, you pay in the end for managing the references and for risking *bad* references. You will find you model the same "feature" more than once, but once you link those features with the layout, any updates require only that the layout be in session.

#### **4. Flex the Power**

By this time, you've created an assembly, skeletons for the assemblies, and layouts to drive the skeletons and assembly components. Now you are most likely ready to start tweaking for a final design. You should start to experiment by modifying values from your skeleton models and layouts to test your model's robustness.

#### Summary of Benefits

Since advocating against copy geometry for production design is likely to evoke a lot of complaints, I want to explain my reasoning and the benefits of my parametric autonomy approach.

*1. Minimal parent/child relationships.* The reason Pro/ENGINEER users (including experts) get into a lot of trouble is through the parent/child mess we can create. Whether designers are rushing or simply don't know better, users often have to deal with something not regenerating because it cannot find its reference.

The solution? Minimize the dependencies. Rather than create a copy geometry in an assembly, create a layout to drive the geometry. You only have to deal with two dependencies—the layout and the part it is declared to. If you declare the layout to more parts, those declarations still have no effect on the first part unless you change the layout. In contrast, if you mistakenly create a copy geometry while in an assembly with 1,000 parts, you need that 1,000 part assembly in session to update any copy geometry.

2. *Rapid changes with skeletons.* If done well, you can create a simplified representation rule at the highest level of your assembly and include only skeletons of everything below it. With layouts and straight geometry modification, you can make large shifts in your model with minimal regeneration time. After saving your assembly and any modified skeleton models, you can then start regenerating the subassemblies to implement the change. In this way, you reduce the amount of hardware resources at any given time to ensure safe regeneration.

3. *The “shown” dimension debate.* When it comes to drawing documentation, the shown versus created dimension debate is a hot topic. Because you’re using dimensional values from a layout, you can in fact still show your dimensions. If you had used copy geometry features, you would have to create your dimensions from those features.

4. *Robust feature creation.* If you are just conceptualizing or in a proof-of-concept panic, copy geometry features can be very useful for a one-off design. If your design is for a “produced” item that others will work on or that will exist for a long period of time, however, parametric autonomy is a better investment. Your model will be stronger and adapt more readily to change. ♦

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