CHAPTER 1

Understanding Assemblies

This chapter serves as an overview of some of the different tools and techniques that are available to SolidWorks users in assemblies. Most importantly, this chapter discusses the various purposes that you might have for creating assemblies. The second emphasis of this chapter is to help you understand various methods for using external references. More than anything, this chapter prepares you for important decisions that you will need to make relating to your modeling methods in SolidWorks found throughout this book.

If you take the SolidWorks training class from a SolidWorks reseller, all assemblies seem to take the same form and have the same function. You may then take this way of working back to your office and start applying it there. However, if you do this, you could be missing out on many other ways of using assemblies. While SolidWorks definitely seems to have a certain orthodoxy in mind for the assemblies functionality, there are actually an array of techniques that you can use to achieve a wide range of goals.

This chapter helps you identify some of the ways in which you can apply SolidWorks assemblies functionality to accommodate your goals and your workflow style. You are encouraged to experiment and evaluate some of these methods to find out what suits your needs best. You don't have to accept the established techniques. In fact, as you will see in this chapter, the established techniques tend to be less efficient, and especially less robust, than some alternative methods when you start making changes to your assemblies.

IN THIS CHAPTER

- **Why use assemblies**
- **Creating assemblies**
- **Positioning parts into assemblies**
- **Working with external references**

A lot of these alternative methods have been developed by many different users of other CAD packages over the years and have become universal to some extent. They have been adapted to SolidWorks use in different forms.

Understanding the Purpose of Assemblies

In the physical world, assemblies exist for several reasons:

- Separating materials
- Allowing relative motion
- Reducing material
- Allowing for different manufacturing techniques
- Allowing for disassembly or repair

In a CAD model, you need to follow these physical-world reasons for making individual parts and putting them together in assemblies, but CAD models can also have additional requirements. Independent of the reasons stemming from physical-world requirements, CAD assemblies might have some unique reasons for existing:

- Depicting an assembly process such as order of operations
- Specifying dimensional assembly relationships and tolerances
- Establishing clearances and limits of motion
- Visualizing motion and spatial relationships between parts
- Designing parts in-context
- Creating a parts list for assembly (Bill of Materials, or BOM)
- Creating a parts list for purchasing
- Automating data entry through PDM (product data management)
- Staging renderings
- Creating data for downstream applications such as animation or motion analysis

You can probably come up with a number of additional reasons for making CAD assemblies. In fact, almost as many reasons exist for making assemblies as there are people making those assemblies.

If you are trying to drive product development with a single top-level assembly, you might run into situations where the various functions of the assembly start conflicting with one another. For example, you might have an assembly where a part flexes. It is difficult or impossible to make flexible parts work effectively in SolidWorks with dynamic assembly motion. Another situation might be in-context relations where the parent and child components

move relative to one another. Or maybe you need an assembly for a rendering and the assembly has to have multiple instances of in-context components, which can be tricky to manage. You get the picture. You can't always do everything with a single assembly.

It is certainly possible to have multiple assembly files for a single product. In fact, in some cases, it may be necessary. Rendering is probably one of the most common reasons for you to create a new assembly. Conflicts between external references and motion are another common reason to create a new assembly document.

Identifying types of assemblies

The average SolidWorks user thinks an assembly is a collection of parts put together with mates that position parts and may also allow motion. In this kind of assembly, you might use patterns, configurations, in-context techniques, and so on. The goal of the assembly is probably to simulate reality in the way it looks and moves.

Driving an assembly with base part and mates

This is considered "orthodox" SolidWorks assembly usage, and is the way the SolidWorks training materials describe creating assemblies. Insert a part or subassembly at the origin, which becomes fixed in place automatically, then start mating parts and subassemblies to the base component and add on from there.

This is the most frequently used assembly type in SolidWorks, but this type is also the most prone to failure, and the least likely to allow for relationships to exist between the parts. If you have been doing this kind of work already, and have made changes to the parts and watched the assembly update, you know that there are all sorts of things that can go wrong with this arrangement. Mates coming into conflict, flipping direction, or losing references are the most common errors that you might see when mating parts directly to one another in an assembly.

It is hard to imagine an assembly in real life where the parts don't depend to some extent on one another for their size or shape. So having a SolidWorks assembly where each part is modeled independently doesn't reflect real-world design intent very well, nor does it really use the advantages of parametrics and associativity that are touted so much in SolidWorks.

It is difficult to criticize an assembly modeling method that has been used for so long by so many people, but the high failure rate of mates attached to edges and faces speaks for itself. This is a problem that SolidWorks has tried to solve for many releases of the software, but the failure rate hasn't changed much, if at all. You may find that the software even tries to hide certain types of mate errors to make them easier to ignore.

The difficulty arises from the underlying methods built into the software. All of the faces and edges of parts are created using a history-based modeling system. Each feature is created in order, in a recipe for creating the final piece. The assembly, however, is generally not history-based. So mates attach to faces. If the faces change in such a way that the mate can't attach

anymore, the mate fails. If the faces change in such a way that the way the mate works changes, parts can move in a way you don't expect. If you add fillets, or draft, or split a face to add color, or do any of a number of things that change the way the software internally identifies the faces or edges, then the mates you have applied will either fail or change the way it locates or allows parts to move.

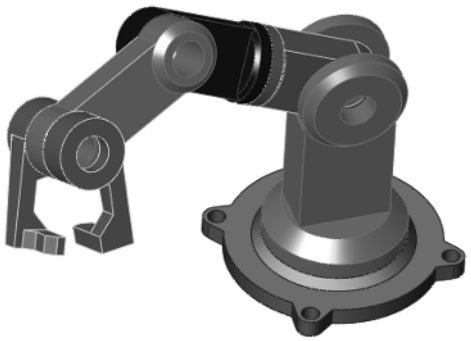
The bottom line for the method of mating to base parts is that it is unreliable through changes. Of the methods that are presented in this book, this is the most common yet least reliable method. This is not the fault of the software, but of the method. The reason this faulty method is the most popular is because it is the easiest, and requires the least planning.

When SolidWorks first appeared on the market in the mid-1990s, it became very popular not only because it was inexpensive but also because it was much easier to use compared to products such as Pro/ENGINEER. Pro/ENGINEER taught methods for putting parts and assemblies together that were tedious but worked better through changes.

As an example of where you might use this kind of assembly, think of a robotic arm. Figure 1.1 shows an assembly that was created with bottom-up techniques and was assembled with face-to-face mates.

FIGURE 1.1

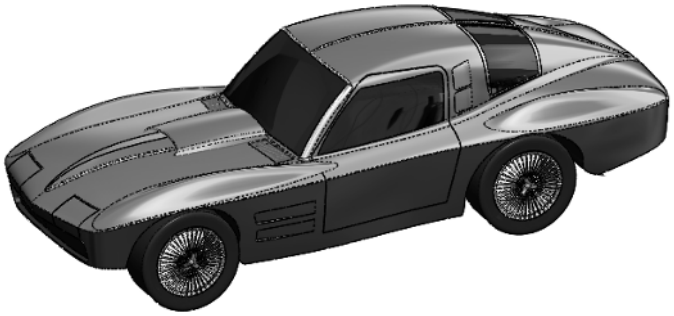
Mechanical parts using mates to locate and enable motion



But if you were to create, say, a scale model of a car, as shown in Figure 1.2, the method of independently designing each component, especially something like body panels, and then mating them together wouldn't make much sense. Other methods in the other types of assemblies shown later in this chapter will help with this type of design.

FIGURE 1.2

Considering how you would design the parts of a scale model car



It would be difficult or impossible to design the body panels of the car such that they fit together well and looked smooth next to one another using the bottom-up with mates method.

Driving an assembly with sketches and planes

One way to avoid the potential pitfalls of mating to a base part is to replace the changeable faces and edges with items that are more stable. The stability hierarchy listing items from the most to the least stable looks like this:

- Assembly or part origin
- Assembly or part standard planes
- Reference geometry (plane, axis, point)
- Reference geometry from inserted parts (from using the Insert➪Part command)
- Sketch lines and midpoints
- Sketch endpoints
- Surface model faces
- Solid model faces
- Edges and vertex points
- In-context items
  - Reference geometry
  - Faces
  - Edges

An easier way to remember this without memorizing the list is that the more parents something has, the less reliable it is as a reference. This becomes more applicable if external references are involved, such as inserted parts or an in-context situation. Edges created by fillets or chamfers are lower on the list of stable references than other edges.

There is no clear answer to the question, "Is this reference stable enough?" It is entirely possible for you to be completely successful using in-context edges for all your model references. In order for that to happen, you would have to plan your model very well and avoid any big topological changes (changes to the number or function of faces) to the model.

Cross-Reference

Chapter 10 covers in-context modeling in more depth, and Chapter 19 covers inserted parts. ■

When you are building a part, selecting references from near the top of the previous list can be challenging, especially when faces and edges are so easy to use. You need to evaluate how much editing and rework you think you will generate when changes that you haven't necessarily planned on have to be made. Much of this ties into the design intent discussion from the *SolidWorks 2011 Parts Bible*.

Now consider the two examples mentioned in the last assembly modeling method — the robot arm and the model car. You could design the robot easily with the sketch layout, but simulating the motion in 3D would be difficult if you did it in conjunction with the sketch. The model car would still be difficult to assemble, and if you were just using sketches as the references between parts, it would be difficult to design the body panels such that they fit together smoothly.

Modeling parts in place

In-context design is discussed in more detail in Chapter 10, but here you will get some idea of what to look forward to, and why this book isn't just about in-context design by itself. Modeling parts in the context of an assembly that contains other parts enables you to make relationships between the parts. Those relationships are managed by the assembly. The parts have to be arranged spatially with respect to one another, and the references to the files must also be managed.

When you see a sales demonstration, the technique of using edges of other parts from the assembly to make a new part looks very compelling, especially when you make a change to the other part and the new part updates as well. It's hard to argue with that kind of functionality. But the price you pay for that sort of associativity is that you have to manage the relationship between three files: the parent part, the child part, and the assembly. And further, within the assembly, the relationships are made between specific instances of the parts, so if you have multiple instances of each, you have to do something to remember which pair of parts is the driving pair.

Also, model history with parts does not work the same with assemblies. The relationship between the parts does not have any memory, so if you started the in-context relationships

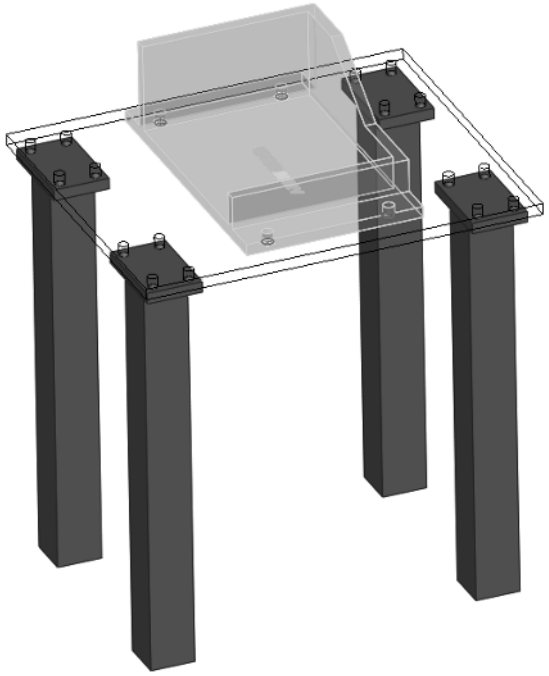
before the parent part was complete, and then put fillets over the edges that you had referenced, your in-context references would fail.

Take another look at the robot arm and the model car examples. Using in-context methods, you could certainly design the robot arm, but again you might run into some problems with getting it to move correctly while maintaining the references. However, with the model car, getting the parts in the right place wouldn't be any problem because you would be modeling the parts in-place. On the other hand, you might be able to get the shape to flow smoothly, but it is still doubtful. In-context modeling can copy 3D surfaces between parts, but for an improved workflow for this type of work, you will have to read further into this chapter.

An example of a part where modeling in-context works well is a table with legs, as well as a fixture that sits on the table, as shown in Figure 1.3. The in-context work lines the holes up between the parts. There is no relative movement between the parts, and the individual parts are not likely to be used in other assemblies.

FIGURE 1.3

Using the in-context method to its best advantage



The ideal situation to use in-context techniques is when two parts are assembled face-on-face, the shape of the contact faces are the same or offset, and there is a set of holes used for fasteners. Complex shapes are generally not a good candidate for in-context methods. The main point is that there is no relative motion between the two parts.

Modeling parts as multi-bodies

Another method you can use to model parts is to start the models in a multi-body part. It is not recommended to use this method for creating finished parts as multi-bodies, but getting some of the major parts on an assembly started as a single part and then breaking them out into individual parts for details can be a very effective method.

Say you are modeling a riding lawn mower, and you need to create the plastic cowling on the front of the mower. The cowling is made up of multiple pieces because some of them are different colors, and some are transparent. The complex shapes of the cowling encompass multiple parts. If you were to model one part, and then try to model another part independently that shared some of the same shape, it would be very difficult or impossible to get the shapes to match acceptably.

One answer to this problem is to create the shape in a single part, then break the single part into individual bodies, and then save the bodies external to the original part. When you put these parts back together into the assembly, each part can be placed so that its origin matches with the assembly origin. Because the parts all started from the same part, they will all share the same origin. This makes putting the parts back together much simpler. It makes assembly for motion more difficult, but parts that have a shape in common are more likely to be fixed with respect to one another.

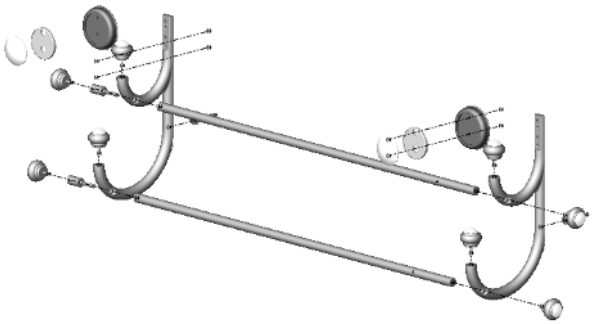
Multi-body modeling has advantages over in-context modeling in that it reduces external references (although saving bodies out as parts creates an external reference), but it also has some drawbacks. If you were to take all the features of individual parts and stack them into a single feature tree in a single part, you would probably be unhappy with the result. By making all of the features for all parts within a single part file, you make troubleshooting much more difficult, and rebuild times are dramatically increased. Add to this the inability to reuse parts, do individual revision management, or perform simple assembly operations such as dynamic motion, exploded views, or BOMs, and following the multi-body method through to finished parts becomes very unattractive.

The best option for using multi-bodies to create parts for an assembly is to start the parts in multi-body mode, and then as soon as the inter-body references are no longer needed, transition the bodies to separate parts.

Multi-body modeling may not do so well when parts are repeated, or where purchased components represent a large percentage of the total parts. While you do have mate-like functionality for placing bodies within a multi-body part, it is probably not the best use of this method. Figure 1.4 shows a product that is designed as a multi-body part but involves many difficulties because of reused parts and hardware.

FIGURE 1.4

Reusing parts is not a strength of multi-body methods.



Revisiting the test for each method, you would find that the robot arm is well suited to being designed as a multi-body part and then reassembled with mates in an assembly. In fact, the multi-body method is probably the best method for this type of work, maintaining references between parts, and then assembling the parts into an assembly mechanism with motion.

The model car, with its shape that flows between parts, would still be awkward, although it could be done as individual parts. You will now look at the last method to see if this helps with the car model.

Inserting a master model

You will learn about the master model technique in Chapter 19. In a nutshell, a master model is a single part where you place sketches, reference geometry, surfaces, and maybe some solids, and then insert that part into other parts to use a reference to build each individual part. Using this technique makes in-context work unnecessary, and eliminates some of the dangers of creating too many features in a multi-body part.

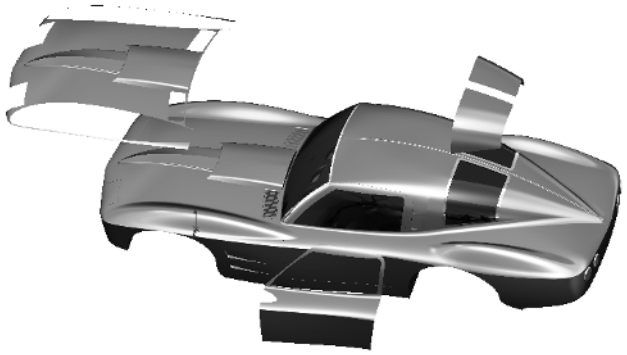
You assemble parts in this manner the same as with the multi-body part method. Take each part, and drop it into the FeatureManager of the assembly. This aligns the part origin with the assembly origin, and because each part was built from the same master model, all parts share the same origin.

Take another look at the robot arm and model car examples. The robot arm may be a little awkward using this method, but it works. The multi-body method is probably best for this type of design.

On the other hand, the master model method brings real power to projects such as the model car. You can design the entire outside of the car as if it were a single part, and then break it up into individual parts. In Figure 1.5, notice how some parts that will be manufactured as a single part can be easily pulled off of the master model. Sketches within the master model can help define breaks between parts and even if there is relative motion — for example, with the doors.

FIGURE 1.5

Pulling parts off of the master model



Excluding some parts

Methods such as those mentioned previously are great for any part that is unique to an assembly, but should not be used for library type parts. If you have a part that will be used in more than one assembly, it should not have any external references. To be clear, all of these methods create external references except bottom-up assembly (where parts are modeled individually, and then assembled to one another or to a skeleton).

Any library parts that you have, or standard hardware such as nuts, bolts, washers, and so on, should not be modeled with references, nor should you use them in multi-body parts.

Without sounding against multi-body modeling, several SolidWorks users think eliminating the distinction between assemblies and parts would be a good thing. That point of view works for only the simplest small assemblies with simple parts. It's easy to come up with situations in which the tree management tools required to maintain models built using that philosophy through changes do not even exist in the software.

Creating an alternative to multiple assemblies

It may seem very inefficient to re-create or copy assemblies for different uses. In Chapter 8, you will learn about another way: using assembly configurations. If you are already familiar with part configurations, assembly configurations work on a similar principle. Assembly configurations are a great tool with a lot of theoretical benefits and practical limitations. As with most other functions in SolidWorks, when you need to create assemblies for multiple purposes, you may find that assembly configurations meet your needs. Or you may find it easier

to just save out a copy of the assembly, knowing that changes for the sake of rendering do not diminish the usefulness of the data for something like exploded views, or managing external in-context references.

Creating Assembly Templates

When you have assemblies with different purposes, you may also need multiple assembly templates. One example of using assembly templates to help reuse work is saving an animation in a template. If you save an assembly template that contains a default turntable animation, then you can put other assemblies right into the turntable template and you've got an instant display animation. (Animation is covered in detail in Chapter 23.)

Another example of a useful assembly template would be a rendering setup with environment, background, and lighting. Putting a part or assembly into the template before doing a stock rendering can help you standardize renderings and work through them more quickly.

The steps to create an assembly template are similar to those for setting up a part template:

1. Specify a custom location for all your templates. You can do this at Tools➪ Options➪ File Locations➪ Document Templates. You should save it in a location such as `D:/Library/Templates/`. For a group of users, you may want to use a network location so you can share the templates.
2. Start with an assembly that has most of the settings you intend to use.
3. Add whatever animation, scene, or light settings you want. You may have to add a dummy assembly to get the settings right, and then delete it before saving it as a template.
4. Set the options in Tools➪ Options➪ Document Properties the way you want them. Pay special attention to the Drafting Standard, Units, Model Display, and Image Quality settings.
5. Rename the standard planes as appropriate. If your assembly template is bound to be used for injection mold tooling, you may name planes to establish the Parting Plane. If you are creating architectural assemblies, you may have planes called Plan View, Side Elevation, and North Elevation.
6. Make sure the custom properties (found at File➪ Properties➪ Custom) for the assembly template are set the way you want them. Remember that you can use Tags in some ways like custom properties. In fact, custom property settings might be a reason for making separate assembly templates. You might have a different set of custom properties for a tooling assembly used in manufacturing as opposed to a

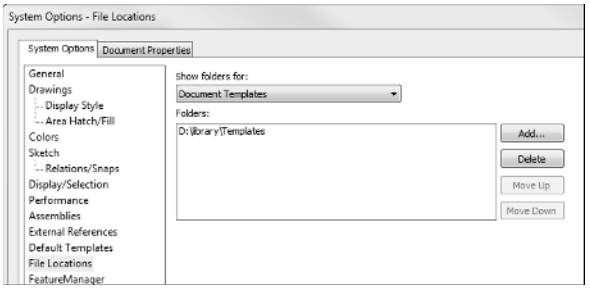
product assembly that is shipped to customers. You may also choose to change the default settings for showing annotations such as cosmetic threads and shaded cosmetic threads (click the right mouse button (RMB) on the Annotations folder in the Feature Manager and select Details for these options)

7. When you have all of the document-specific settings the way you want them, save the assembly file as a template. Choose File➪ Save As, and SolidWorks directs you to the folder where you have specified that your templates will go.

Figure 1.6 shows the Tools➪ Options➪ File Locations interface where you should set your templates' locations.

FIGURE 1.6

Establishing the location of your templates



Using assembly templates is easier than creating them. To choose from a number of assembly templates, you have to use the Advanced interface on the New Document dialog box. If you use the Novice interface, you can only choose the default templates.

If you want to use only a single assembly template, and set that one as the default, first save the template to a location as described previously, and then set the default assembly template at Tools➪ Options➪ Default Templates.

If you would like to devote an entire tab of the New Document dialog box to just assemblies, then in Windows Explorer, in the folder specified in your template locations, create a new folder named Assemblies or something relevant. This folder name will show up as a tab in the Advanced interface for the New Document dialog box.

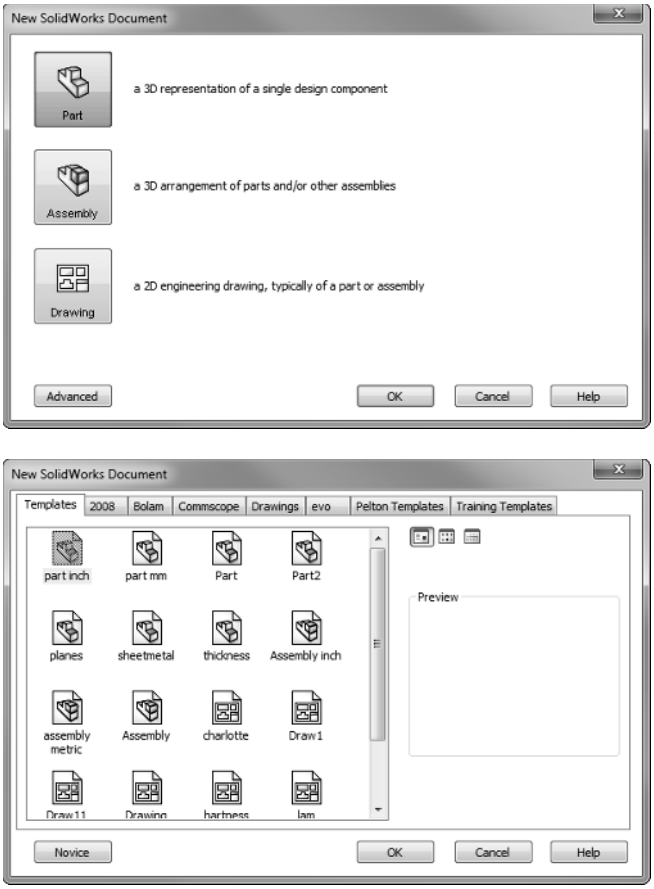
Note

The Novice interface displays a button that says Advanced, and the Advanced interface displays a button that says Novice. ■

Figure 1.7 shows the Novice and Advanced interfaces.

FIGURE 1.7

Starting a new assembly from the Novice and Advanced interfaces of the New Document dialog box



Putting Parts into Assemblies

When you are building an assembly, several ways exist to put parts into assemblies:

- Choose Insert➪Component
- Drag and drop from Windows Explorer
- Drag and drop from other SolidWorks windows
- Drag and drop from the Design Library window
- Ctrl+drag to add a second instance of a part that is already in an assembly
- Create an assembly from a part
- Create a part in-context

The first part that you put into an assembly is always fixed (meaning locked into position). When parts are fixed automatically, the origin of the part is always located at the origin of the assembly, but parts may be manually fixed at any location in the assembly.

After the first part, any additional parts you put into the assembly will either fall where you drop them if you drop them in the graphics window or be positioned at the assembly origin if you drop them into the FeatureManager.

One of the most valuable methods is to use SmartMates and Mate References. SmartMates enable you to Alt+drag a part by specific geometry on the part and drop it onto specific geometry on another part; a mate is then automatically created between the parts. SmartMates do take some practice, but they help you save a lot of time and frustration when putting an assembly together from parts. Mate References are similar, except that you have to set them up beforehand (and so they work best on library type parts), and they enable a part to snap into place when you drag it into an assembly.

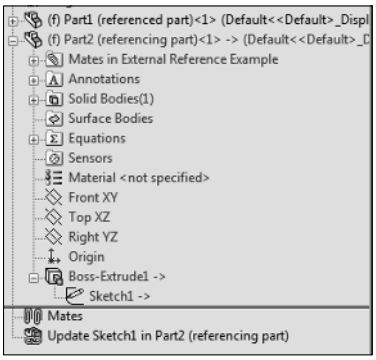
Understanding External References

External references are one of the most time-consuming aspects of assemblies, and all assemblies and assembly replacement techniques have them. An external reference is any reference to a file outside of the current file. So in its simplest form, part files in an assembly create external references, because the assembly references the parts.

You can easily recognize external references of all kinds by the -> symbol following a feature in the FeatureManager. Figure 1.8 shows a portion of an assembly FeatureManager that contains the symbol. You can also find this symbol on externally referenced features within a part.

FIGURE 1.8

The external reference symbol on a sketch and a feature

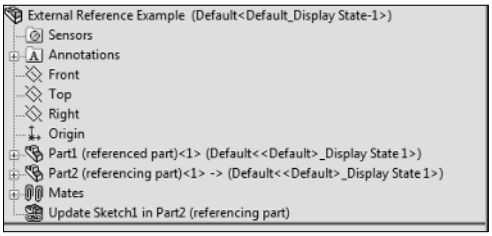


Referencing external files in-context

In-context techniques are all about external references. External in-context references in an assembly occur when one part in the assembly has some sort of geometrical reference to another, such as offsetting an edge of a part, using a vertex of a part with a coincident sketch relation, or using a face of another part as a sketch plane. All of these references are saved in Update Holders that reside in the assembly FeatureManager, but SolidWorks hides them by default. Figure 1.9 shows a simple assembly FeatureManager with a single Update Holder. You can show the Update Holders in an assembly by right-clicking the top level name of the assembly in the FeatureManager and selecting Show Update Holders.

FIGURE 1.9

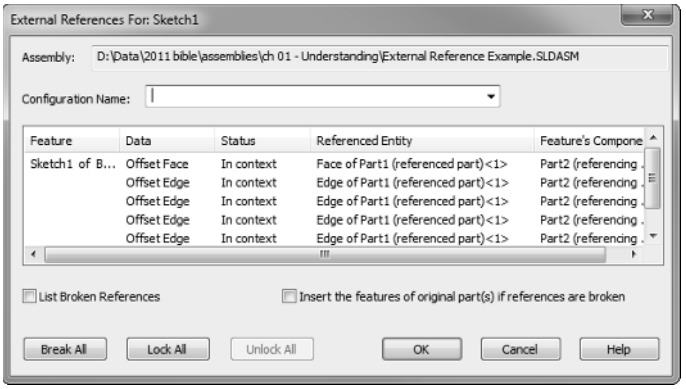
Displaying an Update Holder that keeps external reference information



To view the contents of the Update Holder, right-click the Holder or any part or feature that contains external references, and select List External References from the menu. The information stored in the Update Holder shown in Figure 1.9 is shown in Figure 1.10.

FIGURE 1.10

Showing the contents of the Update Holder



In the example shown in Figure 1.9, four edges of Part1 are offset into Part2. Also, the Part2 sketch into which the edges are offset uses a face of Part1 as the sketch plane.

Note

The Insert the features of the original part(s) if references are broken option in the External References For dialog box does not apply to assembly in-context situations. It only applies to inserted parts (what the Help system calls "derived parts"). Inserted parts are also external references where references can be broken. You will find more information on this topic in the next section. ■

Each Update Holder holds the external reference information for a single feature. If a sketch and a feature have different external references (for example, the sketch might reference face edges, while the feature might reference a vertex in the other model for an Up To Vertex end condition), there will be a different Update Holder for each. The external reference information has several components:

- The name of the assembly where references are made
- The configuration of the assembly where references are made
- The feature where the reference was created
- The type of entity that was referenced, and which part it is in
- The part from which the reference was created

Referencing external files from a part

Chapter 19 is all about referencing external files from a part. As a preview of what you will see in that chapter, you can insert one part into another part to use in a number of ways, such as the following:

- A starting point, for example, adding secondary operations to a cast part
- A tool, for example, using the Indent feature to create a clearance for an interfering part
- A Boolean operator, for example, to subtract one part from another
- A set of reference geometry, for example, inserting the surface of a car door to create the outer skin of the door

To create these external references, SolidWorks provides four different features or functions:

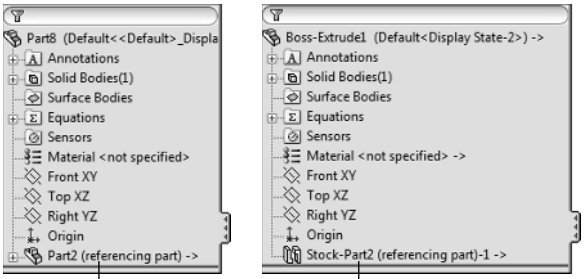
- Insert Part
- Insert Into New Part
- Split
- Save Bodies

External references within parts can be broken in the same way that in-context relations can be broken. External references in parts have just one fewer component compared to assemblies — parts obviously do not list an assembly where the reference was created.

For parts, two different kinds of external references exist: inserted parts (on the left in Figure 1.11) and stock parts (on the right in Figure 1.11). Figure 1.11 shows these two types of references inserted at the top of different part FeatureManagers. Notice that they both have the in-context external reference symbol (->) after the first feature.

FIGURE 1.11

Two kinds of external references in SolidWorks parts



The inserted part allows you to bring across planes, sketches, features, and even the entire part if you want, but on the down side, it forces you to bring all bodies to the new part (you can select solid or surface, but you can't select which bodies). The stock feature enables you to select which bodies you bring to the new part, but it only allows you to bring solid bodies (not surfaces) and does not allow you to bring reference geometry, sketches, or features.

Summary

If you have taken the SolidWorks training classes, even the advanced training classes, you may not have seen the entire range of what you can do with SolidWorks assemblies. Keep an open mind about being able to accomplish more by using assemblies that are built for different purposes.

External references in assemblies tend to intimidate a lot of users, but if you know where to find the necessary information, and how to handle the data, you can have complete control over your assembly models, and you don't have to be afraid of techniques such as in-context or master model.