Design in Industry

Solid Modeling and CAD Fuels Race Team's Success

This case study describes the design of NASCAR race cars for Joe Gibbs Racing using 3-D modeling and CAD. In this case study, you will see examples of how 3-D modeling is used and the importance of 3-D modeling in the design process to improve on the performance of the race car.

Joe Gibbs Racing has won the NASCAR championship two of the last four seasons.

The Race Between Races

With expert drivers like Bobby Labonte and Tony Stewart and fast cars like the #18 Interstate Batteries® Chevrolet Monte Carlo, the #20 Home Depot® Chevrolet Monte Carlo, and, the newest addition, the #11 Federal Express Chevrolet Monte Carlo, Joe Gibbs Racing has the makings for NASCAR success. But winning involves more than sending these men and their cars out onto the track each weekend. So much work must be done on the cars between races that another type of race takes place constantly in the shop as designers and machinists work against the clock.

- **Issues:**
  - Fine-tune cars prior to each race
  - React quickly to rule changes involving equipment
  - Conduct R&D for future vehicle enhancements
  - Stay ahead of the competition
- **Approach:**
  - Model parts and subassemblies in NX
  - Transfer geometry to NX Generative Machining
  - Produce parts on CNC and stereolithography machines
- **Results:**
  - Two NASCAR championships and three 2nd-place finishes
  - 90 percent reduction in cylinder head and manifold grinding time (seven hours versus 70 previously)
  - Suspension adjusted in two hours versus two weeks
  - Scale models quickly available for wind tunnel testing

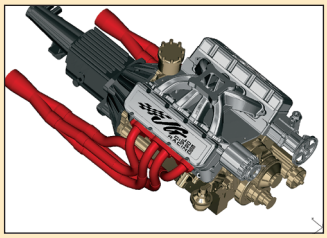
Prior to 1998, the crew had tried using CAD/CAM software to speed the production of custom components. Although this was faster than making drawings and then machining parts by hand, data translation between the two software programs caused errors that slowed the process considerably. That problem was solved with the installation of I-deas® NX series, with its fully integrated design, analysis and manufacturing environments.

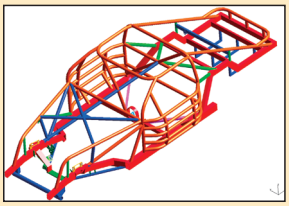
Soon, parts that previously took days or weeks to manufacture were made on CNC machines and available in hours. And as familiarity with the new software grew and more components were modeled within NX, the team started using the software to evaluate new engine and suspension configurations virtually. They also began using the digital environment to optimize weight distribution. Working virtually first and then later in the shop, engineers stripped excess metal from upper components and applied the weight to areas below the centerline of the axles. This way, they were able to improve handling while maintaining NASCAR's required vehicle weight. The software also helped the team find ways to increase engine horsepower. Performance on the track reflected the increasing use of NX, with faster times and better finishes. In 2000, just three years after installing the software, Joe Gibbs Racing won its first NASCAR championship.

Joe Gibbs Racing

A big part of a race team's success depends on the people in the shop and how quickly they can make changes to the car.

Tony Stewart®, #20®, and The Home Depot® licensed under authority of Joe Gibbs Racing, Huntersville, NC.





From Joe Gibbs/UGS.

*Licensed under authority of Joe Gibbs Racing, Huntersville, NC.

CHAPTER 4

3-D Solid Modeling

Objectives and Overview

After completing this chapter, you will be able to:

1. Understand the terminology used in 3-D modeling.
2. Define the most common types of 3-D modeling systems.
3. Apply Boolean operations to 3-D objects.
4. Understand the role that planning plays in building a constraint-based model.
5. Apply generalized sweeps to the creation of model features.
6. Apply construction geometry in the support of feature creation.
7. Apply constraints to a feature profile.
8. Understand how feature order affects feature editing and final model geometry.
9. Apply feature duplication to model construction.
10. Identify the elements used to define a view of a 3-D model.
11. Understand how model data associativity supports engineering design and analysis.
12. Generate 2-D documentation from a 3-D model.
13. Construct assemblies from part and subassembly models.
14. Define the types of analyses that can be used with 3-D models.
15. Understand how CAM information is derived from 3-D models.

Three-dimensional solid modeling is a rapidly emerging area of CAD, revolutionizing the way industry integrates

computers into the design process. Commercial 3-D solid modeling packages, available since the early 1980s, have rapidly made inroads into a wide range of industries over the past few years.

Two-dimensional CAD has matured, in many ways, to the point where simply using a more powerful computer will not have much of an impact on how well a 2-D CAD program functions. Like traditional drafting methods, 2-D CAD programs attempt to represent objects in two dimensions; in fact, the packages were originally developed to be computer drafting tools, with the end product being a drawing on paper. In contrast, a 3-D solid computer model is more like a real object, not just a drawing of the object; 3-D CAD is considered a computer modeling tool.

This chapter introduces the possibilities for, and the limitations of, integrating 3-D CAD operations into the design process. While 3-D solid modeling software has enhanced such integration, it has not yet completely replaced more traditional 2-D documentation or physical prototypes.

The chapter outlines the most common approaches for generating 3-D solid computer models, in addition to how these models are viewed and modified on the computer. A particular focus is put on constraint-based modeling techniques and their relationship to documentation, analysis, and manufacturing technologies.

4.1 Model Definition

A **solid model** consists of volumetric information, that is, what is on the inside of the 3-D model, as well as information about the surface of an object. In the case of a solid model (Figure 4.1), the surface of the model represents the boundary between the inside and outside of the object. The solid model can be thought of as being *watertight*. There must be no gaps between surfaces on the model since it must completely separate the inside of the model from the outside.

Solid modelers typically define only what is termed a **manifold model**. A manifold boundary unambiguously separates a region into the inside and the outside. A lengthy theoretical discussion of manifolds is not appropriate here and also is not necessary because the idea of manifold models is fairly intuitive. It is easy to imagine solid objects as dividing space into what is part of the

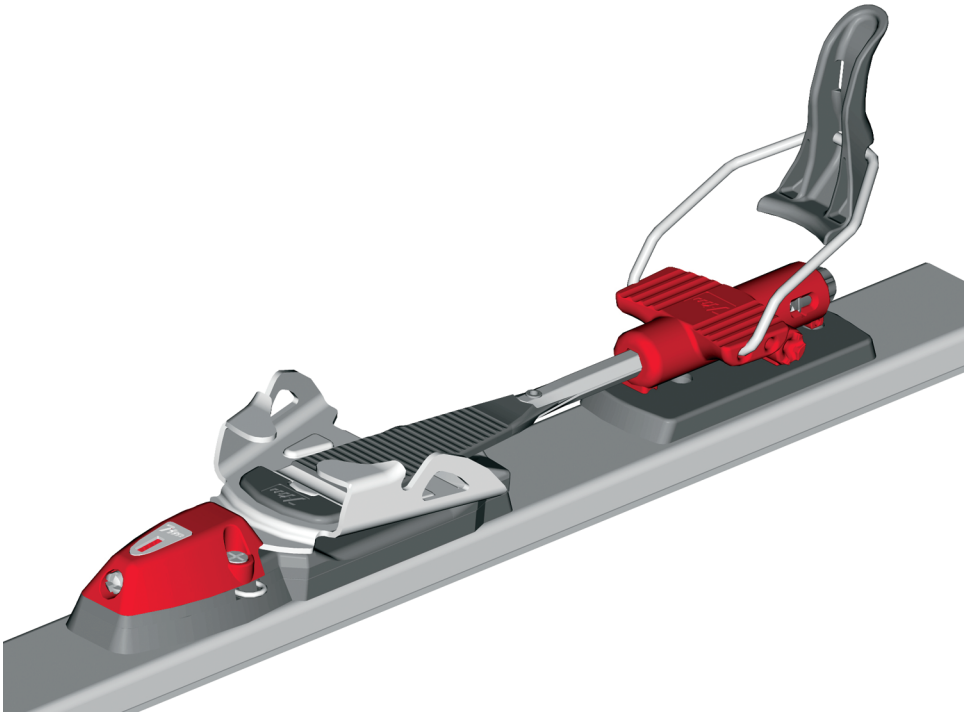


Figure 4.1 **Solid model of a snow ski binding**

(Courtesy of Autodesk.)
Reinhold Zoor for 7tm rezotec GmbH.

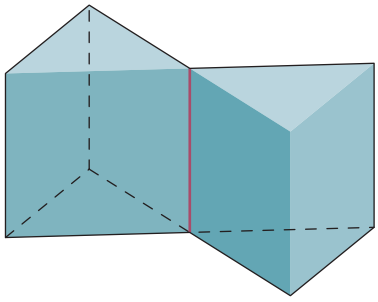


Figure 4.2 **Example of a nonmanifold object**
Most modelers do not support the creation of these types of objects.

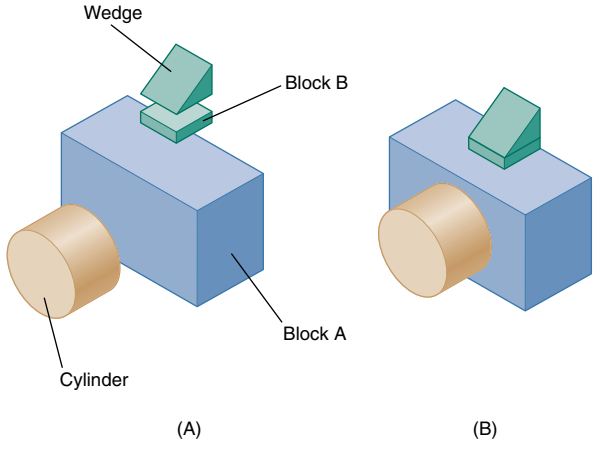


Figure 4.3 **A camera described with geometric primitives**
Additive modeling with geometric primitives allows a variety of objects to be represented.

object and what is not. For example, objects such as the one shown in Figure 4.2 are not allowed. The highlighted edge belongs to four faces of the object, when only two faces are allowed; in real life, you would not have an infinitely thin edge holding two halves of an object together.

4.2 Primitive Modeling

Many objects can be described mathematically using basic geometric forms. Modelers are designed to support a set of geometric primitives, such as cubes, right rectilinear prisms (i.e., blocks), right triangular prisms (i.e., wedges), spheres, cones, tori, and cylinders. Although most geometric primitives have unique topologies, some differ only in their geometry, like the cube and the right rectilinear prism.

A primitive modeler uses only a limited set of geometric primitives; therefore, only certain topologies can be created. This is called **primitive instancing**. However, there is generally no limit to the quantity of instances of an allowed primitive in a single model. Joining primitives together allows the creation of more complex objects. The user mentally decomposes the object into a collection of geometric primitives and then constructs the model from these elements (Figure 4.3).

Once the numerical values for the geometric parameters of an instance are defined, the model must have values for its location and orientation. For example, Figure 4.3A shows each individual primitive that will go into the final object, a camera. Figure 4.3B shows the camera assembled with each primitive at its final location and orientation. To manipulate the camera as a whole, the modeling system must then be able to act on a group of objects all at one time.

4.3 Constructive Solid Geometry (CSG) Modeling

Constructive solid geometry (CSG) modeling is a powerful technique that allows more flexibility than primitive instancing in both the way primitives are defined and the way they are combined. The relationships between the primitives are defined with **Boolean operations**. There are three types of Boolean operations: **union** ($\cup$), **difference** ($-$), and **intersection** ($\cap$). Figure 4.4 (on the next page) shows how each of these operations can be used to create different forms. The critical area is the place where two objects overlap. This is where the differences between the Boolean operations are evident. The union operation is essentially additive, with the two primitives being combined. However, in the final form, the volume where the two primitives overlap is only represented once. Otherwise there would be twice as much material in the area of overlap, which is not possible in a real object. With a difference operation, the area of overlap is not represented at all. The final form resembles one of the original primitives with the area of overlap removed. With the intersection operation, only the area of overlap remains; the rest of the primitive volumes are removed.

In Figure 4.4, Boolean operations also are shown in their mathematical form. The union ($\cup$) operation, like the mathematical operation of addition, is not sensitive to the order of the primitive operands (i.e., $11 + 4$ and $4 + 11$ both equal 15). On the other hand, the difference ($-$)

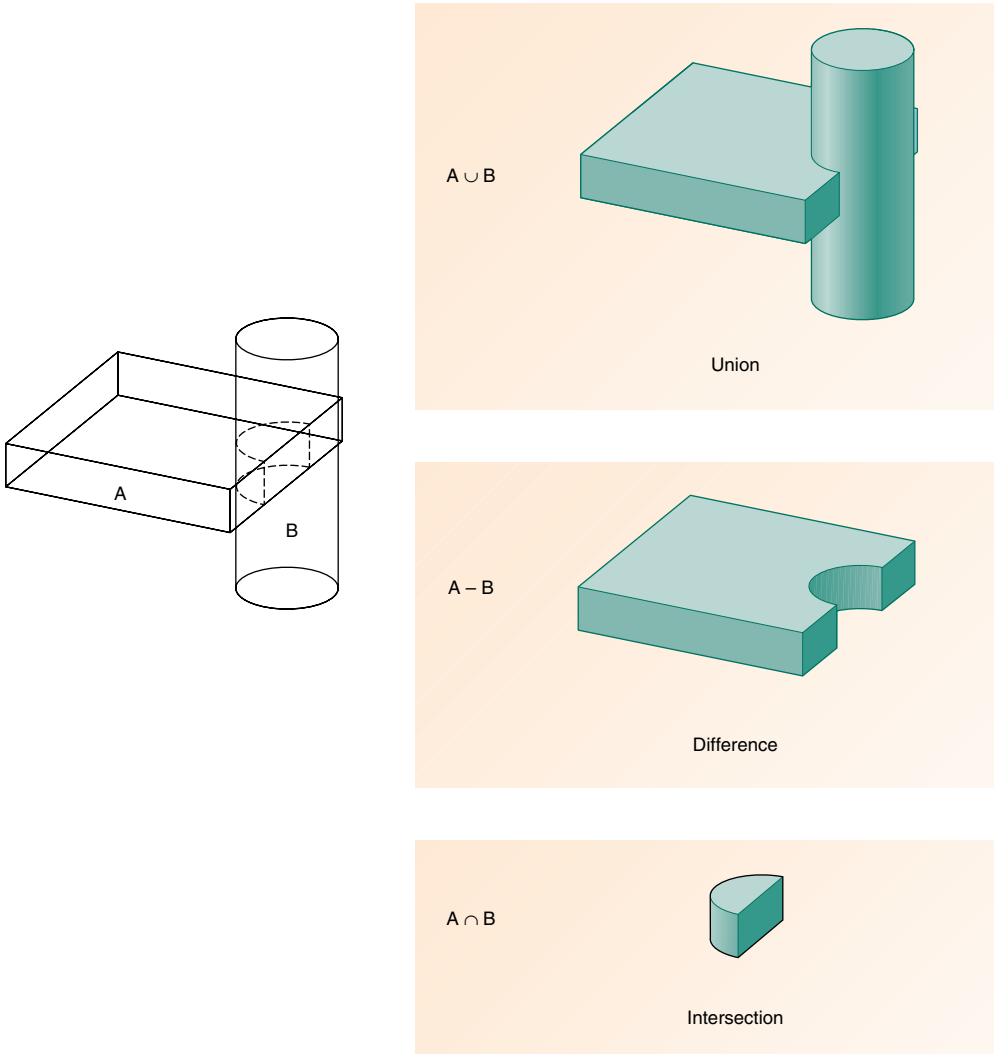


Figure 4.4 **The three Boolean operations: union, difference, and intersection**

The three operations, using the same primitives in the same locations, create very different objects.

operation is sensitive to order (Figure 4.5). For example, 11 − 4 equals 7, but 4 − 11 equals −7. For a Boolean difference operation, the shape of the resulting geometry depends on which primitive (A or B) is first in the equation. The result of the difference operation is that the overlapping volume is removed from the primitive listed first in the operation.

With Boolean operations, it is possible to have a form that has no volume (a null object, ∅). If the second primitive of the difference operation completely encompasses the first primitive, the result will be a null object, since negative geometry cannot be represented in the model.

Primitives that adjoin but do not overlap are also a special case (Figure 4.6). Performing a union operation on such primitives will simply fuse them together. A difference operation will leave the first primitive operand unchanged. An intersection operation will result in a null object since such an operation only shows the volume of overlap, and there is no overlap for the adjoining primitives.

The final form of a model can be developed in several ways. As with pure primitive instancing, you can begin by

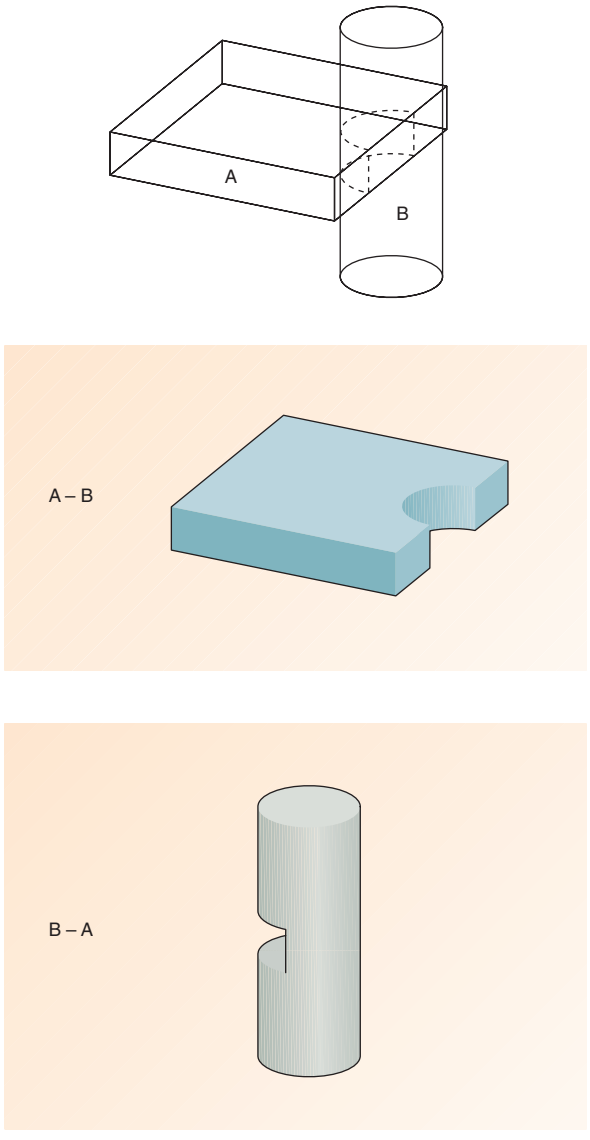


Figure 4.5 **The effects of ordering of operands in a difference operation**

Unlike the union operation, the difference operation is sensitive to the ordering of operations.

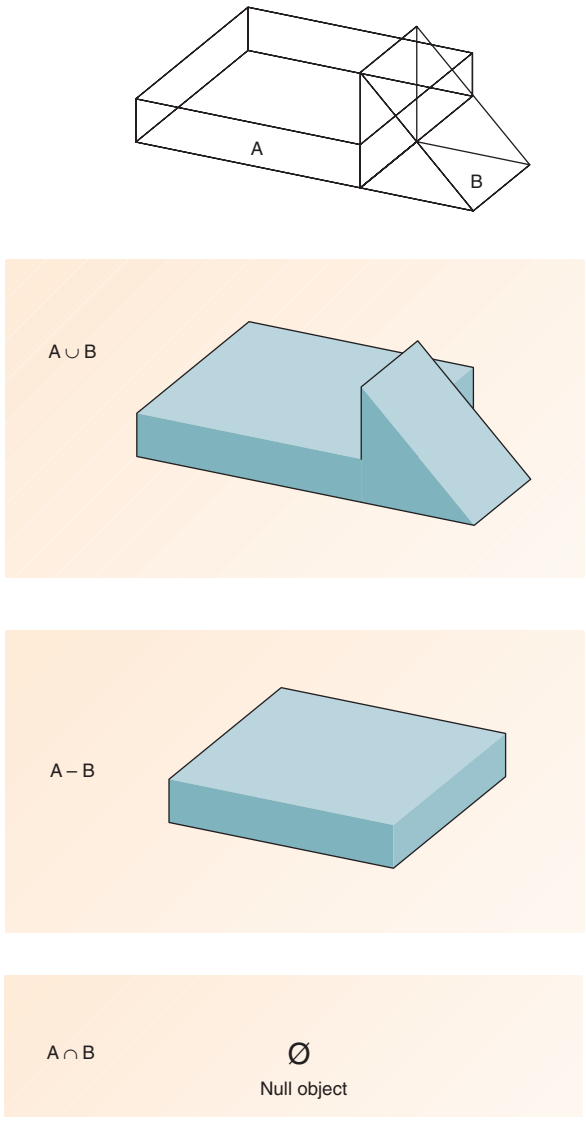


Figure 4.6 **Boolean operations on adjoining primitives**

Only the union operation is effective when primitives are adjoining but not overlapping.

defining a number of primitive forms. The primitives then can be located in space such that they are overlapping or adjoining. Boolean operations can then be applied to create the resulting form. The original primitives may be retained in addition to the new form, or they may be replaced by the new form. More primitives can be created and used to modify the form, until the final desired shape is reached. The use of *sweeping operations* to create primitives can lend even more flexibility to modeling. This technique is discussed in Section 4.7.1.

As with pure primitive instancing, the person doing the modeling must have a clear idea of what the final form will look like and must develop a strategy for the sequence of operations needed to create that form. Often pictorial sketches can be used to plan the sequence of operations leading to the final form.

Practice Exercise 4.1

Using clay or foam, create three pairs of primitive shapes, such as cylinders and rectilinear prisms. Sketch a pictorial of two of the primitives overlapping in some way. With the model pairs, create a single object that reflects the three Boolean operations: union, difference, and intersection.

4.4 Boundary Representation (B-Rep) Modeling

With CSG modeling, surfaces are represented indirectly through *primitive solids*. With **boundary representation (B-rep) modeling**, the surfaces, or faces, are themselves the basis for defining the solid. A B-rep face explicitly represents an oriented surface. There are two sides to this surface: one is on the inside of the object (the solid side), and the other is on the outside of the object (the void side).

Also, like wireframe modelers, B-rep modelers have the capability of containing both linear and curved edges. Supporting curved edges also means supporting curved surfaces, which hinders model performance. For that reason, many modelers often approximate curved surfaces with a series of planar ones. This is called faceted representation (Figure 4.7).

The shape of a B-rep model is created using **Euler operations**. These operations are very similar to those used in mechanical drawing and wireframe model construction. Vertices and edges are added or subtracted from the model according to strict rules to define new faces. Building a solid model one vertex at a time is cumbersome. Therefore, in most modelers, B-rep primitives are often provided for use in creating or modifying groups of faces.

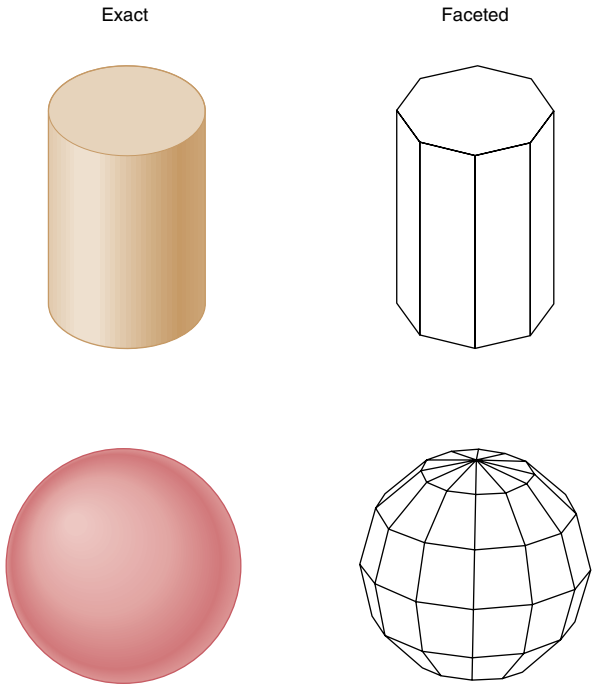


Figure 4.7 **Exact versus faceted surfaces**
Some modelers approximate curved surfaces with a series of planar ones called facets.

4.5 Constraint-Based Modeling

Although the use of 3-D solid modeling grew steadily during the 1980s, companies were not realizing many of the productivity gains promised by CAD vendors. One of the reasons was that the process of creating a solid model was much more abstract than the process of designing real-world products. It wasn't until Parametric Technologies Corporation released Pro/ENGINEER in 1988 that many of the productivity tools now considered commonplace in modeling systems were brought together into a commercial software package.

Among the key innovations which Pro/ENGINEER and other packages have brought to 3-D solid modeling is the idea of having the model defined as a series of modifiable **features**. Each of these features would be defined through operations (described in detail later in the chapter) which as closely as possible represented design or manufacturing features of the final product. For example, a feature might be a hole bored through the model or a fillet added to an interior corner. Each of these features can be created independent of other features or linked so that modifications to one will update the others. The geometry of each of these features is controlled through modifiable **constraints**, creating a dynamic model that can be updated as the design requirements changed. This style of modeling extends to assemblies too. Constraints also are used to bring parts and subassemblies together to represent the final product assembly. Modifications to geometry in a feature are reflected in both the part containing the feature and assemblies containing this part.

4.5.1 Planning

A critical part of constraint-based modeling is the planning that happens prior to building the model. Because

much of the power of constraint-based modeling comes from the ability of the user to modify and otherwise manipulate the features that make up a part, careful planning is needed up front. Careful planning means that the model can be modified later by the person who created it or by others into a new design with a minimum of effort. What constitutes a feature and how it is defined are discussed in detail later in the chapter.

Critical early questions to ask before creating the model are where the model data is coming from and how the model data is going to be used—both in the short term and in the long run. For example, the model might be used exclusively for the generation of exploratory design ideas in the ideation stage of the design process. If so, then there may not be the need to carefully construct the model using features that accurately represent the final manufacturing processes. In addition, the operator may be less concerned about constructing and documenting the model so that other operators are able to modify the model later on. On the other hand, the model might be of an iterative design of an existing product. If so, an existing model may be used and features on the model modified to represent the new design. In this case, the designer hopes that the model has been carefully constructed and documented so that the model *behaves* as expected when features are modified. This behavior should reflect the **design intent** of the product being modeled. That is, changes in geometry of a feature should create model feedback or further changes in the model which reflect design performance or manufacturing constraints of the product.

4.5.2 Sources of Data

Sources of model data vary greatly from company to company and project to project. If the model is of a brand-new design, then all the model data that exists may be rough sketches created by the modeler or another designer. If the company recently has switched over from a 2-D CAD system, then 2-D CAD drawings may be the source of model building data. In this case, you have accurate dimensional information, but often very little of the actual electronic data in the CAD file can be reused to create the model. If the company has switched 3-D modeling systems, then there may be an existing model, but it may be that little of the feature definition or constraint information can be carried over. The best situation, of course, is if you are able to reuse a model created in the same modeling system you are currently using.

The reuse of existing models is an important benefit to using a constraint-based modeler. Quite often, the time that it takes to build a model of the part from scratch is no quicker than it would be to create a set of 2-D CAD drawings of the part. If the model is constructed in such a way as to link dimensions of features together to automate modification of the model, then the time it takes to build the model increases. This time put into model construction will pay off only if you are able to treat the model as a dynamic product data source, which provides a high degree of automation to the creation of alternative designs. This automation is not just within individual parts, but between parts in an assembly.

4.5.3 Eventual Model Use

Another important part of the planning process is understanding how the model data is going to be used once it is created. If you are going to use analysis tools such as finite element analysis (FEA), you will need to make sure that the critical features you have earmarked for careful analysis are modeled in enough detail to give accurate results. Similarly, if you are going to be creating physical prototypes using rapid prototyping tools for visual analysis, careful attention will need to be paid to the visible exterior surfaces. Models used to generate CNC or related manufacturing data will need to accurately represent the geometry of the final manufactured parts, inside and out. For example, internal ribs, bosses, fillets, or draft angles that might not have been of importance to evaluating its external appearance are critical when cutting injection molds.

4.5.4 Modeling Standards

Finally, the internal standards developed by your company need to be followed. Just as with 2-D CAD drawings, models made to standard will be easier to modify and document. As was mentioned earlier, the value of the 3-D constraint-based model increases considerably if it easily can be modified by anyone who needs to generate alternative designs from the model. Standards for modeling may include what geometry should be grouped together to define a feature, in what order features should be created, how the features should be linked together, and how automation features should be documented. Similarly, new or modified features that are added to an existing model should be equally well constructed and integrated so that the next operator who uses the model can also easily update it.

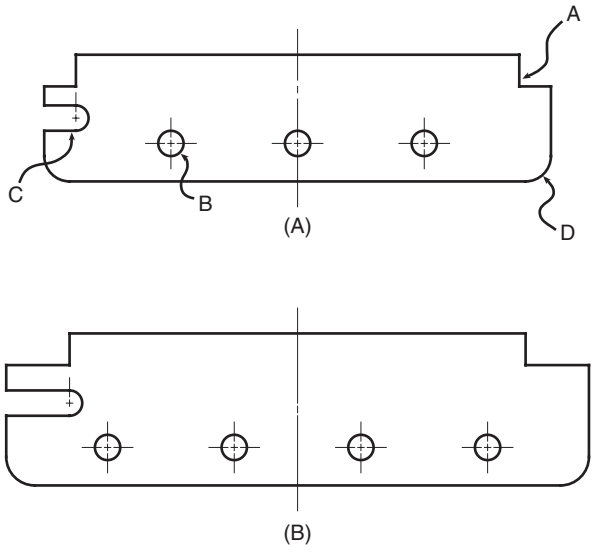


Figure 4.8 **A part defined by features**

Modification of part geometry is done by changing the size, location, and number of features in the part.

4.6 Feature Analysis

Capturing design intent in a model is a process of defining features and the relation of features within a model. The goal is to make sure that information extracted from the model or modified versions of the model for use in other parts of the product development process accurately reflects (as much as is possible) the original intent of the designers and engineers who specified the requirements of the product. 3-D modeling is a process of transforming product requirements into geometry. In a constraint-based model this geometry is dynamic, since the size and location of features in the model can easily be changed to alter the model's geometry.

The part in Figure 4.8 has embedded in it a number of design requirements through the constraints attached to features. Looking in Figure 4.8A, the notches on the top at either end (labeled A) are constrained to be horizontally symmetric in size and location. The holes (labeled B) are constrained to be equally spaced across the length of the part with the holes no farther than 4 cm apart. The slot (labeled C) is constrained a constant distance from the top of the part and has a depth equal to the depth of the notch. The radiused corners (labeled D) have a radius equal to one-fifth the height of the part. A designer modifies the part in two ways: the overall length is increased from 20 cm to 23 cm, the height is increased from 5 cm to 6 cm, and the right-hand notch is doubled in depth. How does the constrained part model respond to these changes? The modified part (Figure 4.8B) shows a number of changes: First, the left-hand notch doubles in depth to match the notch on the right. Second, there are now four equally spaced holes instead of three. Third, the slot is still the same distance from the top of the part, but it has increased in depth to match the new notch depth. Finally, the radii of the corners have increased to stay one-fifth the height of the part.

Figure 4.8 shows examples of a number of types of constraints that can be embedded in a single part. In reality, most parts exist as part of larger assemblies. Feature constraints can also be carried across parts in an assembly to make sure that changes in one part are accurately reflected in dependent parts. Figure 4.9 (on page 142) shows the same part seen in Figure 4.8 as part of an assembly. A change in the first part causes modification in the second part so that the two parts still fit together properly. Besides using the flat mating surfaces of the two parts, the line of symmetry down the middle of the part is used to help center the parts with respect to each other.

Decisions about how to constrain features begin with defining what geometry of the part will be contained within each feature. If the geometry of the part is already well defined, then the decision largely will be one of decomposing the part geometry into a series of smaller geometric elements that can be created with a single feature operation in the modeler. For the part seen in Figure 4.8, the notches, holes, slot, and radiused corners all could be defined as separate features in the part model. In this case, the building of the model would actually begin with a **base feature** represented by a rectangular prism. On this geometry, notches, holes, etc., would be defined as features removing material from the base feature. At the other extreme, the final part seen in Figure 4.8 could be built all as one feature (you'll see how later on). Which is the right way of building the model? There is no easy answer to this question. In fact, depending on the situation, there may be a half dozen or more correct approaches to building this model.

How you define what geometry makes up a feature depends on a number of factors, including:

- How much of the final geometry of the part design has been decided on before you begin creating the model?
- What level of detail is needed in the model?
- What level of modification automation is going to be built into the model?

DREAM HIGH TECH JOB
Designing Snowboards

The engineering design process is used in many types of jobs from the design of consumer product packaging to the design of snowboards and related equipment. An understanding of the design process and 3-D solid modeling—along with formal education in a field of engineering—can lead to exciting job opportunities, such as the one described here of an engineer who worked on the design of snowboards.

Snow Sports

"When I was a kid, we called my grandfather 'Fix-it Grandpa.' He could fix anything, and I often followed him around asking questions about everything. I was always fascinated when he took everything apart; only, if I did it, I couldn't always get it back together. Neither of my parents were mechanically inclined, so, when I needed something fixed, I either had to call Grandpa or fix it myself.

"When I was graduating from high school, I wanted to go to college at University of California, Santa Barbara. My dad and I were looking through the college catalog and ran across a picture of a Human Powered Vehicle under the mechanical engineering section. We both agreed that it looked interesting, and I felt confident that I could study mechanical engineering, because I enjoyed math, science, and physics.

Because of my engineering education, I feel that I can solve any problem and can do whatever I want with my life. The education gave me a set of tools to have a successful life.

Stacie Glass
Former Snowboard Design Engineer, K2 Snowboards
(Courtesy of Stacie Shannon Glass.)

To make extra money, I worked for Joyride Snowboards as a college sales rep. I had been an avid snowboarder for the last 12 years so it seemed like a good fit to use my engineering skills to further the sport of snowboarding.

"My employment at K2 started as an internship after graduation and eventually became a full-time gig. At K2, I designed snowboard footprints, profiles, and constructions with an emphasis on women's boards. I also organized and led on-snow tests on Mt. Hood for prototype testing. My design, the K2 Mix, is still in production and was ranked in the Top 5 Women's boards in the 2002 Transworld Buyers Guide. In fact, Gretchen Bleiler, the winner of the Women's Superpipe in the 2003 X-Games and the Women's U.S. Open Halfpipe Championships, rides my board!"

Skis and Snowboards

Engineers who love to ski and snowboard naturally gravitate toward work in the snow sports industry. Traditionally, when an idea for a new ski or snowboard design came along, engineers would build a prototype, perform laboratory tests for stiffness, and test it on the slopes. Based on the test experience, engineers would make design changes and retest the equipment. This method of design resulted in a slow and tedious process. In addition, the perfectly crafted ski or snowboard is not perfect for everyone. The needs of a 5′2″ female snowboarder are much different than the needs of a 6′0″ male snowboarder. The snowboarder's height, weight, and skill level, as well as the snow conditions and the angle of the slope, all need to be taken into consideration when trying to fit the perfect board to the enthusiast.

Snowboards are made out of several layers of materials, along with glue and paint. Snowboarders believe that the edge design, or effective edge, is the most important part of the design. Edge design determines how the snowboard will turn. The more surface area the edge has, the more control and, hence, the sharper the turns that can be made. Structural strength of the snowboard is also very important. Engineers determine the strength by figuring out the acceleration of the rider.

To accommodate these various conditions, engineers from manufacturers such as K2 and Head are designing intelligent technology that will enable skiers and snowboarders to go faster and have more control.

(Courtesy of Stacie Shannon Glass.)

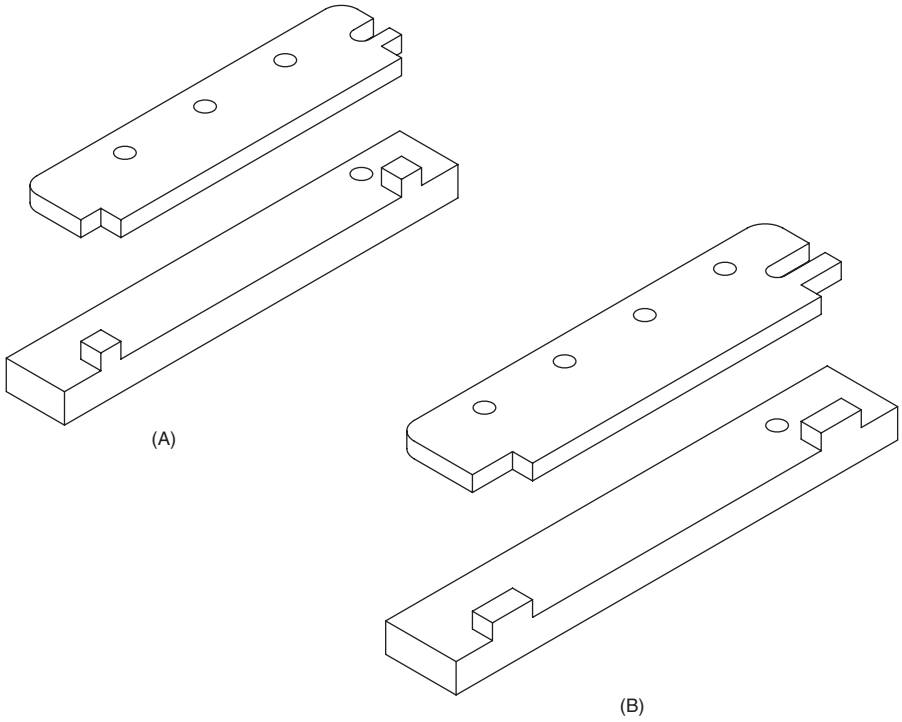


Figure 4.9 **Features related across parts**
Features in a part are often related to mating features on other parts in an assembly.

- What need is there to explore design alternatives that involve the addition and removal of geometric elements, rather than simply changing their sizes?
- Should the geometry be grouped according to the functional elements of the design?
- Should the geometry be grouped based on the manufacturing processes being used to produce the part?

Often it is many of these factors that influence the decisions on feature definition.

4.7 Feature Definition

Knowing how to define features in the model begins with understanding how your modeler allows you to create and edit geometry. Though every constraint-based modeler has its own approaches to feature creation, there are ways of generalizing this process across modelers to understand some of the basic tools used in modeling. This first section will present an overview of this process. Each of these steps in the process then will be treated in greater depth in later sections.

4.7.1 Features from Generalized Sweeps

Many features in a model can be made through the use of **sweeping operations**. Most CAD systems use methods of automating solid feature generation. In a sweeping operation, a closed polygon, called a **profile**, is drawn on a plane and is moved or swept along a defined path for a defined length. Each swept profile can be visualized as a solid object. The first feature, the base feature, will look exactly like this visualized solid. Each successive feature after this will modify the existing geometry based on how the swept form intersects with it and whether the feature is to add or subtract material from the part model.

In the simplest case, the path of the sweep is linear, and a prismatic solid is created (Figure 4.10). If the linear path is coincident with the W axis, a right prism is created. If the path is at any other angle to the W axis, an oblique prism is created. A path parallel to the U–V plane is not allowed because it creates a form that is not three-dimensional. Another path a sweep could follow is circular (revolute). The specifications for a revolute path are more demanding than those for a linear path. With a

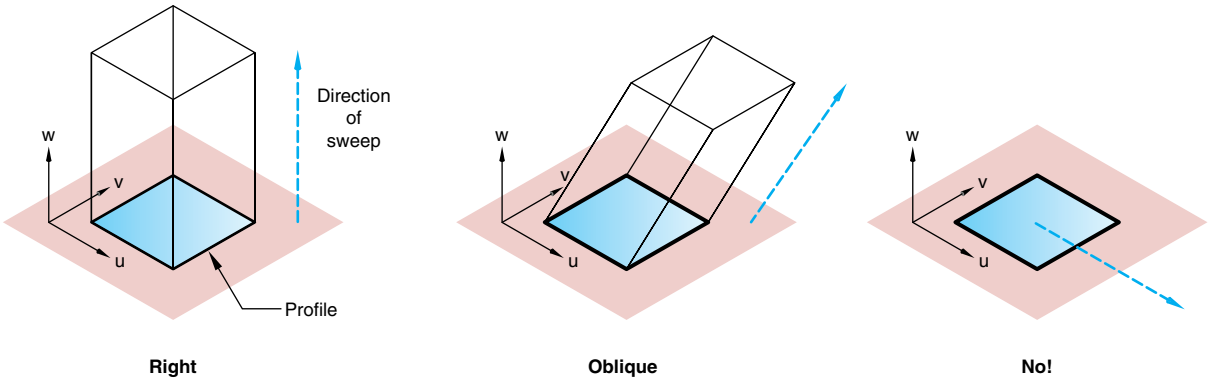


Figure 4.10 **Types of linear sweeping operations**
In some systems, linear sweeps are restricted to being perpendicular to the workplane.

revolute path, an axis of rotation must be defined in terms of both orientation and location. Figure 4.11 (on the next page) shows examples of revolute sweeps.

If the features being created in Figures 4.10 and 4.11 were base features, then the resulting model would look exactly as the geometry is shown. If they are later features, then whether a swept form is adding or subtracting volume from the current model has to be defined. With constructive solid geometry (CSG) modelers, Boolean operations would have defined the interaction between the existing model and the swept profile. With constraint-based modelers, the term Boolean typically is not used, although the description of Boolean operations in Section 4.3 helps to explain how the geometry is being modified.

Step by Step: Creating a 3-D Model Using Sweeping Operations

Sweeping operations can be used to define many of the features in a model. Swept objects can represent both positive and negative geometry, and so they can be used to either add or subtract volume from the model (Figure 4.12 on page 145).

Step 1. With a 2-D rectangular polygon defined on the workplane, determine the direction and distance for the sweep to produce the base feature.

Step 2. Using the top face of the initial solid as a guide, relocate the workplane, create a semicircle, then sweep it to create the half cylinder. Using an addition operation joins these two objects.

Step 3. Rather than using an existing face on the solid for orientation, rotate the workplane 90 degrees so that it is vertical and create a triangle. Sweep the triangle to create a wedge and unite it with the model.

Step 4. Translate the workplane so that it is on the front face of the object. Create a circle on the front face and sweep it to the rear face of the object to form a cylinder. Using a subtraction operation, subtract the cylinder from the object to make a hole through the model.

Step 5. Do a final sweep with a rectangle following a circular path for 180 degrees. Use a subtraction operation to remove a portion of the bottom of the model.

4.7.2 Construction Geometry

All geometry in a model must be located and oriented relative to some 3-D coordinate system. Typically, **world coordinate system** is explicitly defined and available for the operator to use in defining the location of geometry. Refer to Section 3.3 for more information on coordinate systems.

A **workplane** is the most common type of construction geometry used to support the creation of part geometry relative to the world coordinate system. **Construction geometry** does not represent any of the final geometry representing the part, but instead provides a framework for guiding the construction of this part geometry. A common method for starting a part model is to create a set of three mutually perpendicular planes all intersecting the origin of the world coordinate system (Figure 4.13A on page 145). These planes can be thought of as construction geometry used to support the creation of model feature geometry.

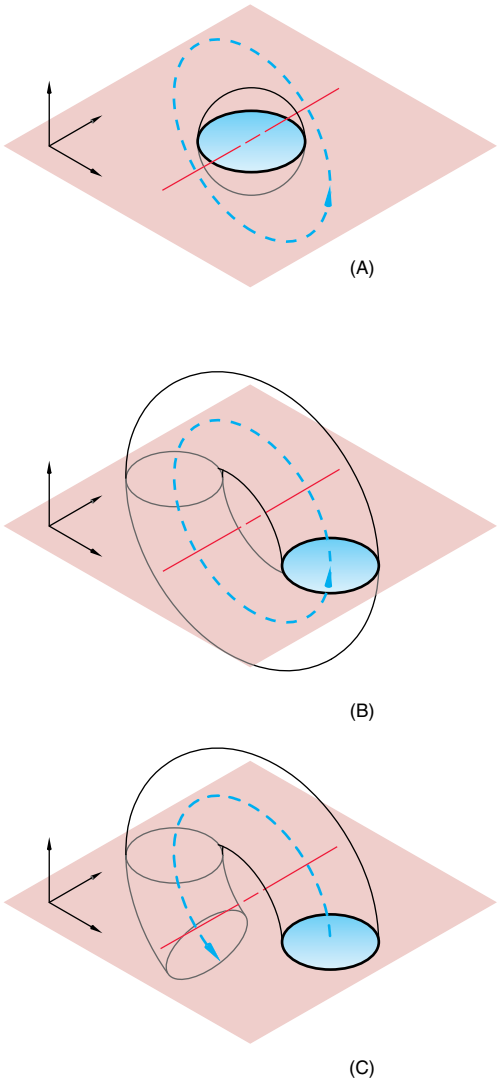


Figure 4.11 **Examples of revolved sweeping operations**
The resulting geometry is dependent on the location of the axis of rotation relative to the profile and to the angular displacement.

A workplane can be used in the same manner as a drawing surface. In a modeler, workplanes typically are used to orient the profile sketch used in feature generation. By adjusting the view of the model to be normal (perpendicular) to the workplane, you can draw on the workplane as though you were looking directly down on a piece of paper. Figure 4.13B shows a sketch created on one of the workplanes. The workplane orients the sketch relative to the world coordinate system while dimensional constraints locate the sketch on the plane relative to the other two workplanes. This profile sketch might have been drawn in this pictorial view, or the view may have first been oriented normal to the workplane (Figure 4.13C).

Each of these infinitely large planes creates an implied *local,* or **relative coordinate system**. An alternative coordinate system is often implied—for example U, V, W—to indicate the orientation of the plane relative to the world coordinate system. In Figure 4.14 (on page 146), U and V are in the plane of the workplane while W is normal (perpendicular) to the workplane. As mentioned before, the sketch geometry on these planes typically is located relative to projections of construction geometry or part geometry onto this plane.

Once the base feature is created, workplanes often are oriented using geometry of the model being built. The simplest way to locate a workplane is to make it coplanar with an existing face of the model (Figure 4.15A on page 146). In addition, three vertices (V_1,V_2,V_3) can be specified to orient the workplane on the model. For example, the first vertex could be the origin, the second could show the direction of the U axis, and the third could be a point in the first quadrant of the U–V plane (Figure 4.15B). An alternative would be to specify an origin at a vertex and then the edges on the model to orient the plane.

Common methods for specifying the location and orientation of workplanes include (Figure 4.16 on page 147):

- Through (includes coplanar)
- Offset/parallel
- Angle
- Point or edge and orientation
- Tangent and orientation

Often more than one of these specifications needs to be used to unequivocally specify a new workplane.

In addition to workplanes, construction axes and construction points can be created. Construction axes are often used to locate an axis of revolution or locate a point where the axis pierces a plane. Construction points can be used to locate a specific point along an infinitely long construction axis or a specific point on a construction plane or face of the model. An understanding of how the fundamentals of geometry are defined (see Chapter 3) is critical to understanding how construction geometry is created and manipulated.

4.7.3 Sketching the Profile

Many features on a part model begin as a **profile sketch** on a workplane. Once a workplane is chosen or created, a

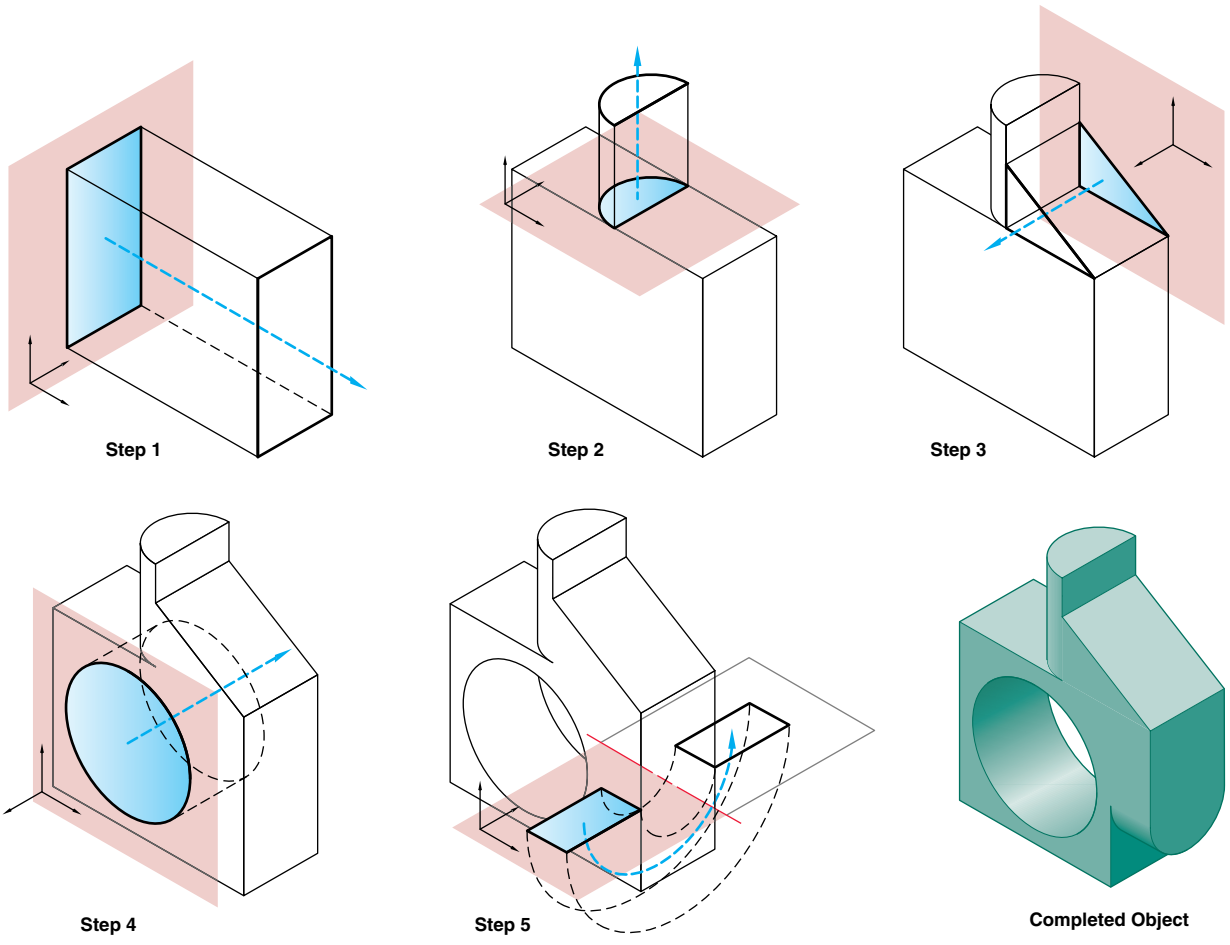


Figure 4.12 **Creating a solid model using sweeping and Boolean operations**

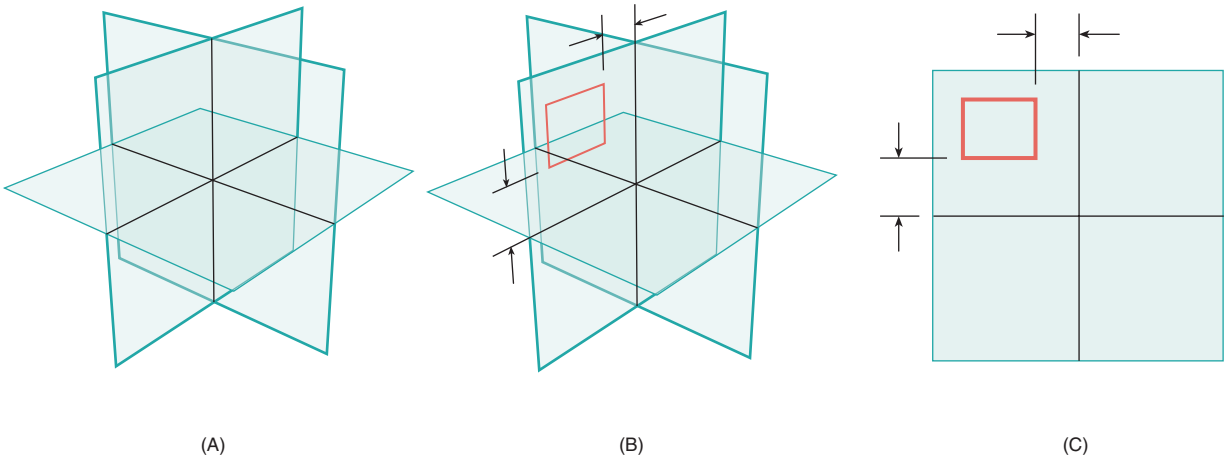


Figure 4.13 **Mutually perpendicular workplanes**

Three mutually perpendicular workplanes are often used as a starting point for defining the first feature of a part model.

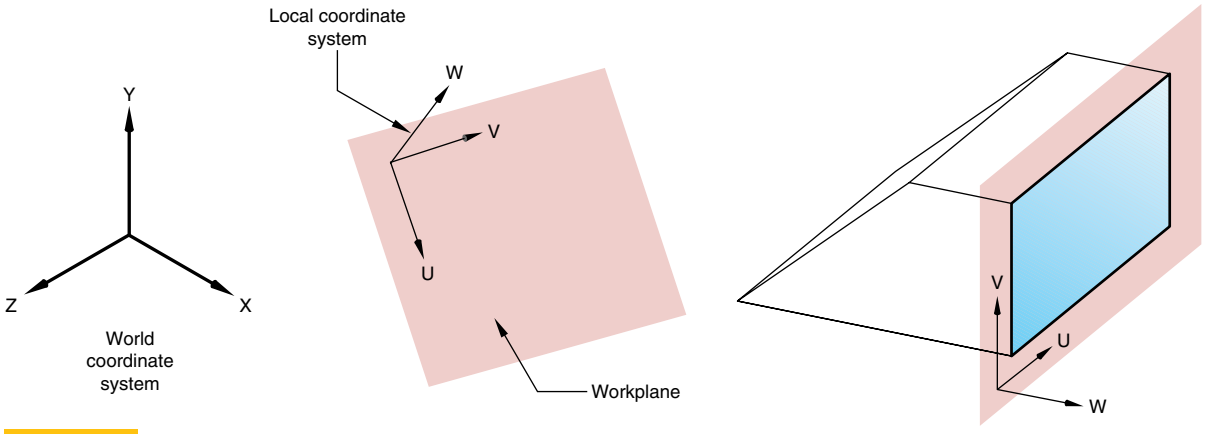


Figure 4.14 **A local coordinate system attached to a workplane**

Local coordinate systems are used to locate a workplane relative to the world coordinate system or the model.

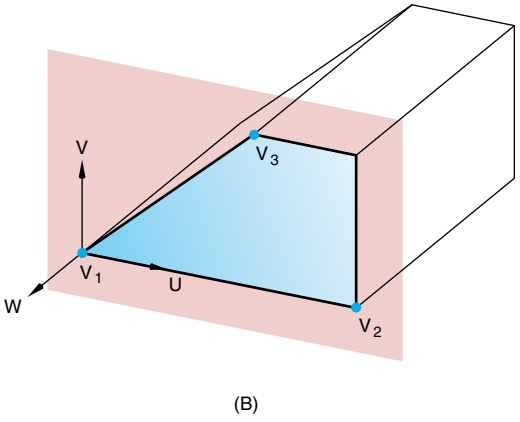


Figure 4.15 **Locating a workplane by the features of the model**

Both faces and individual vertices can be used to orient and locate workplanes.

decision has to be made as to how to view the workplane while sketching on it. If a pictorial view is chosen to work in (Figure 4.13B), you have the opportunity to get an overall view of where the sketch is relative to the other part geometry. On the other hand, a view normal to the workplane (Figure 4.13C) will likely give you a more precise view of how part geometry projects onto the sketched profile. For more complex profile sketches, you may want to shift between multiple views of the model and workplane.

Once a view of the workplane is established, a profile sketch can be drawn on the workplane. This sketch will consist of a series of line elements such as straight lines, arcs, circles, or splines. Tools used for drawing this sketch will be very similar to the tools used for drawing such elements in a 2-D CAD system. One important difference concerns the accuracy with which the sketch needs to be drawn. Unlike a 2-D CAD drawing, the sketch does not need to be dimensionally accurate. Instead, the sketch represents the overall shape, the **topology**, of the profile. That is, the sketch should represent the total number of sides of the final profile, the basic shape of the elements (curved or straight), and the order in which the elements are connected together. The sketch also should represent the basic geometric relationships between the elements (parallel, tangent, etc.) within a reasonable level of accuracy. This level of accuracy will be discussed in Section 4.7.4.

Depending on the type of modeler used, other characteristics of the profile may need to be considered. For example, the sketch might be either a **closed loop** or an **open loop**. A closed loop sketch has its last element connected with its first element to create a sealed path (Figure 4.17A). You could imagine that water poured inside the loop would not leak out. An open loop sketch does not close back on itself and is used when fewer elements can clearly indicate the action of the sketch profile (Figure 4.17B). Whereas a closed loop implies an *inside* and an *outside,* an open loop does not. When a profile sketch contains more than one loop, the loops usually all need to be closed to clearly indicate what is inside and outside (Figure 4.17C).

The definition of inside and outside is needed to specify how the profile is to interact with the existing geometry. For example, in Figure 4.18A (on page 148), the material inside the loop is subtracted from the existing object,

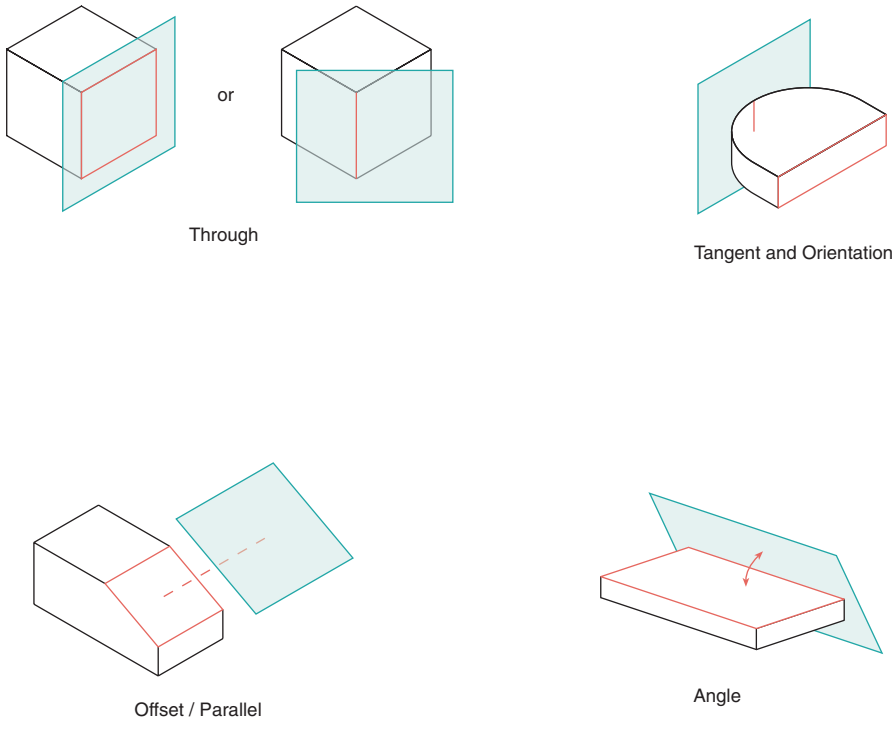


Figure 4.16 **Common methods of creating new workplanes**

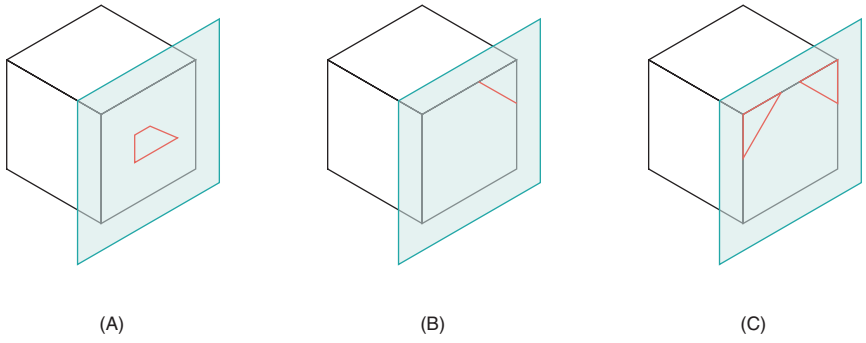


Figure 4.17 **Closed and open loop profile sketches**

Depending on the modeler, profile sketches can be either open or closed.

whereas in Figure 4.18B, the material on the outside of the loop is subtracted from the existing object.

4.7.4 Constraining the Profile

Going hand in hand with the sketching of the profile is applying **constraints**. Whereas the sketching defined the topology of the profile, constraining defines the **geometry** of the profile. This use of the word *geometry* refers to a more narrow definition of the term: the size, location, and orientation of geometric elements that give them an overall shape and form. The types of constraints applied to the sketch profile can be roughly divided into two categories: **explicit** and **implicit**. These two types of constraints differ

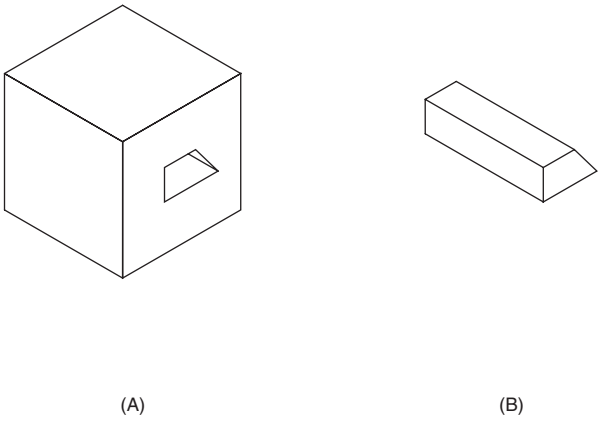


Figure 4.18 **Side of profile sketch**

When subtracting material from a part, whether the inside or the outside of a profile sketch is chosen will make a difference to the end result.

as to whether the modeling system *infers* the constraint based on the way the sketch was drawn, or whether the operator has to *explicitly* apply the constraint to the sketch.

Many systems create constraints based on the implied geometric relationships in the profile sketch. Common geometric relations for which the system might create implied constraints include (Figure 4.19):

- Closure (connected edges)
- Segment overlap
- Endpoint/line overlap
- Tangency
- Parallelism, perpendicularity
- Same size
- Coincident (but not touching)
- Concentric

These relationships are applied not only internally within the profile, but also between elements of the profile and existing geometric elements in the part model. For example, segment overlap is applied exclusively between a profile element and part geometry.

When a system applies these implicit constraints will be determined in part by a predefined tolerance. This tolerance will decide, for example, when two sketched lines are close enough to parallel that they should be constrained parallel. How does the system set this tolerance? In some cases it might be by predefined values. Two lines would be inferred to be parallel or perpendicular if they were within 5 degrees of these respective orientations. Other systems use a tolerance based on view resolution. That is, if two lines look as though they are overlapping on the screen, then they will be constrained as such. Understanding how a system applies implied geometric constraints is important to devising a sketching strategy.

Though the profile does not need to be sketched with dimensional accuracy, how you sketch it will influence how geometric constraints are applied. For example, if you have two lines that are to be 92 degrees relative to each other, trying to sketch this accurately will likely cause the system to apply an incorrect constraint of perpendicularity. What you would do instead is exaggerate the nonperpendicularity of the lines by sketching them at, say, 110 degrees and then later come back and apply a dimensional constraint to pull the segments back to 92 degrees. If implicit geometric constraints are applied which you do not want, there typically will be a facility for overriding or removing these constraints. Similarly, you also may have the ability to force the application of geometric constraints which were not inferred by the system.

Explicit constraints, unlike implicit constraints, are applied by the user to the profile sketch. The most common explicit constraint is a dimensional contraint. The application of dimensional constraints is very much like applying dimensions in a 2-D CAD system, yet they behave very differently. A dimensional constraint indicates that a specific size or location of a profile element is going to be controlled by a variable, or parameter. With a traditional CAD modeler, geometric elements are created at a specific size and location (Figure 4.20A on page 150). For example, if a plate is to have a width equal to 32 mm and a height of half the length, a rectangle 32 mm wide and 16 mm tall would be created. In a constraint-based modeler, constraints are assigned to geometric elements to control their size or location: the width would be defined as $P_1 = 32$ mm and the height defined as $P_2 = P_1 \div 2$ (Figure 4.20B). Though a profile element initially may be sketched at a specific size (and a constraint assigned this as its initial value), the user can go back and change its definition at any time. The power of this approach is seen when the model is modified. Instead of having to individually update all related dimensions, one dimension can be altered, and all dimensions linked through parameters automatically will reflect the change. For example, if the width of the plate is reduced ($P_1 = 20$), the height automatically reduces by the appropriate amount to reflect the change in width (Figure 4.20C). As seen in this example, the constraint parameter might not be assigned a numeric value, but instead an algebraic relation tied to other constraint parameters.

Just as with dimensions in traditional engineering drawings, dimensional constraints specify either the location or

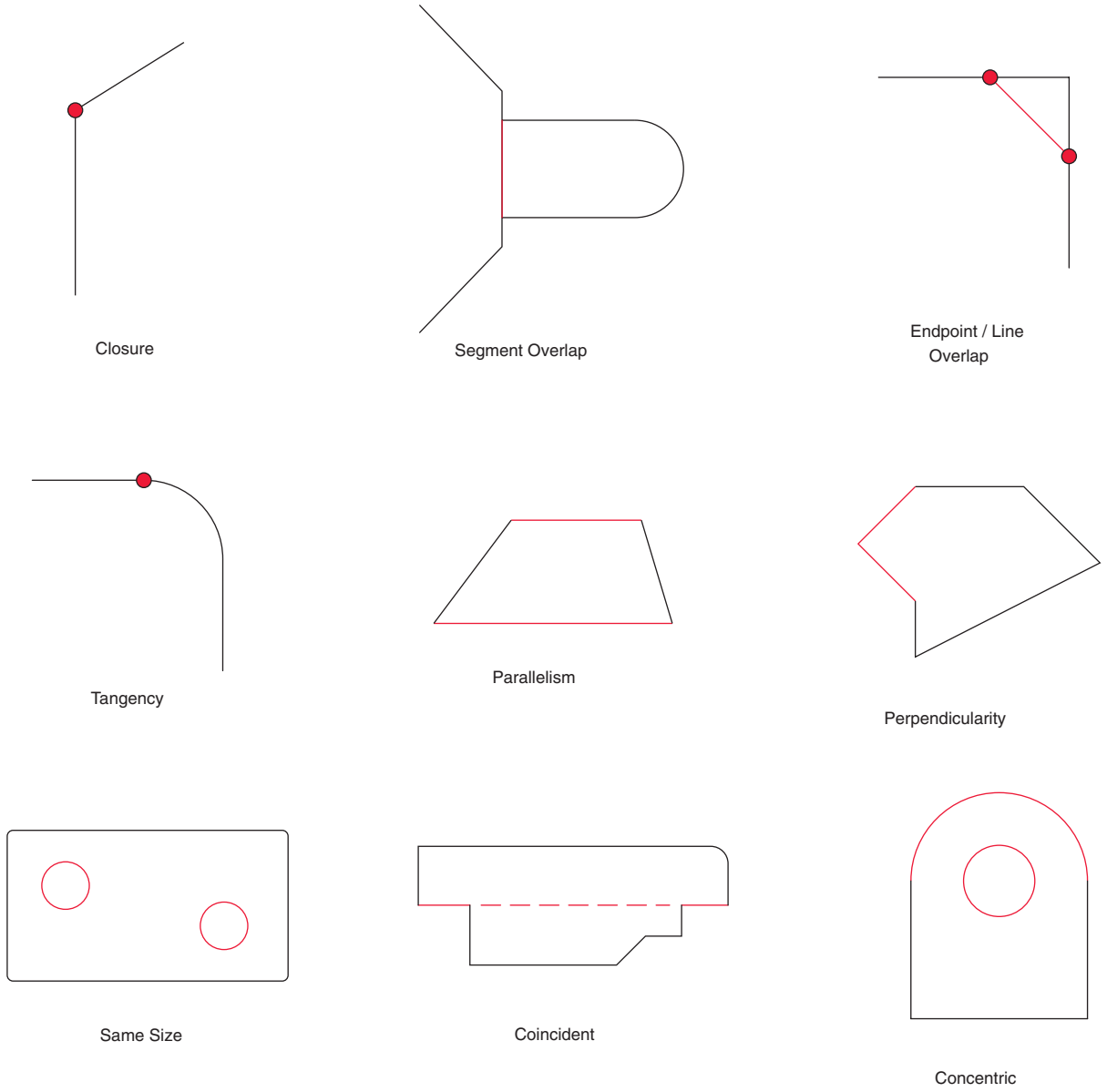


Figure 4.19 **Common geometric relations for constraining a profile**

size of a geometric element (Figure 4.21 on the next page). The constraint may reference elements that are *internal* to the profile, or it may tie profile elements to external part or construction geometry. In Figure 4.21, dimension A represents an internal size constraint and dimension B represents an internal location constraint. Dimension C, on the other hand, is a location constraint with an external reference. When a profile element overlaps an external reference (indicated by D), the system may apply a locational constraint of value zero.

Dimensional and geometric constraints should not be thought of as independent from each other. Figure 4.22 (on page 151) shows how these two different kinds of constraints work together to create the final constrained profile. The initial profile (Figure 4.22A) is sketched to represent the appropriate geometric relations, such as parallelism

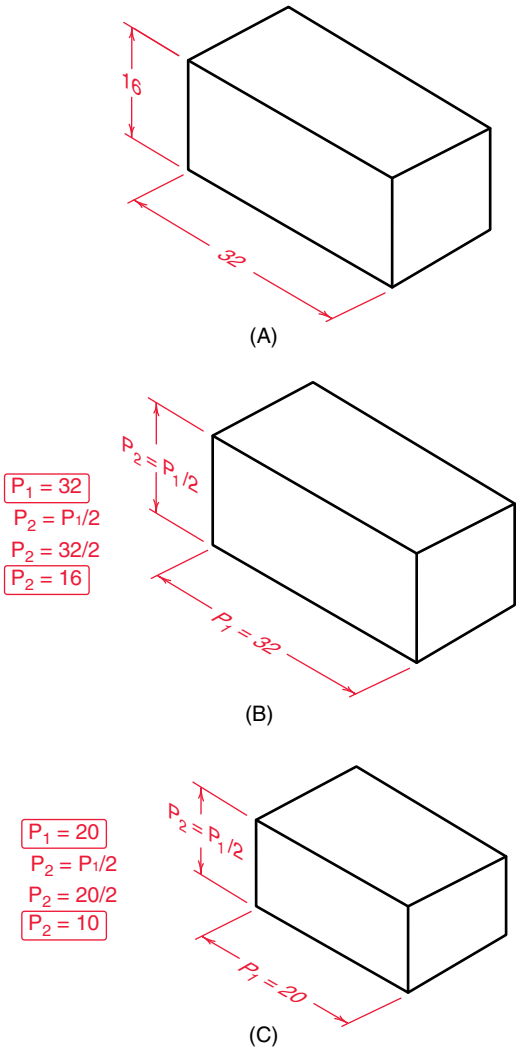


Figure 4.20 **Traditional and constraint-based part definition**

Traditional modelers define geometry directly, while constraint-based modelers use parametric equations.

between the top and bottom elements and tangency with the arc. With the geometric constraints applied, explicit dimensional constraints are applied to control the size of the profile elements (Figure 4.22B). Some of the constraint parameters, such as P_3, are controlled through algebraic relations by other parameters. Once applied, the dimensional constraints easily can be altered to modify the shape and size of the profile (Figure 4.22C). Once constrained and parameter values are modified appropriately, the profile can be swept out (Figure 4.22D).

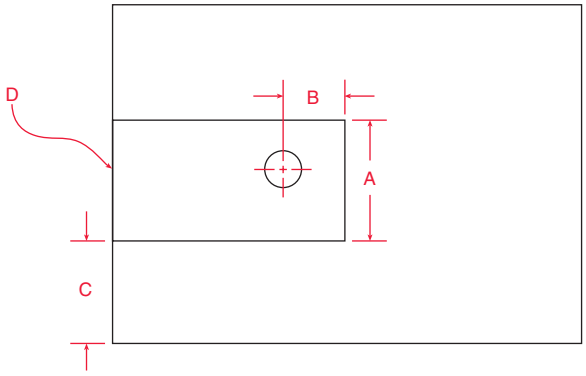


Figure 4.21 **Explicit dimensional constraints**

Dimensional constraints are used to locate and size feature elements.

Not all constraints easily fit into the categories of geometric or dimensional. For example, a modeler may have an offset constraint (Figure 4.23). This constraint allows you to select a group of existing geometric elements, specify a distance and side to offset, and create profile elements constrained to the originally selected elements. This constraint combines implicit geometric relations (the new profile elements stay parallel to the existing elements) with dimensional constraints (a constant offset distance). The size of the profile element ends up being determined indirectly through the connection points of the elements.

Central to developing a strategy for constraining a profile is knowing when the profile is fully constrained, underconstrained, or overconstrained. A **fully constrained** profile has completely specified the geometry of a profile. All elements are sized, located, and oriented relative to each other and to the part model (and, therefore, the world coordinate system). In many cases, the geometry of an element will be determined indirectly based on the size, location, orientation, and geometric relationship of other elements. In contrast, an **underconstrained** profile has one or more elements that have not been fully specified. Underconstrained elements will initially take on the geometric properties represented by the sketch: if it is sketched 94.3 cm long, then it will be represented that way. When constraints on the profile are later modified, the underconstrained elements will not be driven by predefined constraints. This leads both to a freedom and to a certain degree of unpredictability when creating feature profiles. Profiles also can be **overconstrained**. This is generally an unwanted situation, since there are now constraints in conflict with each other with no clear indication as to which should take precedence. You would

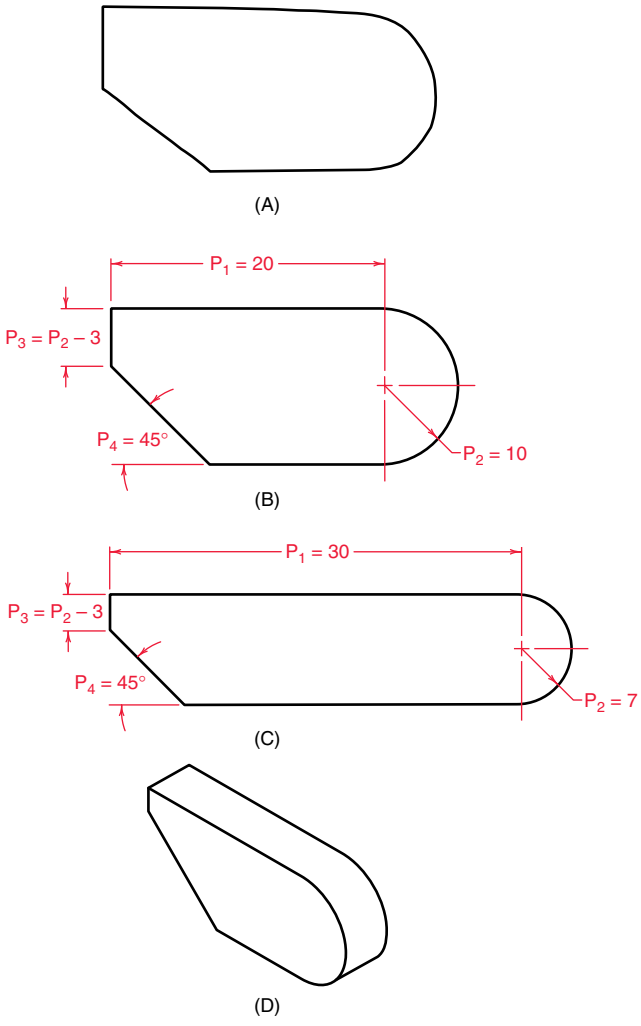


Figure 4.22 **Creating a constrained model from a sketch profile**

A rough sketch is constrained through dimensional and geometric constraints.

need to delete geometric or dimensional constraints on the profile to resolve the overconstrained condition.

As was noted earlier in this section, dimensional constraint parameters can be set to something other than a constant value. The ability to link constraint parameters through algebraic equations or to control values based on logic statements provides tremendous power to the modeler to both embed design intent and automate modifications of the model.

A common type of algebraic formula to associate with a constraint is to have a parameter equal to another parameter plus or minus a constant. This type of *offset* formula can

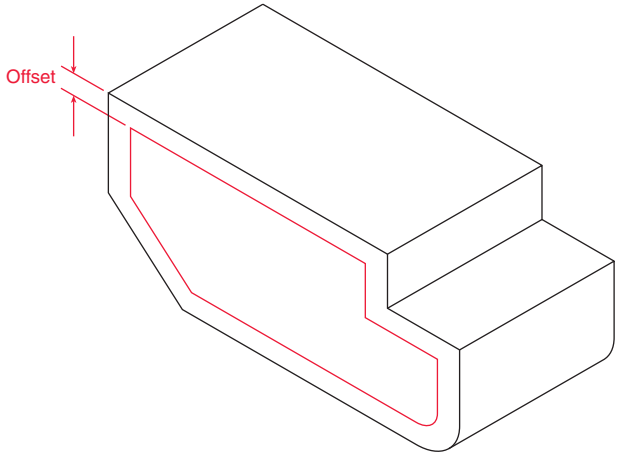


Figure 4.23 **An offset constraint**

Some constraints are not easily categorized as being either dimensional or geometric.

be seen in a constraint in Figure 4.22 where constraint P_3 will always be 3 less than P_2 no matter what the value of P_2. Another type of formula can be seen in Figure 4.20 where constraint P_2 is always a *ratio* of P_1. In this case, P_2 will always be one-half of P_1. It is often necessary to look behind the current value for a constraint parameter to understand the design intent embedded in the constraint. In Figure 4.24A (on the next page), two plates are created with holes and two different design intents. In the plate on the left, the hole was intended to be placed 10 mm from the left edge, while the plate on the right had the hole placed in the center. When the overall width is set to 20 mm, no difference is seen in the two models, but when the overall width is set to a different value, the difference in design intent is immediately seen (Figure 4.24B). With a traditional modeler, an engineer viewing the model in Figure 4.24A would not be able to tell what the design intent was and, therefore, how the hole should shift if the model was altered.

Figure 4.24 is an example of parameters being linked across features in a part. Here, the location of the hole feature is tied to the overall length of the base feature, the plate. In addition to linking parameters within and between features of a part, parameters also can be linked between parts in an assembly. Looking back on the assembly in Figure 4.9 (on page 142), the constraint parameter controlling the spread of the rectangular pins on the bottom piece needed to be linked to the overall length and notch cuts of the top plate. In complex assemblies, creating linkages between constraint parameters that reflect design intent can be very time-consuming. Figure 4.25 (on the next page) shows just a portion of the constraint

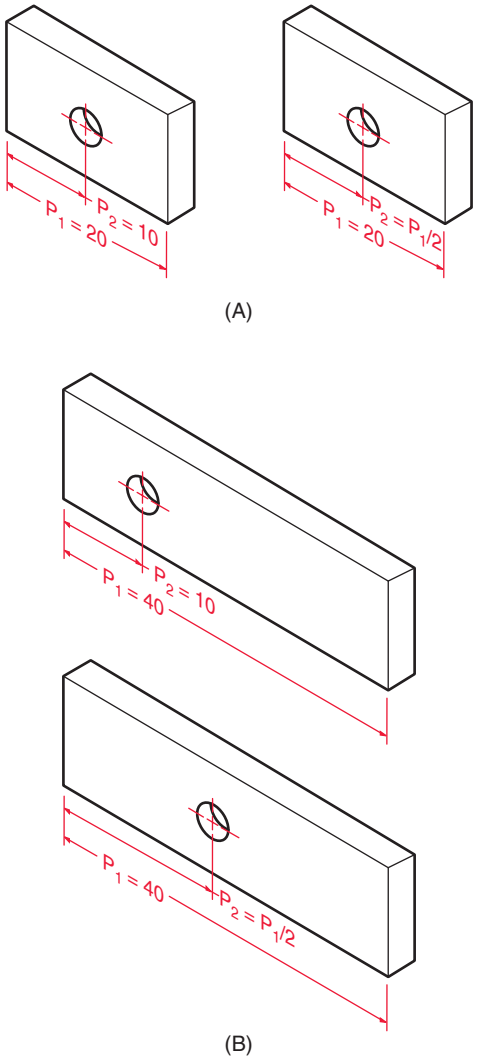


Figure 4.24 **The effect of design intent on model changes**

How a feature behaves when a part is modified depends on how it is constrained.

parameter relations written for a sofa assembly. Savings are seen, however, through increased accuracy and automated updating of parts when component parts in an assembly are modified throughout the design cycle.

4.7.5 Completing the Feature Definition

With the sweep profile drawn and constrained, there remain a few more elements of the sweep which need to be defined. Depending on the modeler, some or all of these definitions may take place prior to or after the creation of the constrained profile.

```
/* global variables
seat_l = 73.5
seat_h = 11
seat_d = 25
splay =5
bo = .5
rail_h = 8
back_a = 14
arm_l = 20
/*set back rail lengths to sofa length
L:0 = seat_l
L:6 = seat_l
/* set front rails to sofa length, splay, and seat depth
L:18 = L:0 + 2*(seat_d * tan(splay))  - 1.516
L:20 = L:0 + 2*(seat_d * tan(splay))  - 1.516
/* set splay cuts on side rails
CUT_ANG1:10 = splay
CUT_ANG1:8 = splay
CUT_ANG1:14 = splay
CUT_ANG1:16 = splay
/* set side rails to sofa depth
L:10 = seat_d / cos (splay)
L:8 = seat_d / cos (splay)
L:14 = seat_d / cos (splay)
L:16 = seat_d / cos (splay)
/* set back skirt rail height
BORE_OFF4_L:2 = seat_h - bo
BORE_OFF3_L:2 = BORE_OFF4_L:2 -(BORE_OFF2_W:0 -
BORE_OFF1_W:0)
ORE_OFF4_L:4 = BORE_OFF4_L:2
ORE_OFF3_L:4 = BORE_OFF3_L:2
/* set side skirt rail height
ORE_OFF1_W:8 = BORE_OFF1_W:0
ore_off2_w:8 = BORE_OFF2_W:0
ORE_OFF1_W:14 = BORE_OFF1_W:0
ore_off2_w:14 = BORE_OFF2_W:0
ORE_OFF1_L:12 = BORE_OFF1_W:0
ORE_OFF2_L:12 = BORE_OFF2_W:0
```

Figure 4.25 **Dimensional constraints controlled through parametric equations**

One part of the sweep definition that still needs to be defined is how the profile is going to be swept out to create a form in 3-D space. When the lines of the profile are swept out from the workplane, the profile creates a surface or set of surfaces. If the profile were an open loop consisting of a single element, say a straight line, and the profile were swept in a linear direction, the resulting sweep would define a plane (Figure 4.26A). If the profile were a closed loop, say a circle, then a linear sweep would create a cylinder (Figure 4.26B). Depending on how the profile is defined, rather than capping the ends to create a solid cylinder, the profile line might be *thickened* to create a tube instead (Figure 4.26C). This thickness option can be used with both open and closed profiles and is used to define sheet metal and other *thin features.*

The direction of the **linear sweep** relative to the workplane will create different sets of extruded surfaces (see Figure 4.10 on page 143). While a sweep of a closed loop profile normal to the workplane will create a right prism, more rarely, an angle other than 90 degrees to the workplane is used to create an oblique prism. In addition to linear sweeps, sweeps also can be circular (revolved) (see

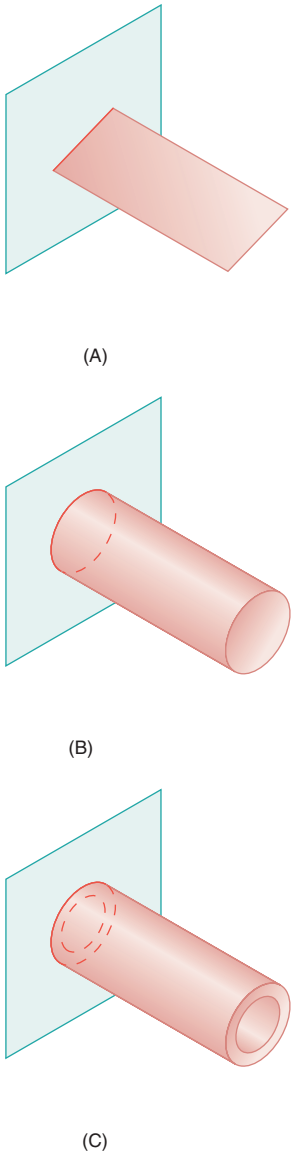


Figure 4.26 **Thin versus solid features**

Depending on how the profile is interpreted, either a thin or solid feature can be created from a profile.

Figure 4.11 on page 144). With a **circular sweep**, an axis of rotation has to be defined. The location of this axis relative to the profile can greatly affect the resulting sweep. Figure 4.11A and B shows two different circular sweeps, each with a different placement for the axis of rotation. In addition, angular displacements of other than 360 degrees can be specified (Figure 4.11C). With both linear and circular sweeps, the sweep can be defined as *one-sided* or *two-sided*. With a one-sided sweep, the profile is swept out only

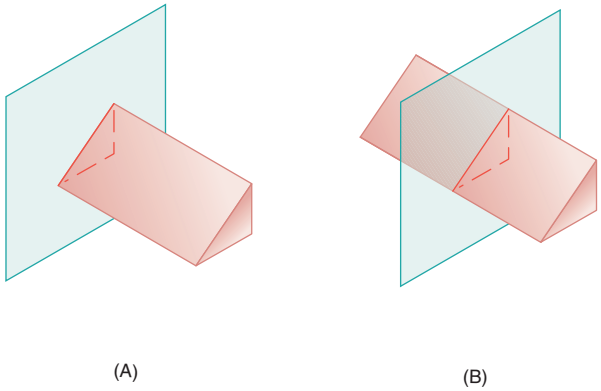


Figure 4.27 **One-sided versus two-sided sweeps**

one direction from the workplane (Figure 4.27A). With a two-sided sweep, the profile is swept out both directions from the workplane (Figure 4.27B).

A less commonly used definition is a **path-based sweep** (Figure 4.28A on the next page). With a path-based sweep, the profile is swept along a path defined either by an existing edge on the part model or by a path drawn by the operator. If the operator draws the path, it typically will have to be sketched and constrained just as a profile would have to be. Finally, some systems will allow you to define multiple profiles on different workplanes. A swept form then is created by connecting surfaces between elements on the different profiles (Figure 4.28B). These loft, or **blend sweeps**, typically have restrictions concerning the orientation of the profiles relative to each other, how they are ordered, and how elements on the different profiles are related to each other.

The distance that a profile is swept can be determined in a number of ways. A **blind** sweep indicates that a finite value is given for the sweep distance (Figure 4.29A on page 155). For a linear, one-sided sweep, this is a linear distance from the workplane to the end of the sweep. For a circular sweep, this distance is given as an angular displacement. For a two-sided sweep, the distance could be given for each side of the sweep separately, or one value could indicate the total distance of the sweep, evenly divided on either side of the workplane. The opposite of a finite (blind) sweep is an infinite **through all** sweep (Figure 4.29B). Sweep distances also can be specified relative to geometry on the part model. A **through next** sweep passes through the next *inside* region, but stops when it hits *outside* (Figure 4.29C). A **to next** (or to surface) sweep passes through an *outside* region, but stops when it hits *inside* (Figure 4.29D).

A central element to all sweeping operations is defining how the swept form will interact with the existing part model. If the sweep is the first part geometry created, then

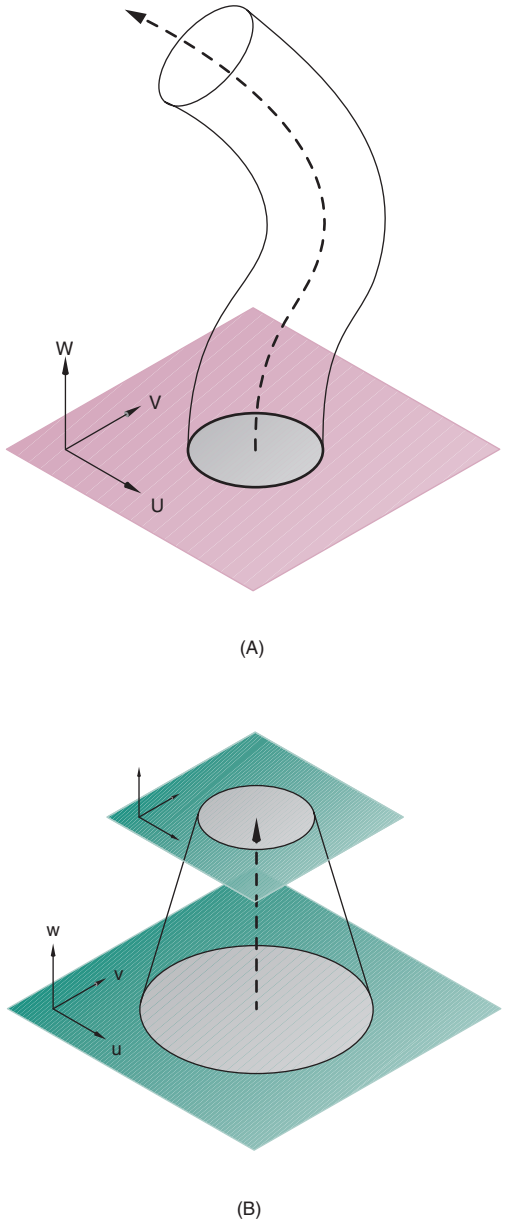


Figure 4.28 **Advanced sweeping techniques**

A path-based sweep moves the profile along a predefined curved path, while a blend sweep interpolates between multiple profiles laid out in space.

it is the base feature and no operation has to be defined. All subsequent features have to either add or remove material from the model. If an open loop profile is going to remove material from the part model, then the *removal side* has to be defined (Figure 4.30). An open loop profile which is not removing material either will be defined as a thin feature or will be attached to surfaces on the part model in a way which allows the new surfaces to form a closed form with the part (Figure 4.31 on page 156). A closed profile clearly defines the inside and outside. When the profile is used for adding material, the addition is done on the inside of the profile. When the profile is used for subtracting material, the user has to specify whether the inside or outside is being removed (see Figure 4.18 on page 148).

Most constraint-based modelers have tools that speed up the definition of commonly used features. Rather than having to define every variable of every feature, options can be given for common design or manufactured features which either have predefined certain feature parameters, bundled variables together in easy-to-use dialog boxes, or otherwise automated the feature definition process (Figure 4.32 on page 156). The ultimate goal, of course, is to make modeling a more efficient process in tune with how designers and engineers actually work.

Examples of manufactured features created through special feature-based dialog boxes include the following:

- Blind and through holes
- Counterbores and countersinks
- Slots
- Bosses

The hole dialog box shown in Figure 4.32 is a good example of automating the process of creating features in a model. The feature is broken down into its essential variables, with each variable represented by an input in the dialog box. Variables such as the hole's diameter have a value typed in, while the depth can be set to "through" by clicking a button or set to a finite value.

The variables entered through the dialog box largely define the shape and size. Once these variables of the feature are defined, the location is defined. A feature such as a blind hole is located by indicating its orientation to a face and distance from two edges (Figure 4.33 on page 156). In a constraint-based modeler, all of the variables of the feature—its shape, size, and location—are controlled parametrically and can be updated at any time. In addition, the parameters defining the feature also can be linked to other parameters defining the part. So, for example, the depth of a hole might be related to the overall thickness of the base part.

4.7.6 Feature Planning Strategies

Though it is impossible to come up with a definitive list of "rules" which should be followed when planning the modeling of every part, there are still certain characteristics of the part geometry that should be evaluated and decisions that have to be made for most parts during the planning process.

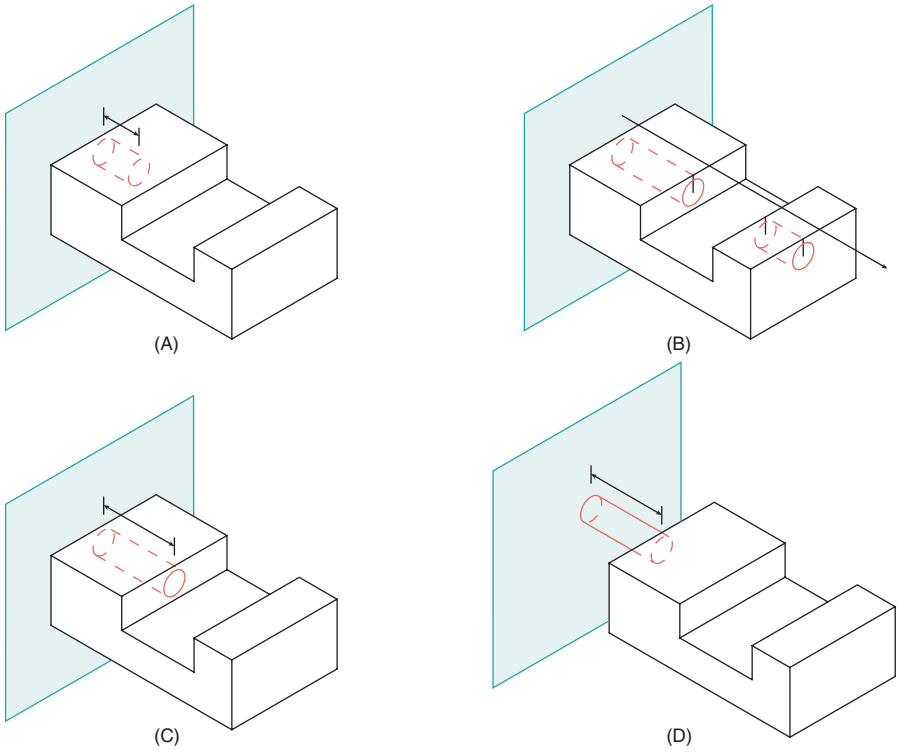


Figure 4.29 **Defining sweep distance**

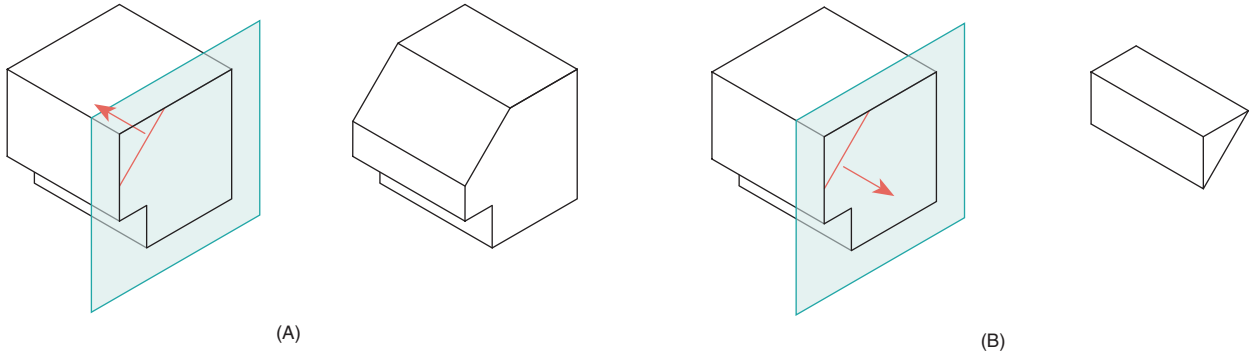


Figure 4.30 **Determining the removal side of a sweep**

Depending on which side of a profile is chosen, different material can be removed from the part.

One of the more important considerations is whether the parts contain lines of symmetry. For example, the best way to leverage the symmetric aspects of the part depicted in Figure 4.34A and B (on page 157) would be to construct the base feature with one of the datums along the line of symmetry. Not only will this assist in the construction of the base feature, but also it will allow mirror commands to be used to duplicate features across the plane of symmetry. Establishing this plane of symmetry, and tying many of the dimensional constraints to this plane, establishes the design intent that this plane of symmetry should be maintained as the part dimensions are modified. This plane of symmetry makes it easier to establish constraints that preserve the symmetry of the part. With two-way symmetry,

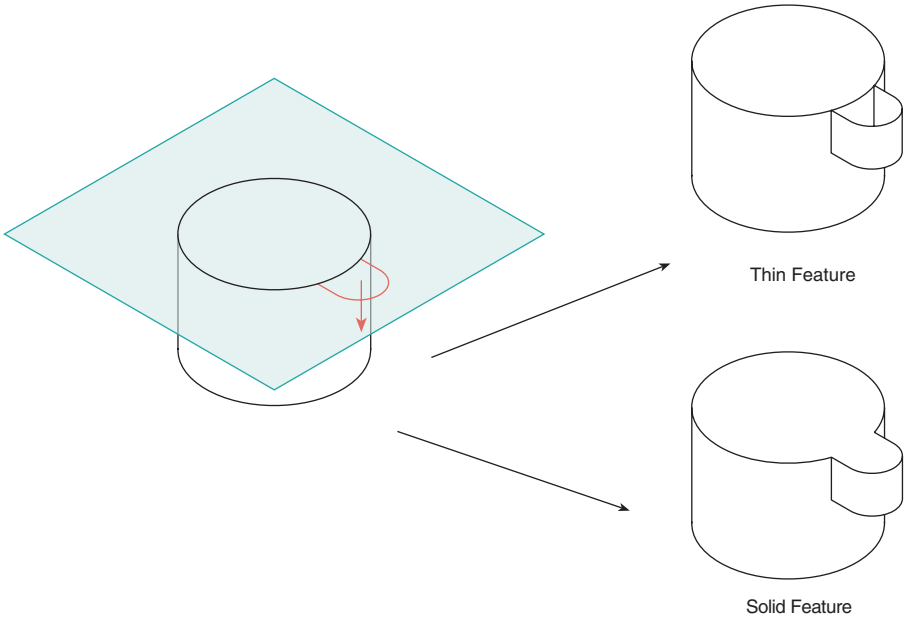


Figure 4.31 **Open loops can define either solid or thin features**

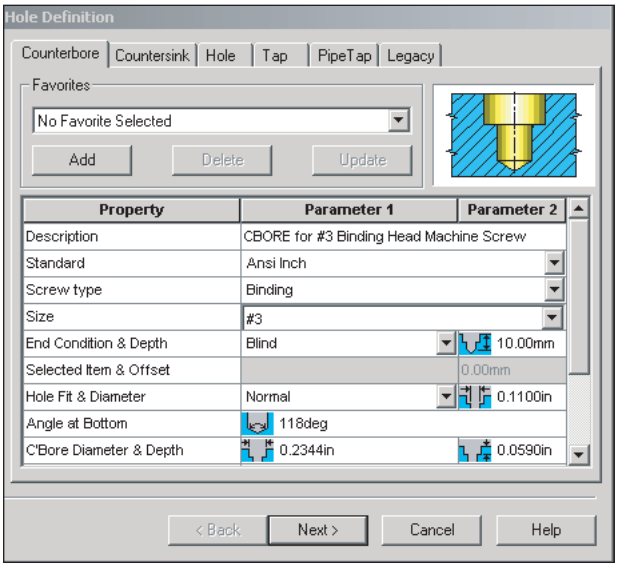


Figure 4.32 **Example of a feature definition dialog box**
The dialog box contains all key parameters needed to define commonly used features, such as holes.

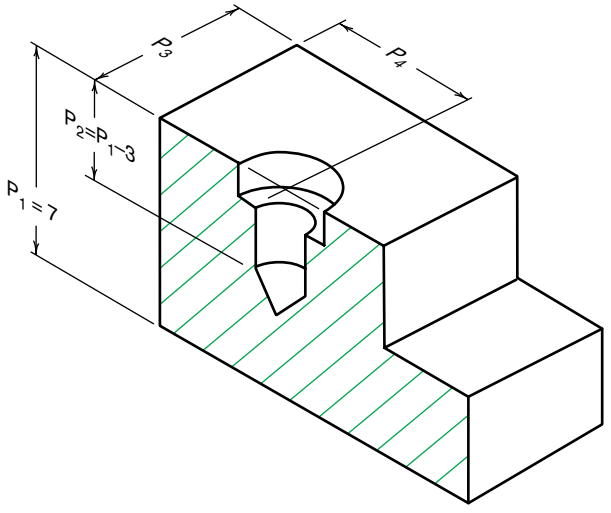


Figure 4.33 **Locating a feature on the base part**
Features such as this counterbore also can be defined relative to the existing part geometry.

such as that seen in the turned shaft in Figure 4.34C, the base feature is established with its axis of rotation at the intersection of two datum planes. Again, this strategy makes it much easier to maintain a constant location for the axis as dimensions on the shaft are modified.

Another decision that usually has to be made is how geometric features are distributed across part features of the model. For example, the part seen in Figure 4.34A and B could be created with a single profile (Figure 4.35A) or it could be divided into a series of feature operations

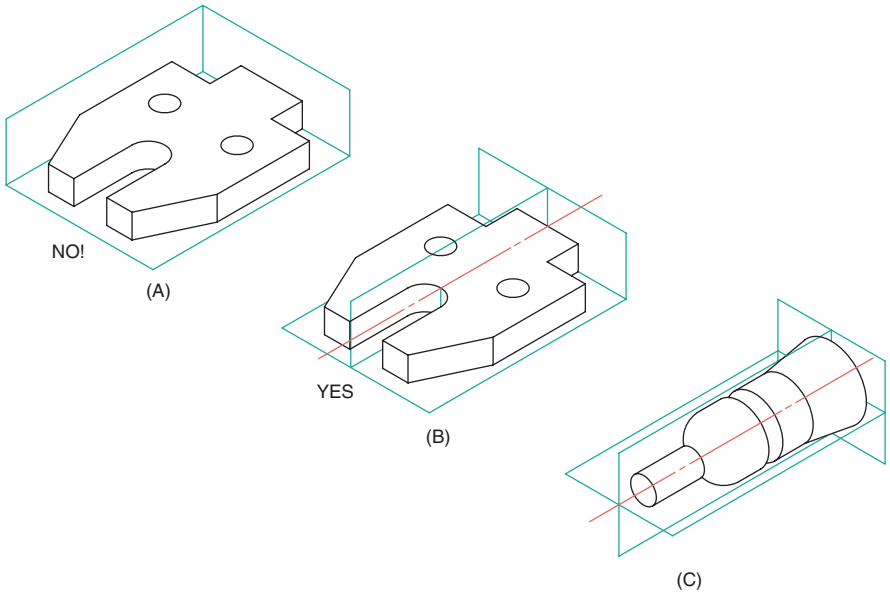


Figure 4.34 **Using symmetry in feature definition**
Construction planes can be used to help define symmetric features.

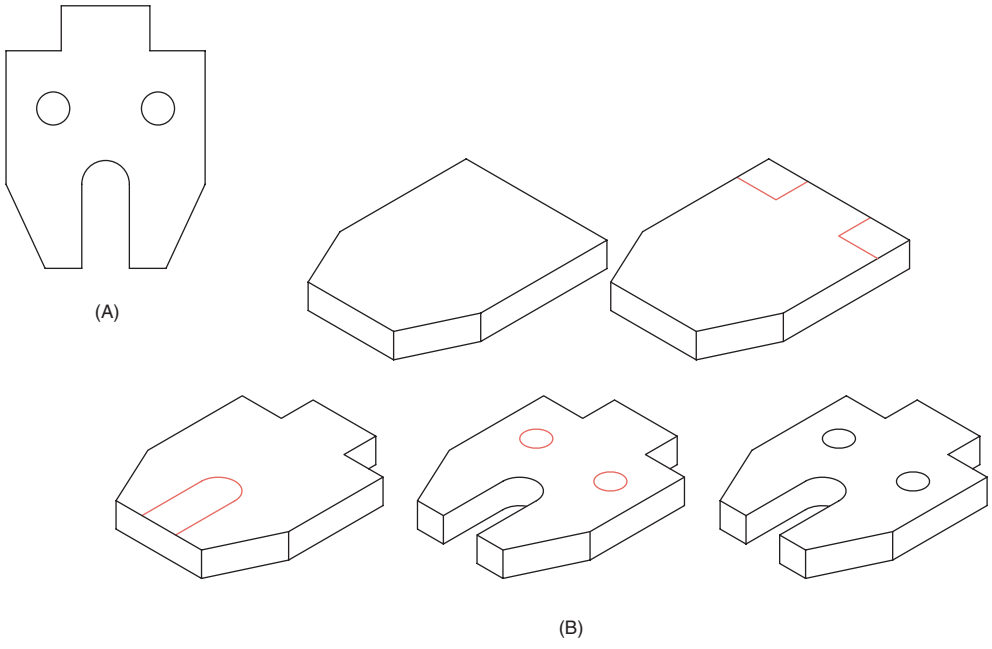


Figure 4.35 **Geometric decomposition for features**
How geometry is decomposed into features depends on an overall strategy for model use.

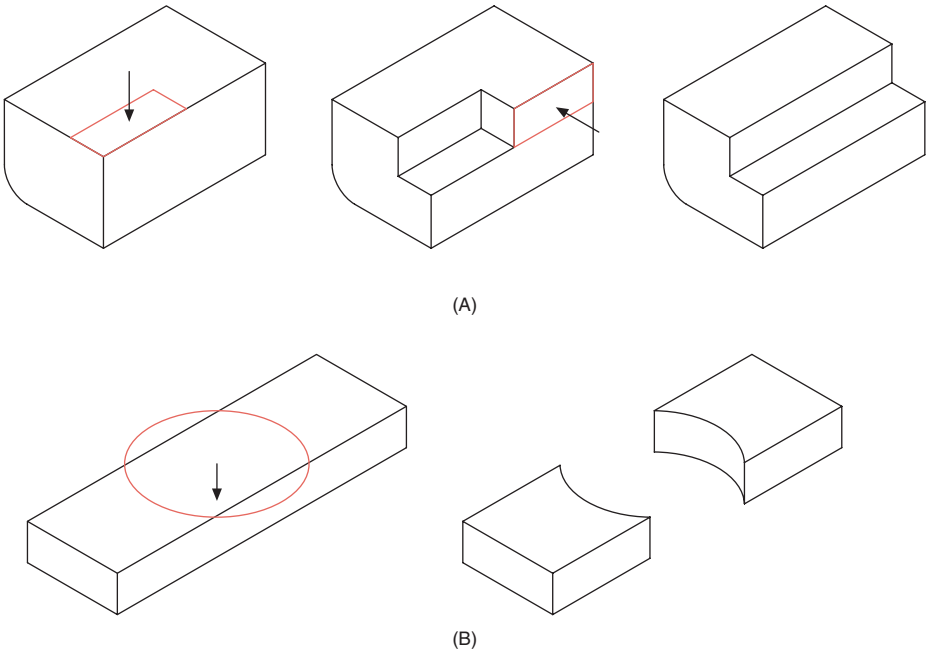


Figure 4.36 **Examples of poor feature definition**

(Figure 4.35B). What would be the advantages or disadvantages of each approach? Generally speaking, the more complex the geometry is in a feature profile sketch, the harder it is to apply all the desired geometric and dimensional constraints. Also, geometry, which may not exist in all design variations of the part, should be broken out as separate features. This way, the features can be either suppressed or permanently deleted from the model with a minimum amount of disturbance to other feature profiles. On the other hand, there is no sense in needlessly decomposing geometry into the simplest possible profiles and thus creating an unnecessarily large number of features in the part. Large numbers of overly simplistic features can also make management of the model difficult. Ultimately, the level of complexity of feature profile geometry comes down to what is a *logical* decomposition of the part geometry. This logic is driven by how features are defined in the design and manufacturing process.

One way to logically decompose the geometry of a part is to divide features into *primary* and *detail* features. Primary features would define the overall functional form of the part, while detail features would create geometry necessary for a particular manufacturing process, functional role, fastener attachment, or tactile/visual qualities of the part. Whereas primary features might define the mass of a part within 10 percent of its final form, detail features might add fillets and rounds, through holes or mount points for assembly, or knurling on a surface. Generally speaking, you would want to create the primary features first with their constraints defined based on the larger functional design characteristics of the part. Detail features are created later.

If detail features are tied to each other, then this should be done in a way that represents logical design or manufacturing groupings. For example, all of the fillets and rounds may be grouped together logically in the feature tree so that they easily can be suppressed as a group. The ability to turn detail features "on and off" is important for some end uses of the model. For example, noncritical detail features may add considerable time to finite element analysis without adding noticeable improvement in accuracy. Similarly, unnecessary detail may bog down the refresh rates of large assemblies on the computer screen without adding appreciably to one's understanding of it.

Finally, good modeling practice calls for the user to avoid certain types of feature operations in order to preserve the integrity of the model geometry and to allow for easier management of the model. To begin with, a single feature should be created by a single feature operation, if at all possible. For example, the notch in Figure 4.36A could have easily been created with a single operation. Now two model features have to be manipulated to make changes in

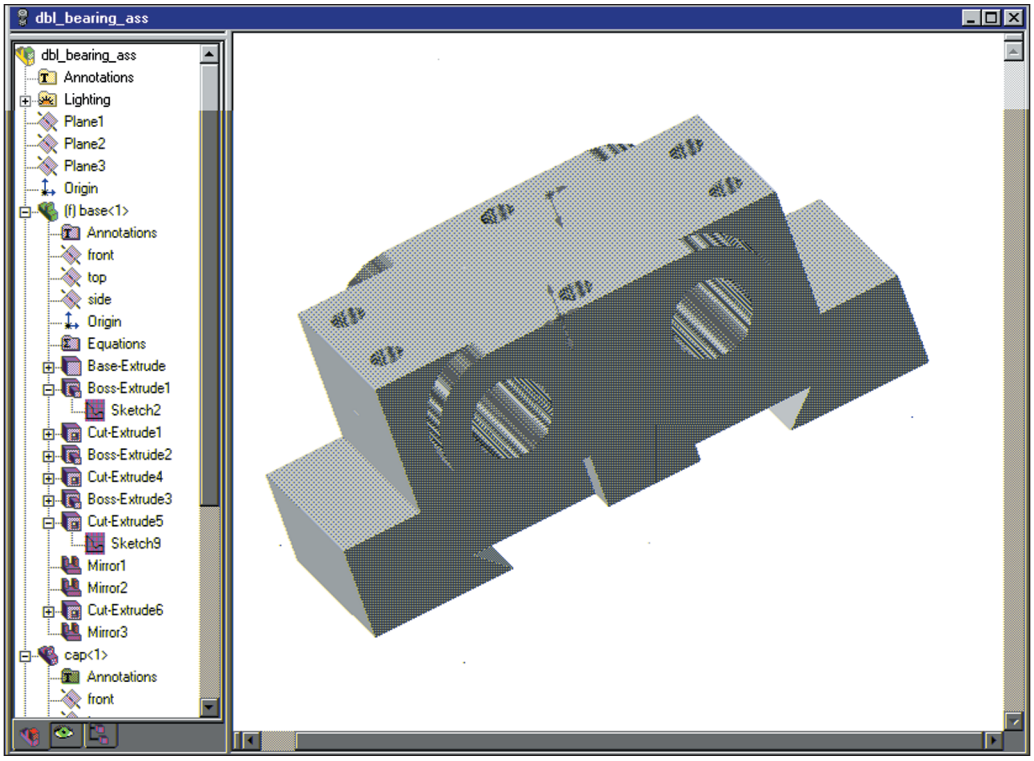


Figure 4.37 **Example of a part feature tree**

the logical geometric feature. At the other end, don't use a feature operation to create two logical parts when, in fact, the modeler still considers it one part model. Figure 4.36B shows an operation allowed by most modelers. This creates what looks to be two parts, but is in fact still one part model.

4.8 Editing Part Features

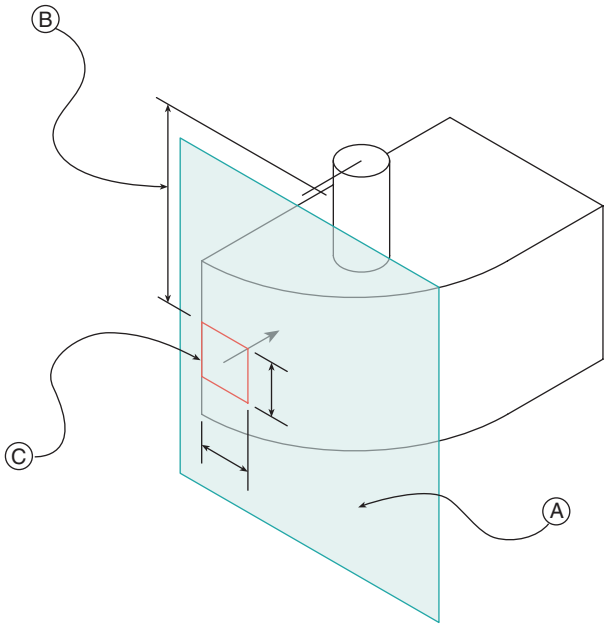
Once one or more features have been created on the model, the designer can go back and modify parameters used to define the features. The most common parameter to redefine is dimensional constraints. The dynamic nature of a constraint-based modeler makes it easy to modify the size, shape, and location of feature geometry on-the-fly. Constraint-based modelers were designed with the understanding that designing parts is a continual process of modification.

The ability to successfully redefine a feature will depend, in part, on how sound the planning was that guided the development of the initial features. Sections 4.8.1 and 4.8.2 will touch on a couple of elements of strategic planning for future editing of the model.

4.8.1 Understanding Feature Order

Most constraint-based modelers record the features created for a part in a *tree*. This tree may or may not be directly visible to the operator of the system. Many modelers have a window that depicts the features in a part model (Figure 4.37). Features, as they are created, are placed at the bottom of the **feature tree**. If a new feature is created as a copy or instance of another feature in the part model, the new feature on the tree may reference the original feature. Because features can be moved to other locations up and down the feature tree, the tree cannot be considered an exact *history* of feature creation. With many modelers, however, the order in the tree is the order in which features are applied to the construction of the model. Each time a new feature is added to the model, the user explicitly rebuilds/regenerates the model, or the modeler is otherwise triggered to do a rebuild, the feature tree is traversed from top to bottom; building the part model through a succession of feature operations.

Closely related to the idea of feature ordering is the concept of **parent-child** relationships between features. As in a real parent-child relationship, the child feature is



Parents of the new feature:
A) The sketch plane
B) Location dimensional constraints to existing feature
C) Overlap constraint with existing feature

Figure 4.38 **Feature interdependencies**

Parent-child relationships are established when new features reference existing geometry.

dependent on the existence of its parent feature. How is this dependency established? Every feature consists of constraints that establish both shape and size and also locates it. Though all of the shape and size constraints may reference other elements of the feature profile internally at least some of the location constraints must reference external features in order to locate it in 3-D space (Figure 4.38). This external referencing begins with the selection of a sketching plane. Whatever construction plane or model surface that is chosen as a sketch plane is considered a parent of the new feature. Whenever a profile sketch is located on the sketch plane by pulling a dimensional constraint from an edge on an existing feature, this feature becomes a parent of the new feature. Similarly, if an element in a feature profile is constrained through overlap with an existing feature edge, that feature also becomes a parent of the new feature. It follows that parent features must exist before (above) the child feature in the feature tree since the parent features are needed to define the new feature.

When creating a model, you must always be aware of feature dependencies, both when you create the model and when you edit it. Deleting a parent feature means that you must either delete the child feature that depends on it or redefine the child feature so that it no longer depends on the feature to be deleted. Changing the geometry of a parent feature also may alter the geometry or location of the child feature. For example, if the top shaft in Figure 4.38 were lengthened, then the child feature would have to move up in order to keep the value of the dimensional constraint between them constant. Besides deleting a feature, changing the topology of a parent feature may also invalidate the child feature. The dependency with the parent feature is typically not on the parent feature as a whole but on specific geometry in the parent feature. If, in Figure 4.38, the edge of the parent feature that the child profile overlaps were to be rounded, then there would no longer be an edge to overlap with.

When planning the construction of your model, there are a number of items to consider to make sure that feature dependencies are used to your advantage. In general, create dependencies with existing features as early in (far up) the feature tree as is logical. Linking locational dimensional constraints or overlap constraints with the initial three datums or the base feature will mean that the deletion or modification of an intermediate feature is unlikely to disturb the new feature. A corollary to this is to create all the features which are likely to become parent features as early as possible.

In more complex models it may be neither possible nor wise to place all dependencies high up on the tree. Instead, dependencies may be linked based on the logical grouping of design or manufacturing features. For example, an injection-molded part may require the creation of a geometrically complex fastening feature on the surface of the part. Rather than trying to tie all of the features of this fastener directly back to early features in the tree, a new datum plane and datum axis can be created from the original datums. From these new datums, a new *local* base feature is created from which other subfeatures of the fastener are made. With this setup, moving the new datums will move all of the features related to the fastener. Similarly, suppressing or deleting the new datums also will suppress/delete the fastener features without disturbing any other features.

Editing the order of features means moving features up or down in the feature tree. Dependencies between features means that features can't be moved to every possible position on the feature tree. Child features, for example, cannot be moved above any of their parent features on the tree. The fewer parent features a child has, the more flexibility there is likely to be in moving the feature.

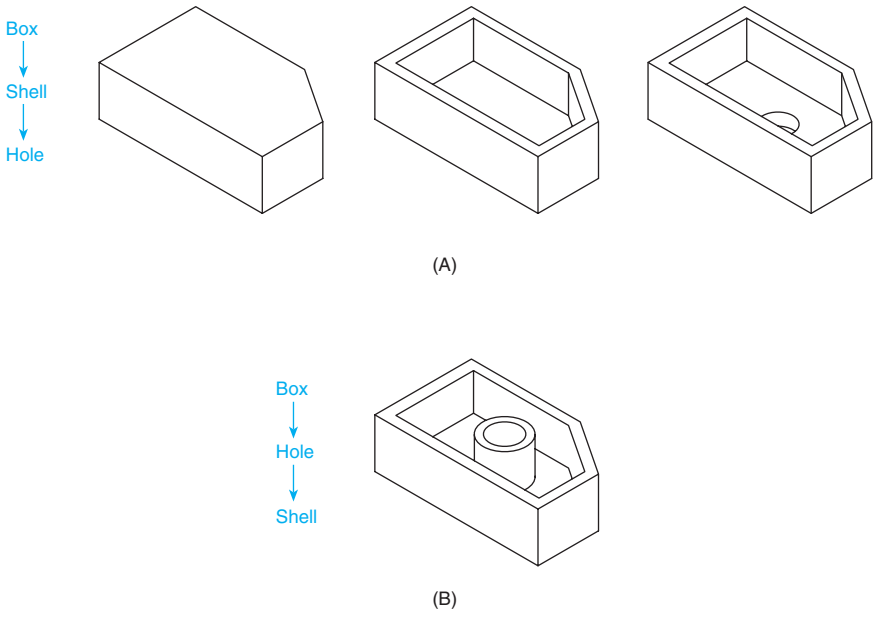


Figure 4.39 **Feature ordering affects final geometry**

The order in which features reside in the feature tree can affect the final part geometry.

Alternately, a feature can be redefined to change the parent feature, providing new possibilities for reordering features. Why would you want to reorder features? To begin with, you may be trying to more logically group features within the tree or you may be reordering as the result of deleting another feature. Reordering features also gives the operator a powerful tool for redefining the end resulting geometry of the model. Figure 4.39 shows a part with three model geometry features: a box, a shell, and a hole. When the features are created in this order, the sequence may look like that seen in Figure 4.39A. What happens when the hole operation happens before the shell operation? The end result is a very different model (Figure 4.39B).

4.8.2 Editing Feature Properties

In addition to changing the order of features within the feature tree, many of the parameters that initially defined the feature can be edited at a later time. If an error is made in defining a feature, it often is quicker to correct a feature parameter than it is to start the feature over from scratch. Since the sketch profile of a feature is considered a child of the plane on which it is defined, movement of the sketch plane—whether it is a construction plane or a face of another feature—also will move the sketch profile with it. Similarly, you also may be able to assign the sketch profile to another plane, creating an alternate parent-child relationship.

Within the sketch profile, elements of the profile can be deleted or modified. Typically, constraints associated with those deleted elements also will disappear. Alternately, all of the elements can be left alone, but the constraints associated with them altered. Dimensional constraints are usually the easiest to delete and create, but geometric constraints also can be modified. In some cases a constraint simply might be *suppressed,* allowing a new explicit constraint to take its place. Probably the most common modification of a feature is changing the values associated with dimensional constraints.

Other parameters besides the sketch profile also can be altered. The parameters that might be modifiable are:

- The type of sweep path
- The distance of the sweep
- Whether the sweep is one- or two-sided
- The direction of a one-sided sweep
- The side of the profile a removal operates on

Often the type of operation—removal or addition—cannot be changed. In addition, features that are automated will

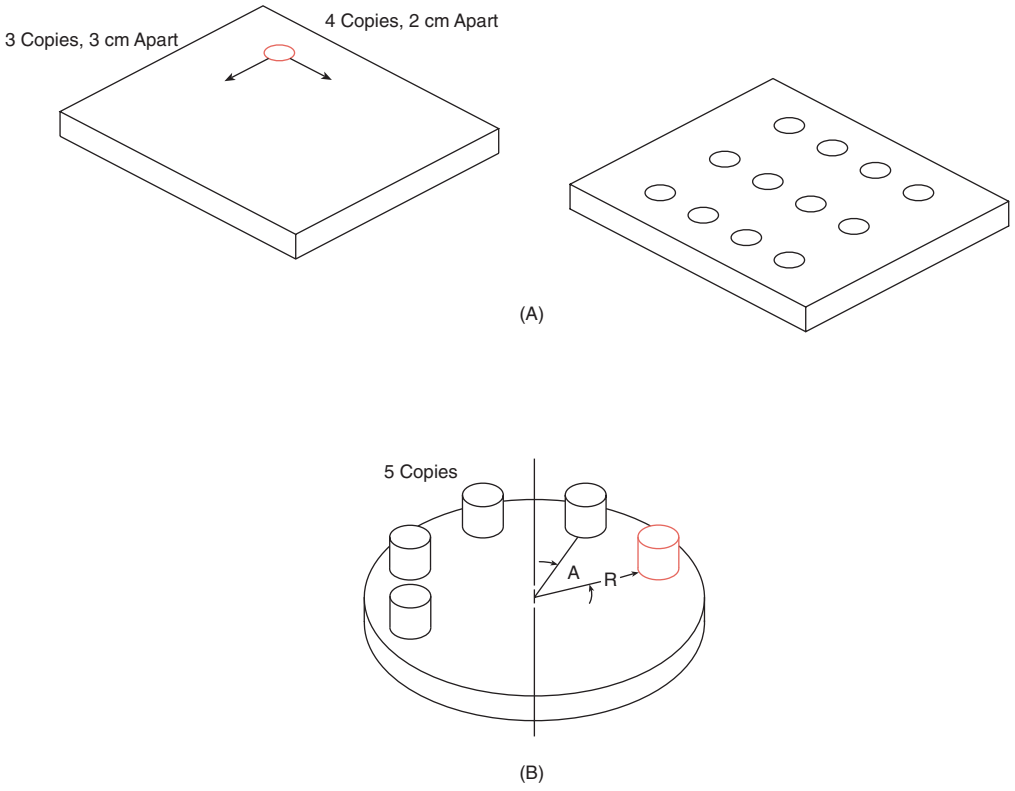


Figure 4.40 **Linear and radial arrays**
Array copy counts usually include the original plus the number of copies.

have more limited options. For example, you probably can't change the circular profile in a hole feature to a square.

4.9 Duplicating Part Features

The ability to duplicate geometric elements is a powerful attribute of all CAD programs. Constraint-based modelers typically allow the user to duplicate at the level of features. Often, the level of dependency between the original and newly copied feature can vary. Typically, the topology of the profile will be the same, as will the primary feature parameters such as direction and distance of the sweep and whether it is an addition or removal operation. Whether all of the dimensional constraints are tied between the parent and the child copy often is determined by the options chosen. For example, it may be that the size of a copied hole is independent of its parent hole. Locational constraints often are modified as part of the copying process. For a general copy, any of the following might be set independent of the parent feature:

- The value of locational constraints
- The model geometry to which locational constraints are attached
- The plane on which the feature profile resides

Often the copying process, especially if the copying involves the creation of more than one child, is automated to some degree. A common tool is an *array* option. With a **linear array** the parent feature is copied in one or two dimensions with specifications given for distances between copies and the total number of copies (Figure 4.40A). Alternately, a total number of copies might be specified along with a distance within which the copies are to be evenly distributed. With a **radial array**, an axis of revolution is specified along with a radius, angular displacement, and total number of copies (Figure 4.40B). Another common copying process is a **mirror**. In this case, a mirror plane is specified along with features to be copied/

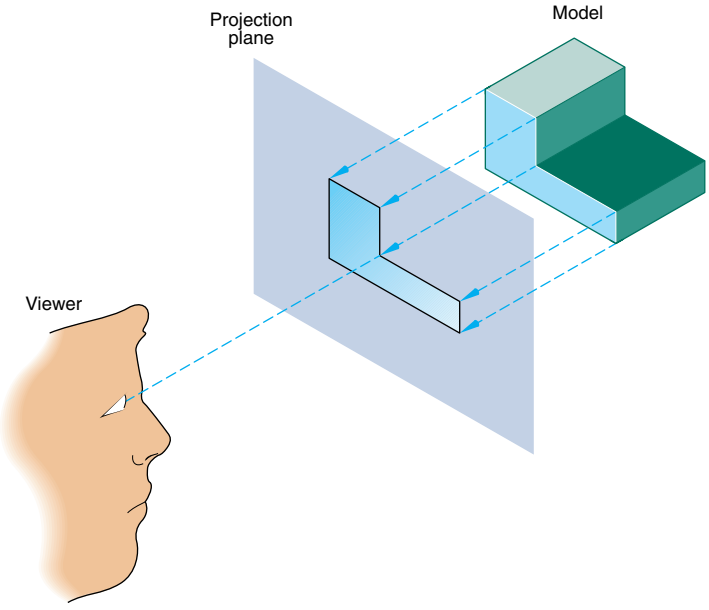


Figure 4.41 **Elements of a projection system**

mirrored. Often with a mirror copy, most of the constraints cannot be set independent of the parent feature, since the design intent is to keep the child a mirror image of its parent. Moving the mirror plane, however, will alter the location of the child parts.

4.10 Viewing the Part Model

The techniques used for viewing 3-D models are based on the principles of projection theory described in Chapter 5. The computer screen, like a sheet of paper, is two-dimensional. Therefore, 3-D forms must be projected into 2-D. For review, the primary elements in creating a projection are the *model* (object), the *viewer,* and a *projection (view) plane* (Figure 4.41). A coordinate system is attached to each of these elements and is used to define the spatial relationship between the elements. The world and any associated local coordinate systems define the model. The viewing coordinate system also has three axes, which are defined by the viewer's orientation in space: vertical, horizontal, and depth. Even though it would be convenient to associate the computer screen with the image plane coordinate system, that could lead to incorrect assumptions. Therefore, it is best at this time to imagine a 2-D plane in 3-D space, similar to a workplane.

4.10.1 View Camera Operation

The **view camera** is a metaphor used to describe the viewing process with respect to 3-D models in various CAD systems. Most systems support more than one view of the model at a time. For each view, there is a camera, and there is an image plane onto which the model is projected (Figure 4.42 on the next page). The camera records the image on the plane and broadcasts that image to the computer screen. The broadcasted image is contained within a **viewport** on the screen, and viewports may be resizable and relocatable or fixed, depending on the system.

In nearly all cases, the image plane is oriented such that the viewing direction is perpendicular to the image plane, creating an orthographic projection. For the most part, oblique projections are not allowable with 3-D modeling systems. Most systems also default to setting the view camera infinitely far away from the model, creating a parallel projection. Changing a parallel projection to a perspective projection is usually a matter of setting the view camera distance to something other than infinite. The closer the camera is to the model, the wider the view angle required (Figure 4.43 on page 165). Some systems allow the view angle to be manipulated, while others change the viewing distance. In either case, the effect is a change in the convergence of parallel edges on the model, from nearly parallel to extremely convergent.

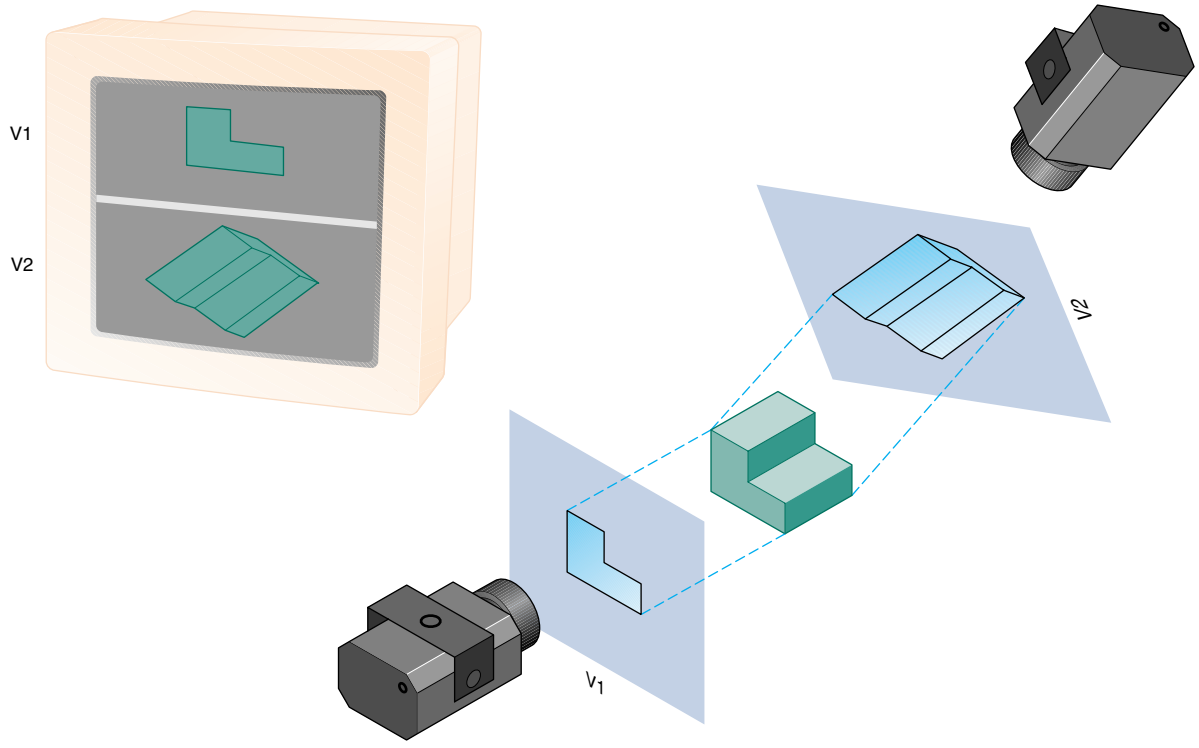


Figure 4.42 **The view camera**
The view camera captures a projection of the model on the image plane.

View camera orientation can be established in a number of ways. Often a construction plane or a face on the model is chosen as being normal to the line of sight. Besides depending on set views, modeling software allows for real-time, *dynamic rotation* of all but the largest models. Once a view camera has been oriented and a projection calculated, a number of auxiliary commands such as zoom and pan can be used to manipulate the view of the model (Figure 4.44 on page 166).

A related issue in view specification is how to display the geometry of the model. The most common methods, shown in Figure 4.45 (on page 166), are:

- Wireframe
- Hidden lines rendered
- Hidden lines removed
- Shaded

In addition, the lines representing tangency can be:

- Hidden
- Displayed solid
- Displayed as an alternate linetype

Exactly how the model is rendered can change numerous times during the construction of the model. Each rendering mode has its own advantages, with the decision of which mode to use often based on balancing the need to minimize the number of lines shown on the screen with having access to tangencies and hidden features. It is important to note that a model can be displayed in wireframe and still contain solid model information in the database. The rendering of the model is independent of the underlying geometric database.

4.10.2 View Camera Strategy

The orientation of any particular view camera with respect to the world coordinate system can vary considerably during the course of model building. This lack of fixed orientation often can be disconcerting. The fact that view cameras are moving all around the model while you are seated firmly in front of the computer screen also contributes to the disorientation. If the camera is rotated in one direction about a model, the object itself appears to rotate in the opposite direction on the screen. An important distinction

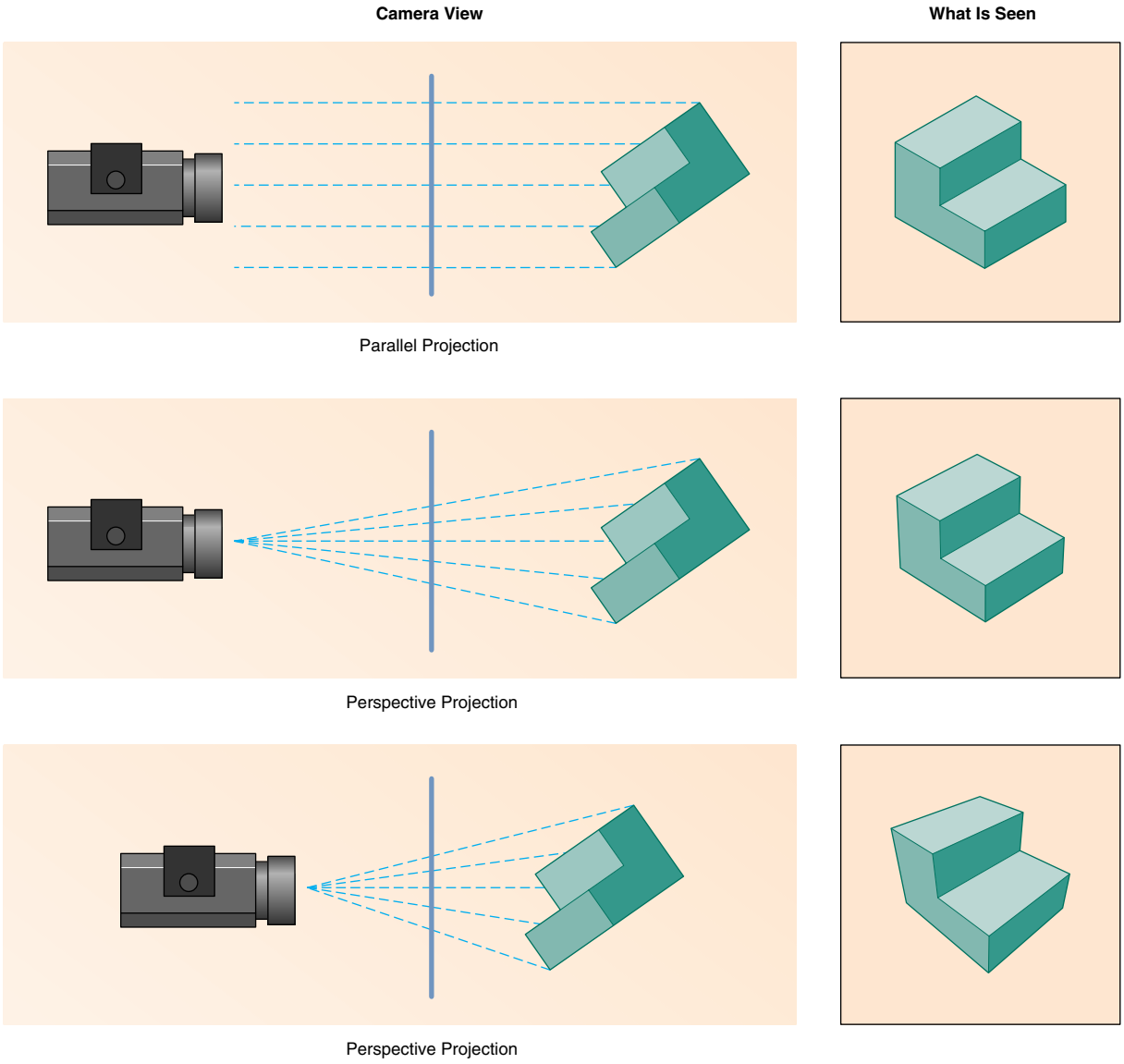


Figure 4.43 **Parallel versus perspective projection**

The closer the camera is to the object, the wider the angle of view and the more severe the convergence.

must be made between the two actions. If you rotate the model, its orientation to the world coordinate system changes, as will its geometry in the database. If you move the view camera, however, the camera has changed location relative to the world system, but the geometry of the model has remained untouched. Preserving the location and orientation of a model can be critical when multiple parts in an assembly are being coordinated. Getting a new view of a part most likely will involve rotating the camera and not the part.

Projection calculations are not sensitive to the point of view; a traditional front view of the object is as easy to calculate as an isometric pictorial view. This is not the case with 2-D CAD or mechanical drafting. The implication is that, except for the occasional need to conform to a standard view, there is no reason to stick solely to the

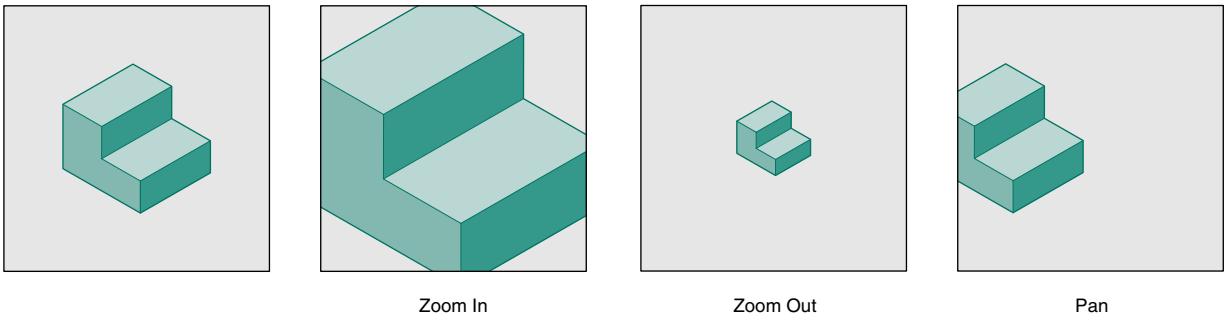


Figure 4.44 **View commands that do not involve changing the viewpoint**
Pan and zoom commands do not change the projection of the model.

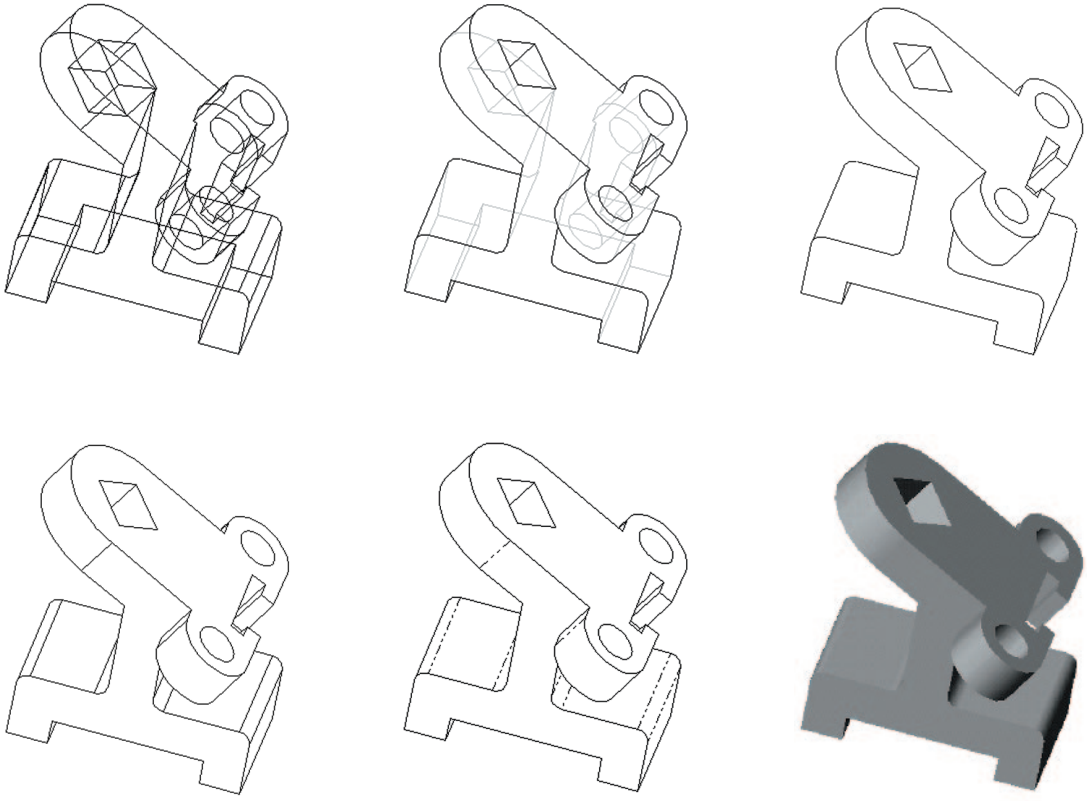


Figure 4.45 **Different options for rendering a model**
There are many options for how to depict hidden edges and tangents.

traditional orthographic or pictorial views when working with 3-D modelers.

Viewpoints should be chosen on the basis of the task to be performed. With multiple viewports, the views often are distributed between those used strictly for viewing and those used for constructing the model. A pictorial view of the model is useful for observing the progress of the overall construction. The pictorial is often an axonometric view oriented such that the features currently being worked on are seen with a minimum amount of foreshortening.

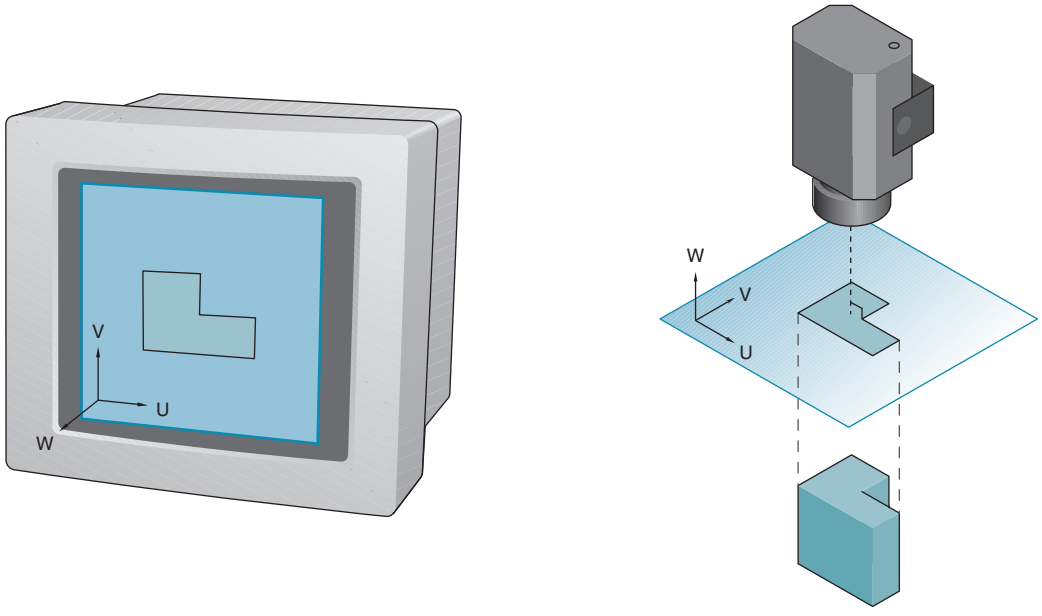


Figure 4.46 **Orienting the workplane coincident with the viewplane**
Workplanes often are used to orient the view camera.

Pictorial views are a compromise that allows all three major dimensions of the model to be seen. Rather than being limited to a certain pictorial view, the user can interactively orient the model to depict the features to the best advantage.

During model construction, traditional multiviews also are used. The workplane is aligned to an existing face on the model, or along a global axis, and the resulting view matches a traditional front, side, or top view (Figure 4.46).

To choose viewpoints for construction or for viewing a completed model, use the same rules as for sketching or drawing. They are as follows:

- Avoid views that are close but not quite a standard orthographic view (Figure 4.47A on the next page). Such views would have the features along one primary dimension severely foreshortened and therefore very distorted.
- Clearly identify the features of interest and orient the view camera to depict those features (Figure 4.47B). If there are important features along all three primary dimensions, an isometric or near-isometric view may be appropriate.
- If most of the features of interest are along only two of the three primary dimensions, choose a view that favors those two; however, retain the third dimension. If there are features on more than three sides of the model, another viewport with a view from the opposite side of the model may be required.
- For features that must be carefully analyzed, choose a view where the applicable faces can be seen in their true size and shape (Figure 4.47C).

4.11 Application of Part Model Data

The part model, and its associated features, is a primary module of product information that can be shared with other applications. These applications may be other modules of the 3-D modeling system, another modeling system, or other related technology used by the organization. As has been demonstrated in earlier sections, constraint-based modelers offer numerous powerful techniques for dynamically modifying the part model. An organization can leverage this powerful capability by linking, or associating, the dynamic part model to other product information and tools that depend on the part model so that they are always working with the most current part information. The part model should be thought of as part of a larger product database being managed and updated by the company. Dynamically linking the most up-to-date information means that engineers, technicians, marketing professionals, and management always have access to the most current information.

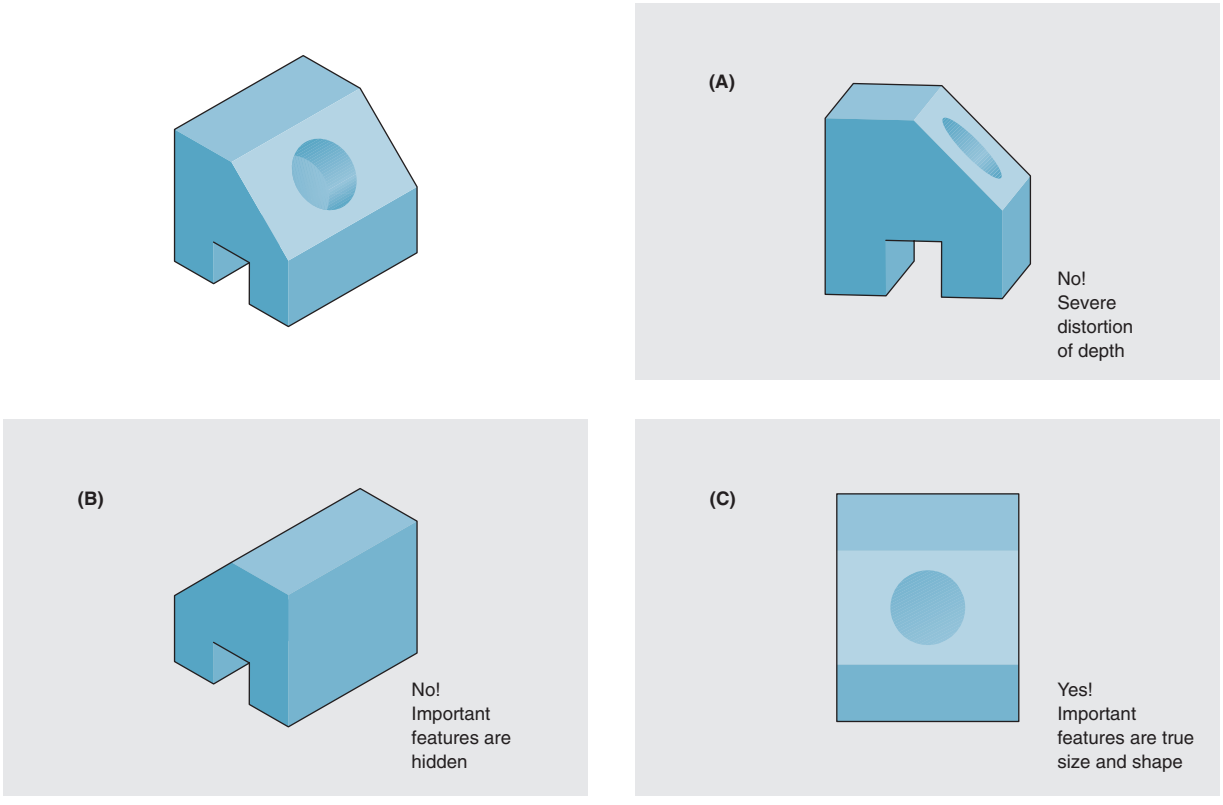


Figure 4.47 **Good and poor viewing practice for modeling**
The choice of view should be feature driven. Which view best depicts the features being operated on?

4.11.1 Model Data Associativity

In a concurrent engineering environment where many people are working on a design simultaneously, there is a great risk that not all team members have up-to-date information to perform their jobs. On one hand, they do not want to extract model information too soon if the model specifications still are changing, yet if the process of extracting model information is time consuming, they cannot leave this step for the last minute. Data associativity addresses this dilemma by creating a dynamic linkage of information between the 3-D model database and the supporting applications. Whenever the model is altered, the associated data in the supporting application also are updated automatically, giving all design team members the most current information to work with.

The dynamic linkage of data can be established in a number of different ways. The linkage can take place all within one software package which contains multiple discrete applications. Links also can be established between different applications running on a single workstation. With the assistance of networking and Product Data Management (PDM) tools (see Section 2.6), data can also be dynamically linked across networks to all users working on the same design project. Links made between the 3-D model and supporting applications can be either *unidirectional* or *bidirectional.* With **unidirectional associativity**, the supporting application's data can be altered by changing the 3-D model, but not vice versa. With **bidirectional associativity**, changes in either the 3-D model or data in the supporting application will affect the other. In addition to the direction of the data flow, the updating can be set to be done automatically whenever there is a change, or done manually when an update command is chosen.

It is important to emphasize that associativity means the model and the application that is linked to it must always be able to find each other. Management of model files and the files created by the associated applications

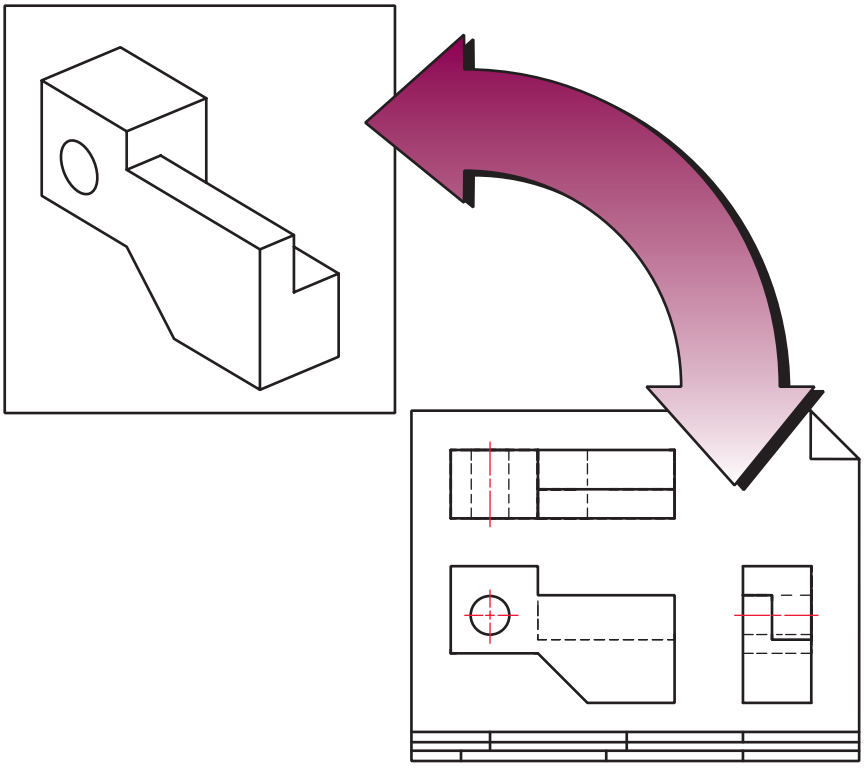


Figure 4.48 **Associativity between 3-D model and 2-D drawing**
With bidirectional associativity, changes in the 3-D model are reflected in the drawing and vice versa.

becomes a critical issue. The deletion or move of a part model file may mean that drawings of that part or an assembly that references the part may no longer function. In larger organizations with many people accessing thousands of files, PDM software, databases, and other file management tools are essential to preserving these linkages.

4.11.2 Documentation

One of the most common types of data associativity is between the 3-D model and a 2-D production drawing. For example, if the 3-D modeling system also has a 2-D drafting module, then links can be set up between the model and the views represented in the production drawing (Figure 4.48). As in 2-D CAD, the starting point for creating documentation in a 3-D modeling system is to establish paper size and to apply both a standard or custom title block and border. At this point, rather than drawing projections of the model from scratch, projections are extracted directly from the 3-D model. The views in the production drawing can be thought of as live projections of the 3-D model from different viewpoints.

Typically, a *base view* of the model (such as a front view) is anchored in the production drawing. Then, principal and auxiliary views are established relative to this base view. These views can be thought of as children of the base view, since changes in scale and location of the base view will alter these additional views. The scale established for the base view typically will hold for child views created from the base view. However, special detail or removed views can be created at alternate scales. These views typically are not tied as closely to the base views. What portion of a view is shown can be controlled through the use of section, partial, and broken views. What are more difficult to add, however, are those standard conventions drafters often use that violate true projections of the object. For example, aligned sections or the revolving of spokes or ribs are not typically allowed. Hidden lines may be turned off all together or selectively.

Once views are placed, notation can be added to the views. Since dimensional constraints should represent the design intent of the model, displaying these dimensions in the drawing should suffice for the majority of the dimensions shown in the documentation. Modelers who know that documentation will be required can try to constrain the model in ways that also will meet documentation standards. For example, if coordinate dimensioning is required for the documentation, this may influence how the model is constrained. Similarly, if geometric dimensioning and tolerancing (GD&T) notation will be required, careful selection of construction plane and dimensional constraint placement means that these elements can be reused as part of the GD&T notation (see Chapter 12). Ideally, modeling strategies, documentation standards, design intent, and manufacturing processes should all be aligned closely with each other, requiring a minimum of reworking of dimensions for the final documentation. Other text and symbol tools can be used to apply any additional notation needed on the documentation. More information on proper notation of documentation can be found in Chapters 9 and 10.

It is important to note that in documentation with bidirectional associativity with the model, dimensions in the document that represent constraints can be modified to drive changes in model geometry. This becomes a powerful alternative method for modifying models and exploring alternative designs. With all necessary dimensions displayed in well-defined multiview and pictorial views, this becomes a useful alternative means of exploring the impact of dimensional changes in the model. Note that all dimensions associated with the document views should update when the model changes. However, dimensions added to the views, which were not originally constraints in the model (often referred to as reference dimensions), cannot be used to *drive* the model. As long as they are attached to geometry on the model, they should update as the model is altered.

4.11.3 Assembly Modeling

The assembly of parts into larger models uses many of the same techniques and concepts used in part modeling. In an assembly model, **components** are brought together to define a larger, more complex product representation. A component is either a part or another assembly brought into an assembly model and associated with other components (Figure 4.49). Assemblies, when brought in as components, are now considered subassemblies in the new larger assembly. These subassemblies, in turn, are made up of components themselves. Any assembly can be thought of as a hierarchy of subassemblies and/or parts and can be represented in a *tree structure* much like the features in a part. A part or subassembly can be brought in multiple times to an assembly, creating multiple *instances* of the component (Figure 4.50 on page 172). Instancing of components does not add appreciably to the size of the assembly model since all of the instances refer back to a single part model (or part models of a subassembly). The same part or subassembly also can be used across multiple assembly models. Common hardware, fasteners, and other parts used in multiple designs by a company can be kept in networked component repositories for use by engineers and designers all over the company (Figure 4.51 on page 172). Care is needed to manage this repository since change in a part here may affect multiple assemblies referencing it.

Constructing an assembly begins with bringing in a *base component.* As with the construction of a base feature in a part, a base component usually will be selected because of its central role in defining the overall assembly. Each successive component brought in needs to be oriented and located relative to other components in the assembly. Location and orientation is achieved by defining geometric relations between geometric elements of a component in the assembly and the elements of components being brought in. These elements may be part model geometry or construction geometry associated with the component. Directionality of the geometric elements is often an issue in orienting the new component. A face on a part model will have an *outside* and *inside,* often with the positive direction defined with a vector on the outside surface pointing away from the model. Construction planes do not have a natural inside and outside, so the directional vector usually has to be defined on the fly. Edges on the part models and construction axes may or may not have directionality to them. Coordinate systems, both global and local, can also be used to orient and locate components.

Defining these geometric relations primarily is done with two basic tools (Figure 4.52 on page 173):

- **Mate.** Two part surfaces/construction planes are set coplanar with the directional vectors opposing each other.
- **Align.** Two part surfaces/construction planes are set coplanar with the directional vectors pointing the same direction. Alternately, two edges/construction axes are set collinear.

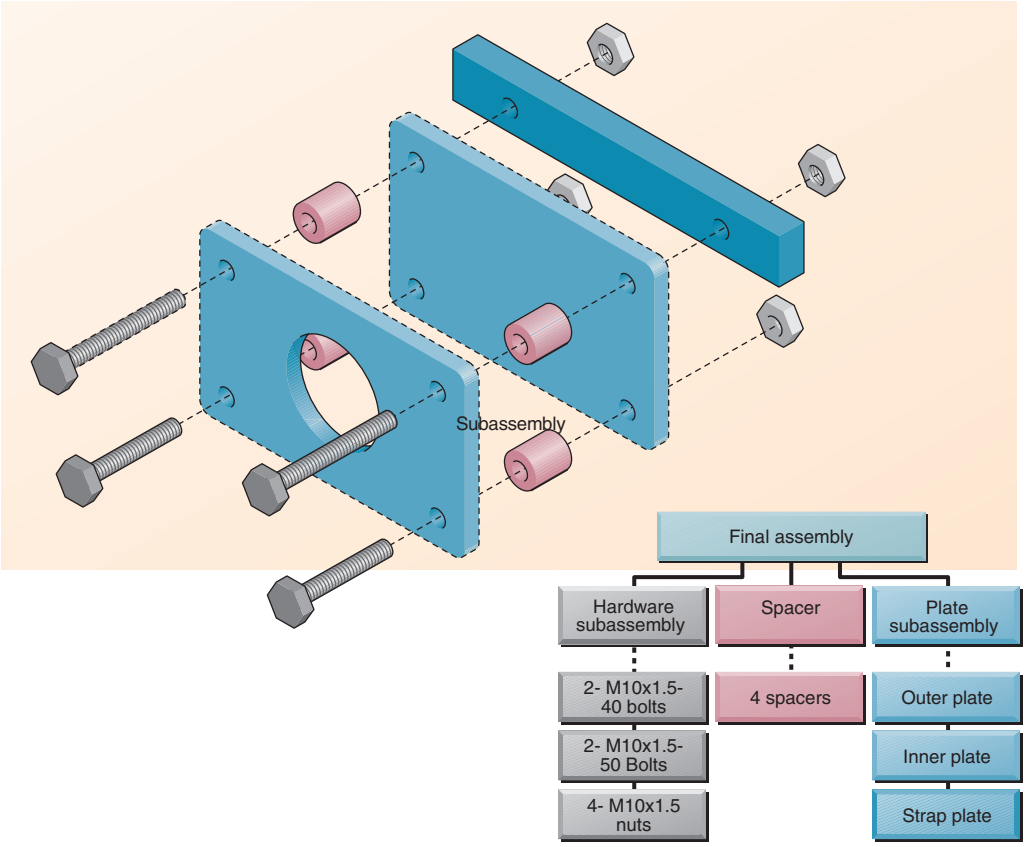


Figure 4.49 **Hierarchical parts structure of an assembly**

An assembly usually consists of a hierarchy of parts, some of them brought in as multiple instances.

A modifier for both mate and align is *offset,* where an offset distance for surfaces is defined. The surfaces continue to be parallel to each other. In addition to mate and align, there may also be tools to define:

- Parallelism (without specifying distance)
- Tangency
- Perpendicularity
- Surface intersecting an edge/axis
- Edge/axis intersecting a point/vertex
- Angles of surfaces/planes to each other
- Relationship of geometry to a coordinate system

The assembly modeler also may allow the creation of construction geometry or coordinate systems on the fly as components within the assembly to help with the construction process. Establishment of these geometric relations between components creates parent-child relationships between the existing components and the new components coming in. Operators want to heed the same strategic principles in establishing parent-child relationships between components in an assembly as they did with features in a part model.

An assembly modeler tracks the relationship between components through *degrees of freedom.* The establishment of each geometric relationship between two components reduces the degrees of freedom components have to move relative to one another. Degrees of freedom are either rotational or linear, with a fully free 3-D part having six degrees of freedom: three rotational and three linear. When component parts have zero degrees of freedom relative to each other, they are considered *fixed.* Figure 4.53 (on page 173) shows the process of restricting the degrees of freedom of two component parts. Depending

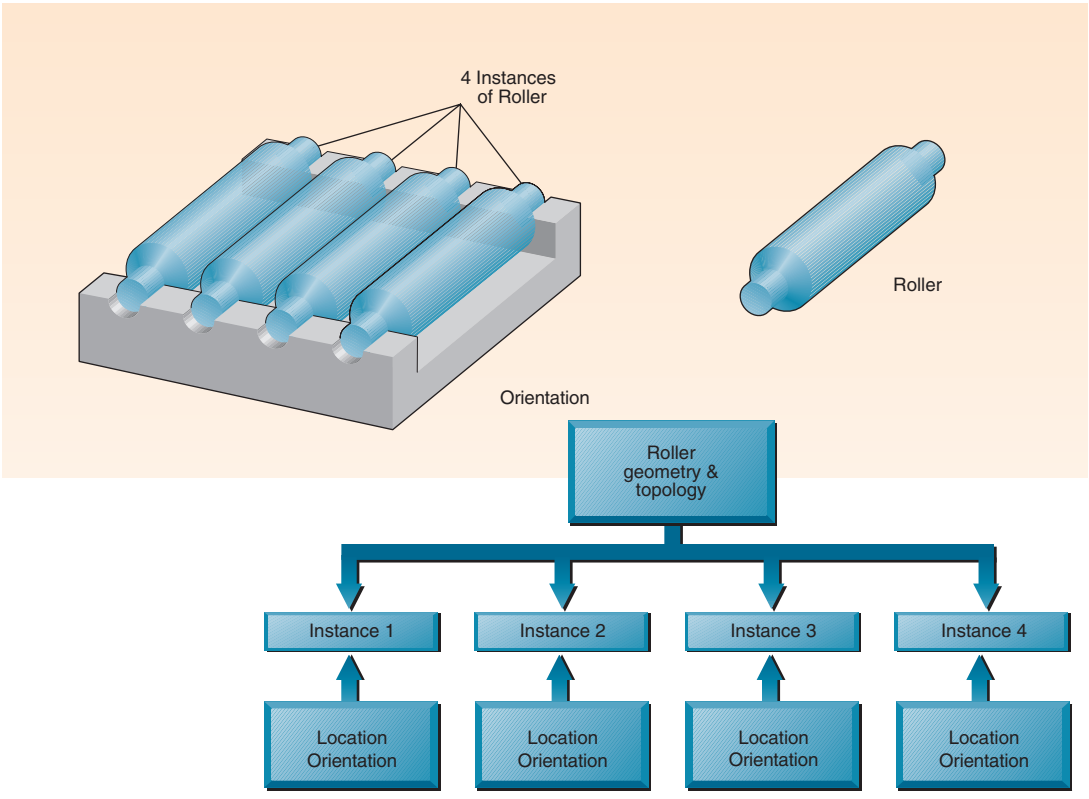


Figure 4.50 **Creating multiple instances of a roller**

Part components can be instanced multiple times in an assembly to help with management.

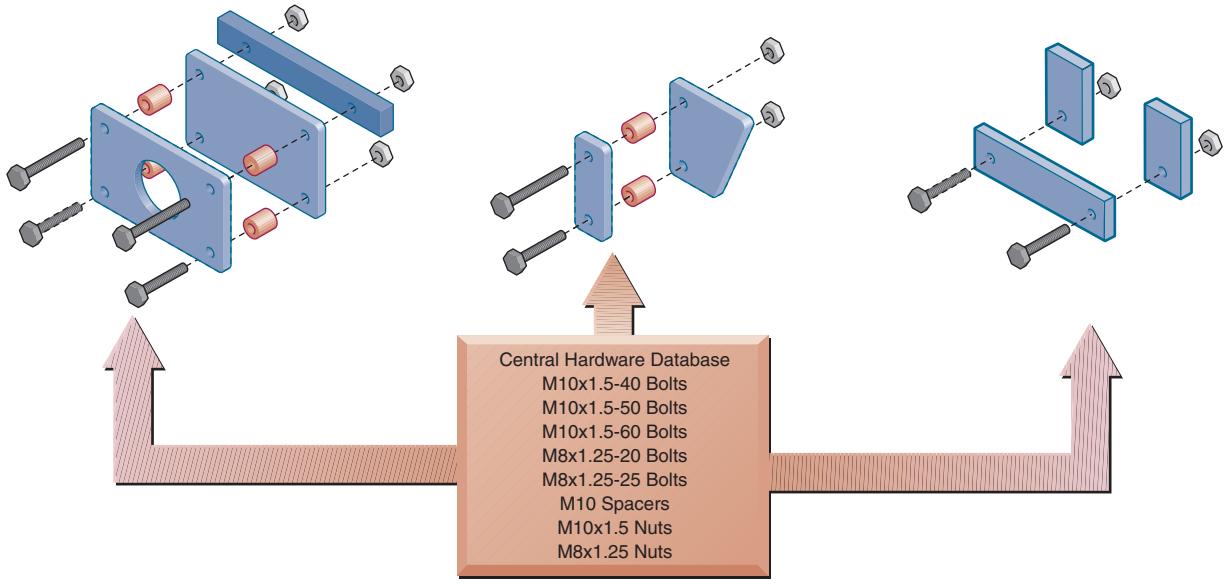


Figure 4.51 **Shared common parts**

Standard components can be instanced and shared across multiple assemblies.

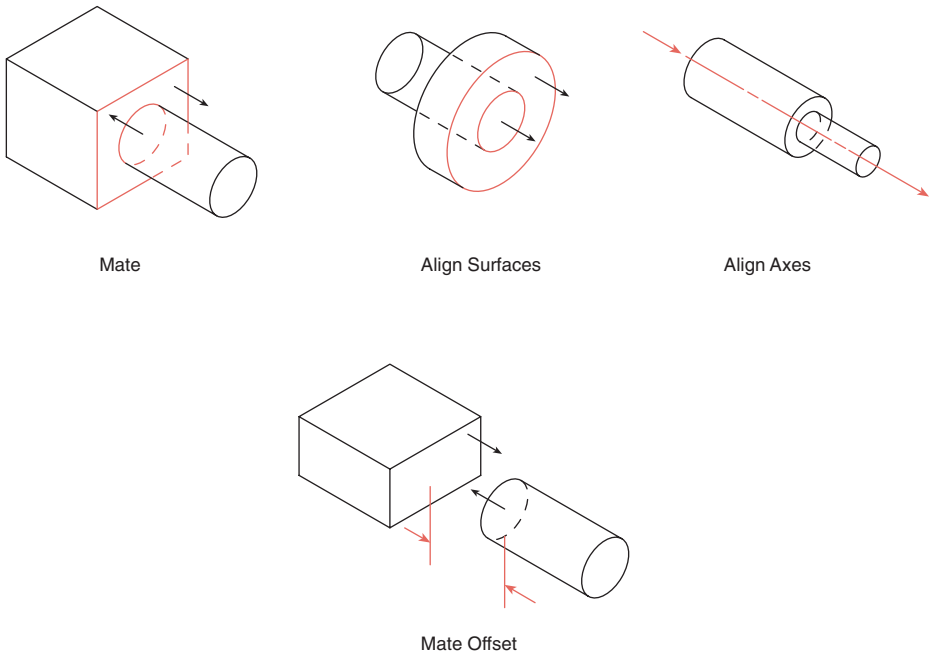


Figure 4.52 **Methods for joining parts in an assembly**

Mating and aligning are the most common methods for relating parts to each other in an assembly.

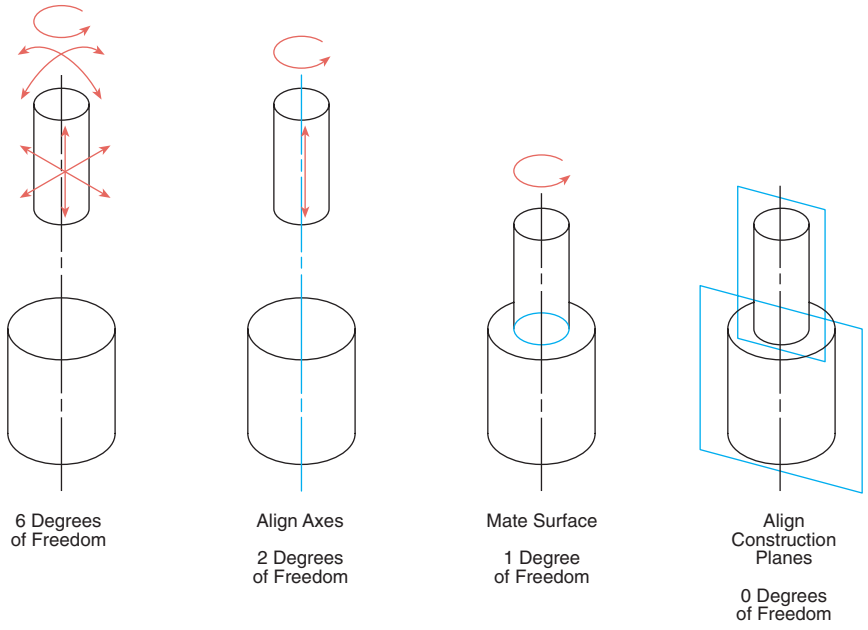


Figure 4.53 **Degrees of freedom between components in an assembly**

on the modeler, component parts may or may not be allowed to be left with degrees of freedom. If the assembly model is going to be used in kinematic or dynamic analysis, then degrees of freedom representing how the component parts are actually going to move in the assembly will need to be represented in the model.

Just as there often are limited ways in which part models can be modified within drawings, part models often can be modified within assembly models too. If the assembly modeler has bidirectional associativity with the part modeler, then dimensional constraints on parts can be modified in the assembly with the results reflected in both the assembly model and the individual part model. Simply typing in a new number can make modifications to constraint values, or they can be driven by other constraint values through parametric equations. Just as equations can be used to link constraints across features in a part, equations also can be used to link constraints across part models in an assembly. These *assembly-level equations* must reference both a constraint parameter and the part that the constraint is associated with. This technique can be an extremely powerful tool to make sure that interacting features across parts—such as pins, holes, notches, etc.—continue to stay aligned and the proper size.

A particularly useful tool found in many assembly modelers is the ability to remove material from a component part, not with a feature operation, but with geometry from another component. For example, a Boolean subtraction operation can be performed between two parts in an assembly with the resultant material removal being represented as a new feature on the part. This can be a valuable technique for modifying a part to conform to a particularly complex fit in an assembly. Note that because of the dynamic associativity between part and assembly, the modified part now will be dependent on both the other part and the assembly model to define the new feature.

Modification of dimensional constraints and performing geometry removal within the assembly modeler are both examples of **bottom-up design**, where the final geometry of the parts has not been defined before bringing them in as components within the assembly. Often, assemblies are the best place to evaluate design goals for a product, so the final geometry of a part may not be decided until it is fit with other parts in their near-final configuration. The reverse of this approach would be **top-down design**, where all of the part geometry is defined before it is brought into an assembly. Though it may be possible to model parts from scratch within the assembly modeler, most part design uses a combination of both top-down and bottom-up design. Basic geometry for a part is established first; then it is brought into an assembly where it can be further refined, as necessary.

Just as individual parts can be documented, assemblies can also be brought into the document module of a modeler. The same techniques used to bring in a single piece are used for the assembly. As is the case with more traditional engineering drawing practices, what views are used and how they are notated is often different for an assembly than it is for individual parts. One additional tool that is very useful when documenting assemblies is the ability to create an **exploded view**. Often a default exploded view can be created automatically by having the model components move away from each other along the lines of the geometric constraints applied in the assembly. The location and orientation of the parts then can be adjusted to create a more optimal view. Flow lines then can be added between the part components. In addition to exploded views, tools to create bills of materials and to attach part codes with balloons and leaders are also standard in most document models. More information on proper notation of documentation can be found in Chapter 10.

4.11.4 Analysis

The specification phase of the design process establishes the requirements for the needed product. Periodically, the various design concepts being developed should be evaluated against these requirements. As the design process progresses, changes become more expensive to implement, and fewer design options can be explored affordably. However, evaluations take time and resources, and they should not be done unnecessarily. Selecting the right analysis method is as important as determining when or how often evaluations must or should be done.

The speed of the analysis depends, in part, on the scope of the analysis and the type of analysis performed. Decisions need to be made as to what parts in an assembly and what regions of a part need to be analyzed. A careful evaluation has to be made as to what the interactions are between features in a part and parts in an assembly in order to know what to evaluate. Having unidirectional and bidirectional associativity between the part/assembly model and the analysis tool can speed the analysis process considerably. Having geometry changes in the model directly reflected in the analysis model, without the increased time and error probability of translation, encourages continual, iterative evaluation over the course of the design refinement.

Figure 4.54 **Ray traced rendering**
(Courtesy of Blue Moon Studio, Inc./Shade.)

Figure 4.55 **Physical models made directly from a 3-D CAD database**
(Courtesy of StrataSYS.)

Visual **Visual inspection** is an evaluation technique that is quick and easy, although very subjective. The visual inspection may involve making sure all the necessary parts are in an assembly model. In addition, multiview analysis of parts and assemblies can provide an initial confirmation of feature size and placement. Technicians and engineers familiar with the end product can often make well-educated design decisions based purely on a visual analysis. Visual analysis also is used to make aesthetic decisions concerning the "look" of the model. Industrial designers and marketing professionals depend heavily on visual analysis to judge aesthetic appearance.

Rendering techniques that enhance the visual analysis process involve steps ranging from simply removing the edges and surfaces normally hidden from view (Figure 4.45 on page 166) to adding shading or color to make some surfaces or features stand out. More advanced rendering techniques, such as ray tracing, can accurately model the response of different materials to light rays (Figure 4.54). Such techniques assist not only in aesthetic design decisions but also in safety decisions where reflected light (glare) could pose problems.

Prototyping Even with the capability of developing virtual models, physical mockups often are needed as the design progresses. Prototyping techniques allow physical models to be made directly from a 3-D database, as in Figure 4.55. With some systems, the outer surface of the model is translated into a series of paths traced by cutter heads on a milling machine. This technique can be used for high-precision manufacturing of the final product, or for lower-precision production of prototypes made from inexpensive materials, such as wax or foam blocks. An increasingly common technique, called **rapid prototyping**, uses a number of technologies to "build" models to match the form and geometry of critical features of the virtual computer model. With most rapid prototyping techniques, the CAD model goes through a two-step conversion (Figure 4.56 on the next page). The first step is to *tessellate* the surface of the CAD model into an interconnected network of triangles. This tessellated model is written out to a data exchange file, usually in .STL format. The next step is to 'slice' the tessellated model into a series of thin slices. Determining the thickness of the slicing is a trade-off between the level of accuracy of the model and the time it takes to build the model. Each prototyping technology will also have its own upper and lower limits to slice thickness. Earlier generations of the technology only were able to achieve thicknesses in the range of 0.020 inch (0.5 millimeter), while common thicknesses today are in the range of 0.002 inch, about the thickness of a piece of paper.

In some cases, it is not practical to make a prototype because of size or cost. In other cases, the prototype would not respond the way the actual product would. For these situations, as well as others, **virtual reality (VR)** systems offer a viable analysis approach. VR systems use the principles of perception to develop completely immersive

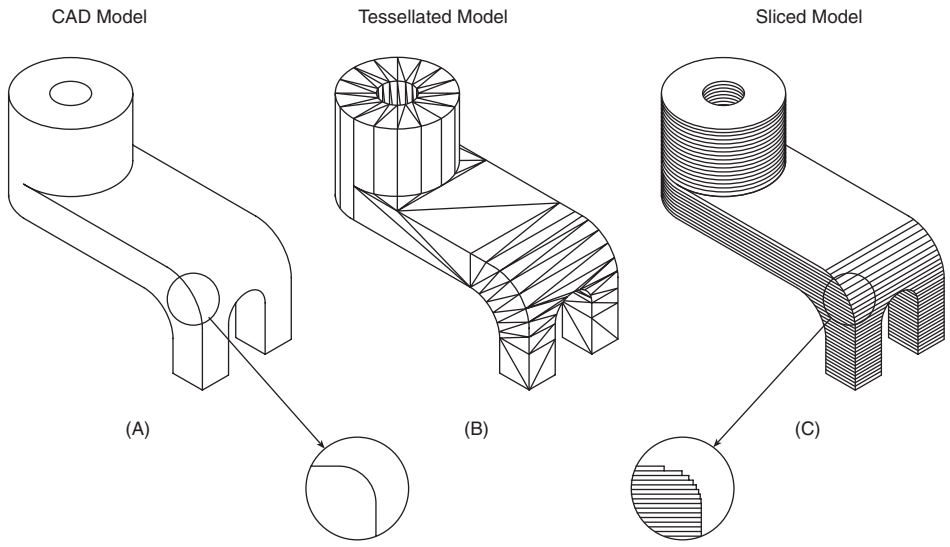


Figure 4.56 **Processing a model for rapid prototyping**
A solid model needs special preparation before a rapid prototyping machine can create a physical model of it.

environments in which the user can interact with the object through some or all of the senses. In such an environment, the user has the feeling of actually interacting with the virtual model.

Kinematics **Kinematics** is an analysis technique used to evaluate the design of a mechanism, that is, an assembly of multiple parts, some of which move with respect to other parts. A mechanism contains two main components: the part models themselves, which represent the links, and the joints, which describe how the links can move relative to each other. Joints constrain movement by controlling the **degrees of freedom** allowed between the parts (see Figure 4.53 on page 173). Often the assembly modeler can double as a kinematic analysis tool by both allowing varying degrees of freedom and specifying the range of motion for each degree of freedom (e.g., the arm can rotate 160 degrees about the pin on the Z axis).

Mass Properties Analysis Additional information about the model can be obtained by performing a **mass properties analysis**. With those 3-D modeling systems capable of calculating the volume of a solid, density values can be added to calculate the overall mass of the solid. In addition, the centers of gravity (centroids) and the inertial properties also can be calculated. Such calculations are used either on a single solid of uniform density or on a complete assembly containing parts of varying materials and densities. A simple but important application of this analysis involves calculating the final shipping weight of a product.

Finite Element Analysis Mass information helps calculate the forces acting on a part but not necessarily how the part responds to those forces. A real-world object is a continuous mass that responds in a very complex manner to forces acting upon it. Currently, only the responses of very simple geometric shapes can be calculated easily. A process called *discretization* divides more complex geometries into simpler forms so that the response of a solid to forces can be estimated. The process of creating a model of primitive geometries is called finite element modeling (FEM), and the analysis done on the resulting model is **finite element analysis (FEA)**, as in Figure 4.57.

Ergonomics **Ergonomics** examines the interaction between technology and humans, as in Figure 4.58 (on page 178). The point of interaction could be the handgrip on a vacuum cleaner, the seat in a car, or a control panel in a nuclear power plant. Ergonomic analyses revolve around the goals of functionality, efficiency, and safety. With virtual models, both the products and the human operators can be modeled. More sophisticated human modelers allow various anatomical components to be modeled as separate

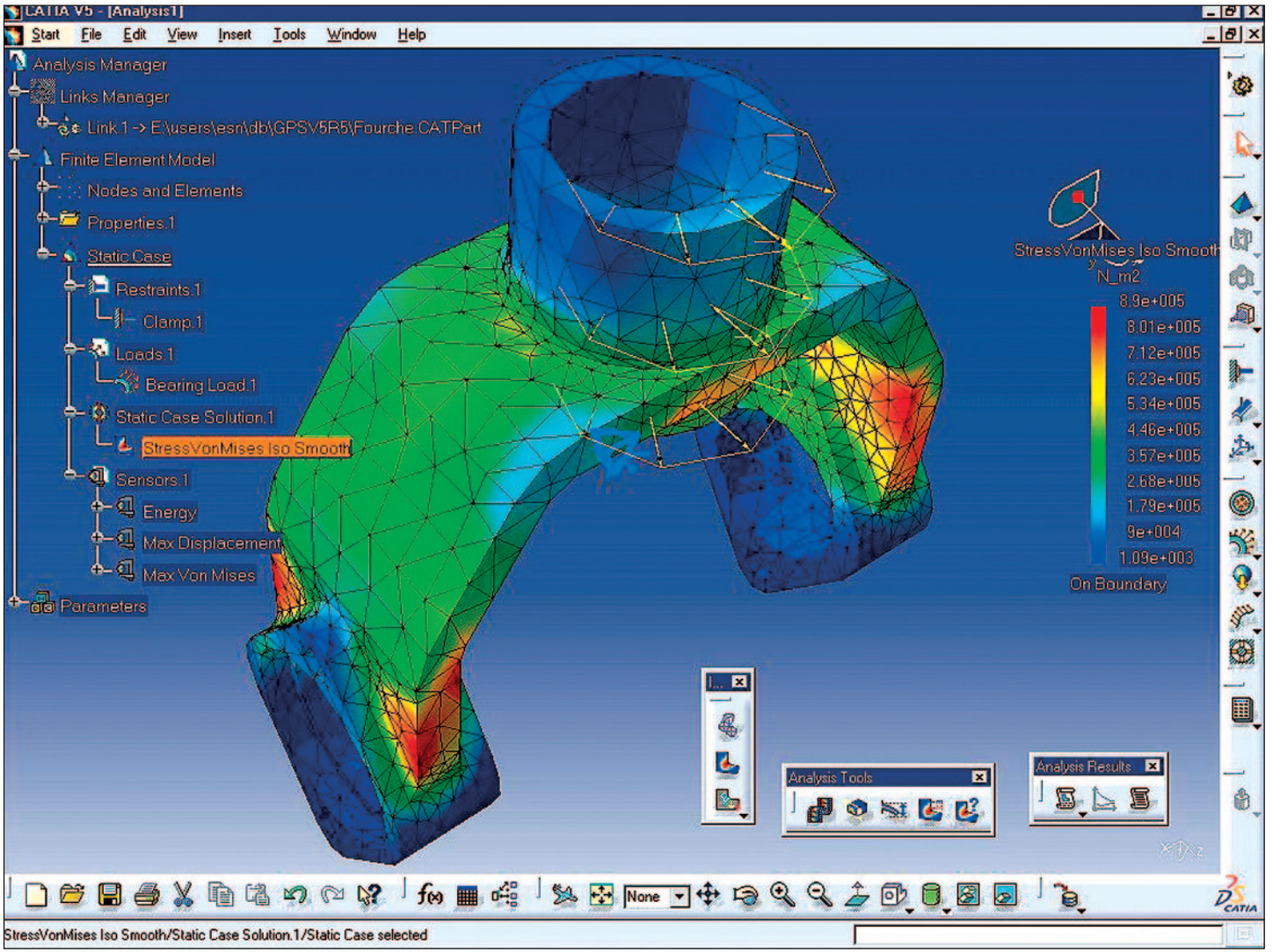


Figure 4.57 **Finite element model**

(Courtesy of Dassault Systemes.)

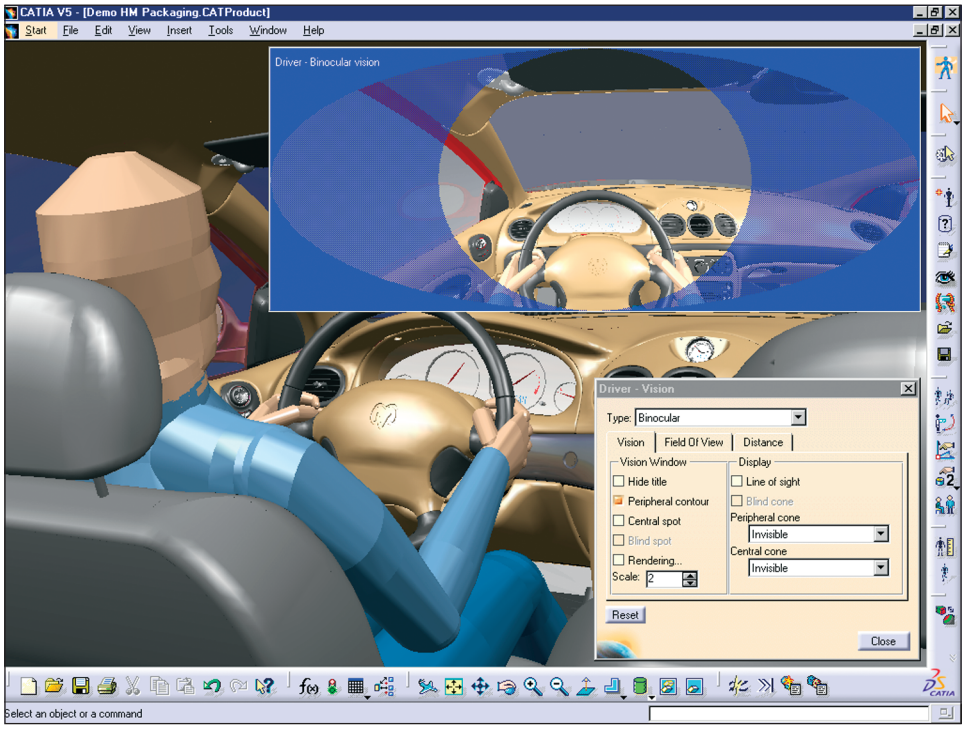


Figure 4.58 **Human interaction and ergonomic simulation within a 3-D car**

(Courtesy of Dassault Systemes.)

parts and linked to a kinematic model. These models are manipulated to mimic how a human would walk, bend, crawl, etc., through a given space or to evaluate whether certain controls are too far away to reach. *Reach envelopes* can be swept for human limbs in the same way they are for mechanical parts. Other geometric solids are created to represent other limits of human capability. For example, a right-angled cone is used to represent the *cone of vision* of an aircraft pilot. The intersection of this cone with the cockpit model indicates the controls that can be seen by the pilot at that moment.

Computer-Aided Manufacturing (CAM) Three-dimensional modeling techniques can be combined with **computer-aided manufacturing (CAM)** capabilities to ensure that a product design satisfies the desired manufacturability requirements as closely as possible. Three-dimensional models and their associated databases ease the transition from design to manufacturing by reducing or eliminating the need for traditional working or production drawings. In many instances, the computer-generated model and database can be translated directly to the computer system controlling the manufacturing operation. As with analysis techniques, unidirectional and bidirectional associativity between the CAD model and the CAM tools means reduced time and errors in generating manufacturing information.

The first step in the manufacturing of a product is process planning, in which the most efficient approach for producing the product is determined. Since individual parts are manufactured separately, the product, and therefore the model, is divided along its natural hierarchical structure. The parts also can be separated between those parts that are ready-made and those to be fabricated on-site. For those that are to be fabricated on-site, models can be made showing what each part will look like at each stage of the manufacturing process (Figure 4.59). These models provide information used to determine how much time, material, and labor would be required to manufacture the product as modeled. If special types of tooling (e.g., cutters, jigs, etc.) are required, 3-D models also can be made for them.

Increasingly, the machinery used for fabrication is programmed using the computer models of each part. The

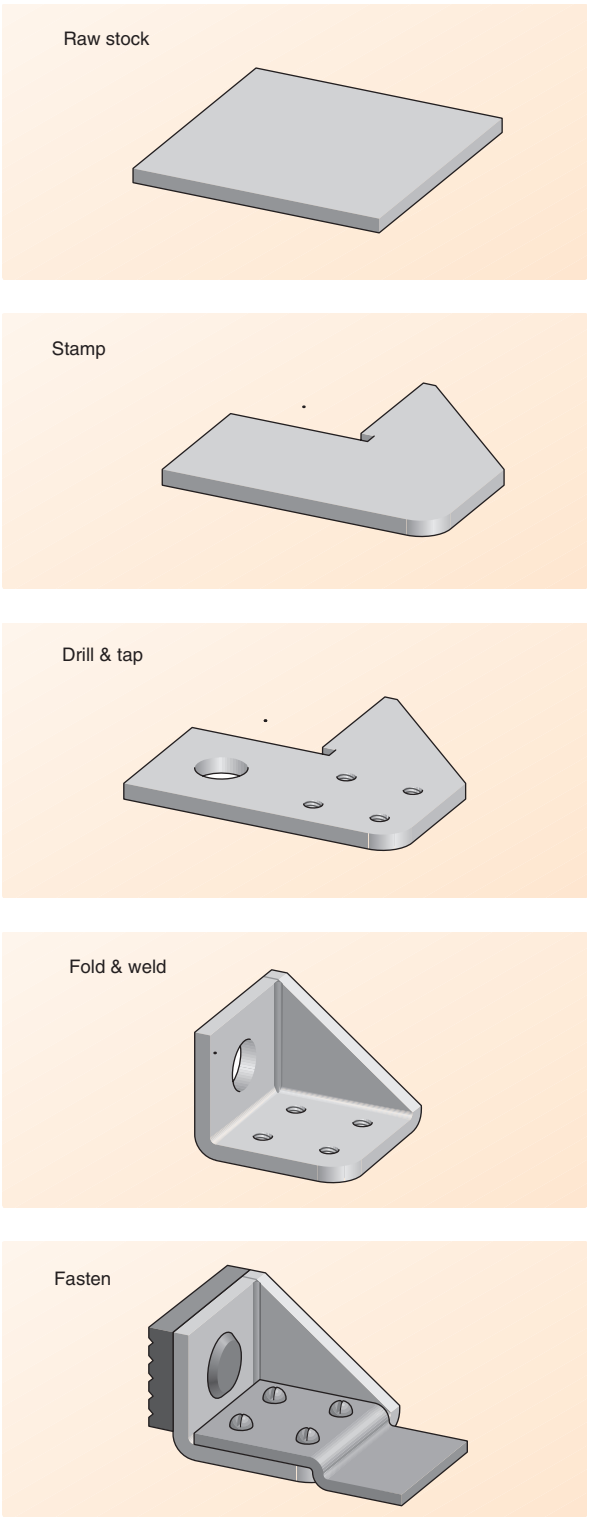


Figure 4.59 **Modeling the incremental fabrication of a part**
Virtual 3-D models can be used to simulate the manufacturing process.

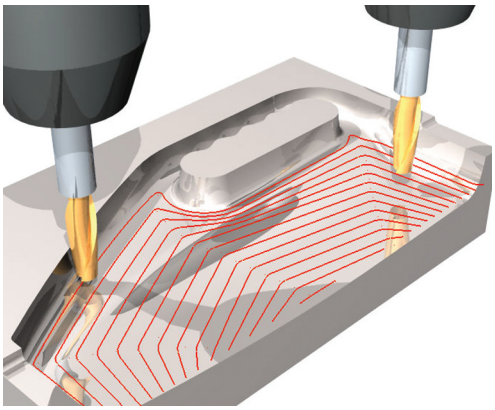


Figure 4.60 **Planning tool paths using a virtual 3-D model**
Tool paths can be troubleshot on virtual models without risking damage to expensive equipment.
(Courtesy of Delcam, Inc.)

information related to the model is translated into manufacturing operations by specialized programs. These programs control the machine tools through a process called numeric control (NC). Originally, information was provided to NC machines by punched tapes. Improvements in technology have led to the full-scale integration of computers with machine tools and the development of computer numeric control (CNC). The use of CNC means less translation and less chance for error. In the current generation of CNC technology, simulations of the tool-cutting action are created and tested on virtual models before actual materials and equipment are used (Figure 4.60). This cuts down on material waste and reduces troubleshooting time, freeing up the equipment for greater use in production.

4.12 Summary

Three-dimensional solid modeling is becoming the standard method in engineering for developing product designs for many industries. The advantages of using 3-D modeling versus 2-D drafting are numerous. New technical design methods require the use of intelligent graphics, that is, graphics in the form of 3-D models that contain important information beyond the basic geometric shapes. The more information contained in the model, the more useful the model is for designing, manufacturing, marketing, and servicing a product or structure.

Online Learning Center (OLC) Features

There are a number of Online Learning Center features listed below that you can use to supplement your text reading to improve your understanding and retention of the material presented in this chapter.

- Learning Objectives
- Chapter Outline
- Questions for Review
- Multiple Choice Quiz
- True or False Questions
- Key Terms
- Flashcards
- Website Links
- Animations
- Image Library
- AutoCAD Exercises
- Case Studies
- Stapler Design Problem

Goals Review

1. Understand the terminology used in 3-D modeling. All sections.
2. Define the most popular types of 3-D modeling systems. Sections 4.2, 4.3, 4.4, 4.5
3. Apply Boolean operations to 3-D objects. Section 4.3
4. Understand the role planning plays in building a constraint-based model. Section 4.5.1
5. Apply generalized sweeps to the creation of model features. Section 4.7.1
6. Apply construction geometry in the support of feature creation. Section 4.7.2
7. Apply constraints to a feature profile. Section 4.7.4
8. Understand how feature order affects feature editing and final model geometry. Section 4.8.1
9. Apply feature duplication to model construction. Section 4.9
10. Identify the elements used to define a view of a 3-D model. Section 4.10.1
11. Understand how model data associativity supports engineering design and analysis. Section 4.11.1
12. Generate 2-D documentation from a 3-D model. Section 4.11.2
13. Construct assemblies from part and subassembly models. Section 4.11.3
14. Define the types of analyses that can be used with 3-D models. Section 4.11.4
15. Understand how CAM information is derived from 3-D models. Section 4.11.4

Questions for Review

1. What is a nonmanifold object? Sketch an example of one.

2. Describe the differences and similarities of B-rep models and CSG models; do the same for wireframe models and B-rep models.

3. Define the three types of Boolean operations and sketch examples of each one. Can you derive the same final object using different Boolean operations and/or primitives?

4. What is design intent? Why does this play a role in planning the construction of a constraint-based model?

5. What are the basic elements of a generalized sweep? Describe the major types of generalized sweeps used in feature creation.

6. What are workplanes used for? What are five ways a workplane can be defined?

7. What is the difference between geometric and dimensional constraints? Give examples of four types of implicit geometric constraints.

8. Give an example of a parent-child relationship. How is a feature tree used to identify parent-child relationships?

9. What are the two primary types of duplication methods? What input parameters are needed to define each one?

10. What are the elements used to define a view of a 3-D model? Which types of view commands don't change the projection of the model?

11. What is the difference between unidirectional and bidirectional associativity? What are the advantages and disadvantages of bidirectional associativity?
12. How is the base view used in generating multiviews from a 3-D model?
13. What are the two primary ways of constraining parts in an assembly? What modifier is often used with these?
14. Define the types of analyses that can be used with 3-D models. Will all of these always be used when designing a part?
15. What advantages are there for using data from a 3-D model when analyzing the manufacture of a part?

Further Reading

LaCourse, Donald, ed. *Solid Modeling Handbook.* New York: McGraw-Hill, 1996.

Machover, Carl. *CAD/CAM Handbook.* New York: McGraw-Hill, 1996.

Mortenson, Michael E. *Geometric Modeling.* 2d ed. New York: Wiley, 1997.

Smith, P. and Silva, W. *Introduction to Solid Modeling.* Distance Engineering Incorporated, 1999. CD-ROM.

Workbook Problems

Workbook Problems are additional exercises to help you learn the material presented in this chapter. The problems are located at the back of the textbook and can be removed from the book along the perforated edge.

4.1 **Circular Sweep.** Sketch the resulting solid model if the given profiles were to be circularly swept 360 degrees about the Y axis.

4.2 **Linear Sweep.** Sketch the resulting solid model if the given profiles were to be linearly swept 2 units along the +Z axis.

4.3 **Boolean Operations.** Given the three overlapping solid primitives, make an isometric sketch of the resulting solid after applying the following Boolean operations: A – B – C.

Problems

4.1 (Figure 4.61) Create wireframe or solid models by sweeping the profiles shown in the figure using a scale assigned by your instructor.

Do the following with each of the profiles:

a. Sweep linearly 5 units along the +Z axis.
b. Sweep linearly along the vector (2,–3,5).
c. Sweep 360° about the Y axis.
d. Sweep 360° about the X axis.
e. Sweep 90° about the +X axis.
f. Sweep 270° about the −Y axis.
g. Sweep 360° about a Y axis offset 2 units in −X direction.

4.2 Create a coffee mug using sweep operations. In a sketch, clearly define the profile shapes to be used and the axes about which they will be swept.

4.3 In Figure 4.62, there are 12 objects swept using 12 different profiles. Match the objects with the same profile used to create 3-D objects. (*Hint:* Unlike Problem 5.1, the profiles may not always be swept at axes perpendicular to each other.)

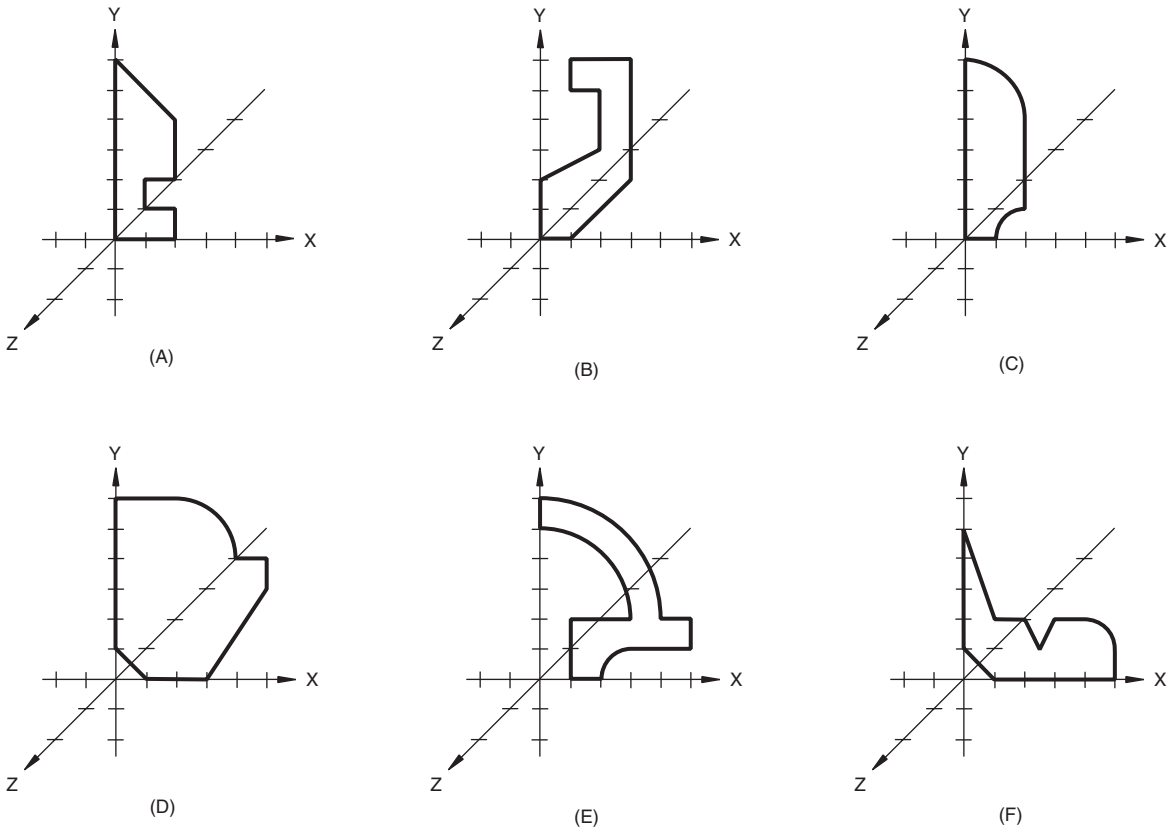


Figure 4.61 **Profiles to be swept**

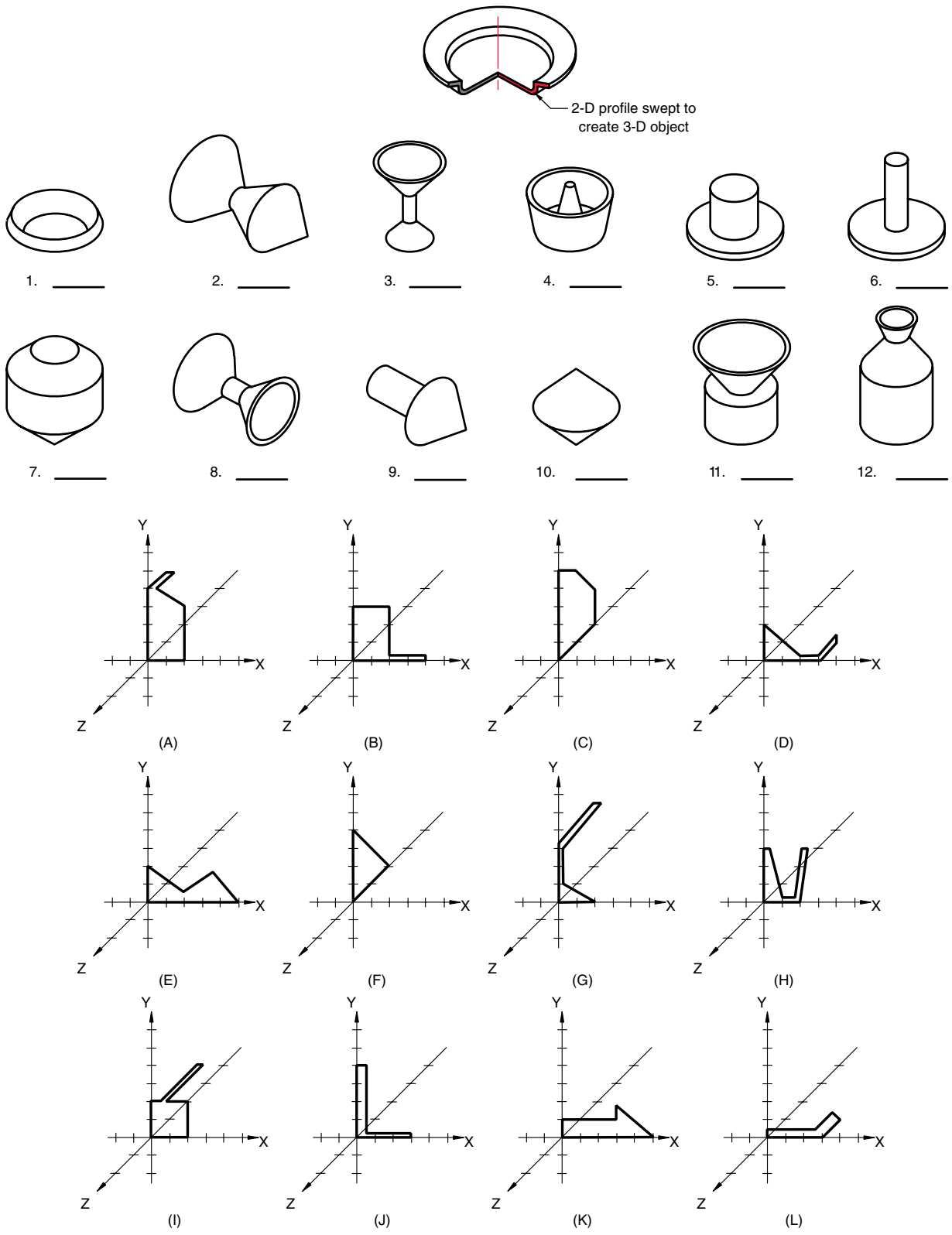

2-D profile swept to create 3-D object
1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
Y
X
Z
(A)
(B)
(C)
(D)
(E)
(F)
(G)
(H)
(I)
(J)
(K)
(L)

Figure 4.62

4.4 Model the primitives in Figure 4.63A, using either general modeling techniques or specialized parametric primitive modeling tools and a scale assigned by your instructor.

a. Using purely additive techniques, combine the primitives to create the objects in Figure 4.63B.

b. Create at least five other objects, using combinations of the primitives. Use five or more primitives in each of the new objects.

4.5 Figure 4.64A–C contains groups of three overlapping primitives shown in wireframe. On separate sheets of isometric grid paper, sketch the objects resulting from the following Boolean operations:

Figure 4.64A:

a. (A ∪ B) ∪ C

b. (A ∪ B) – C

c. (A – B) – C

Figure 4.64B:

a. (A – B) – C

b. (A ∪ B) ∪ C

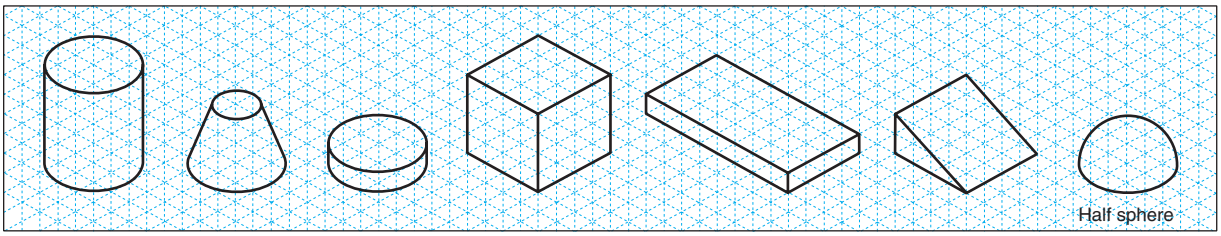
c. B – (A ∪ C)

Figure 4.64C:

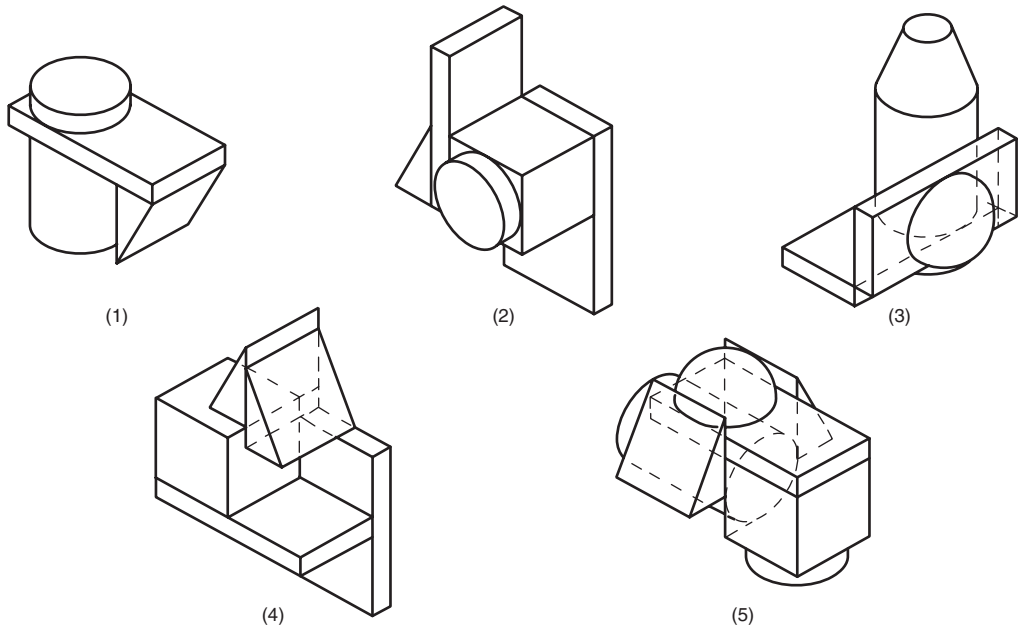
a. (C – A) – B

b. (A ∪ C) – B

c. (A ∩ C) – B



(A) Model these primitives



(B) Combine the primitives to make these objects

Figure 4.63 **Primitives to be modeled**

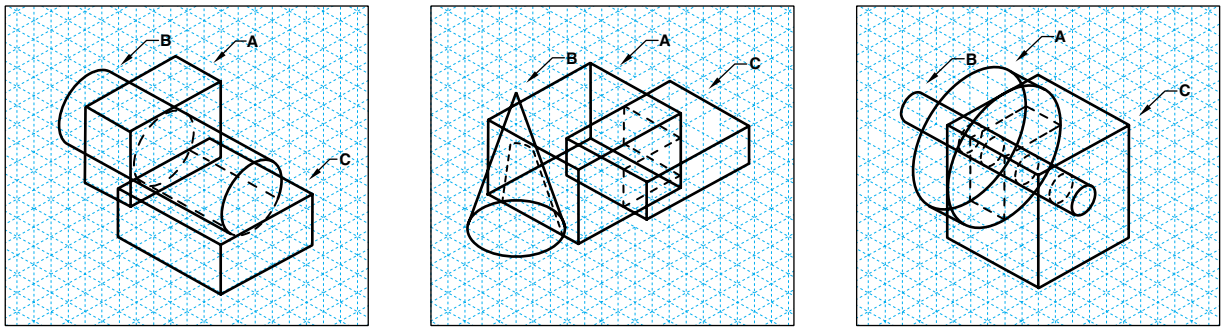


Figure 4.64 **Groups of three overlapping primitives shown in wireframe**

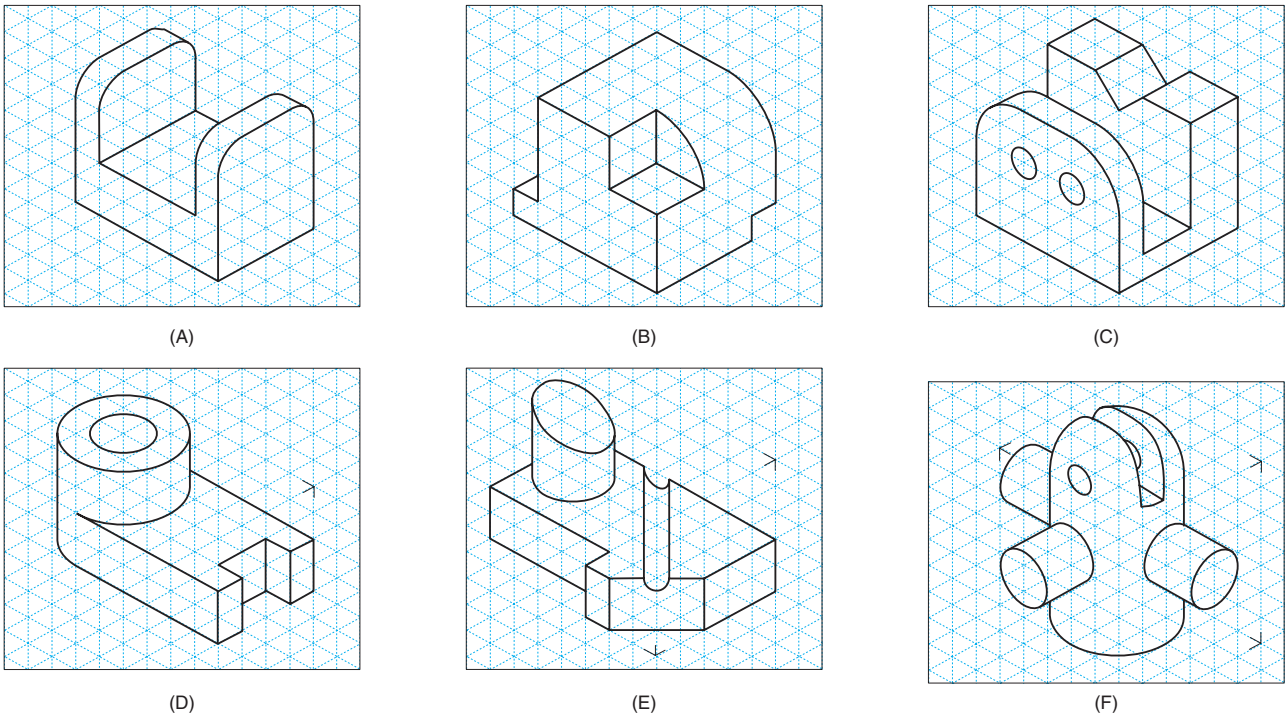


Figure 4.65

Circle 3-D models of the objects. All holes are through.

4.6 Create the objects in Figure 4.65 using wireframe or solid modeling techniques and a scale assigned by your instructor.

a. Print or plot an isometric pictorial, displayed as a wireframe.

b. Print or plot an isometric pictorial, with hidden lines removed.

c. Capture both standard orthographic views and an isometric view of the object. Organize these views in a standard border with a title block and print or plot the drawing.

d. Same as (c) except add dimensions and notes to the drawing.

4.7 Create the objects in Figure 4.66, using wireframe or solid modeling techniques and a scale assigned by your instructor.

a. Print or plot an isometric pictorial displayed with hidden lines removed.

b. Capture both standard orthographic views and an isometric view of the object. Use auxiliary and section views as appropriate. Organize these views in a standard border with a title block and print or plot the drawing.

c. Same as (b) except add dimensions and notes to the drawing.

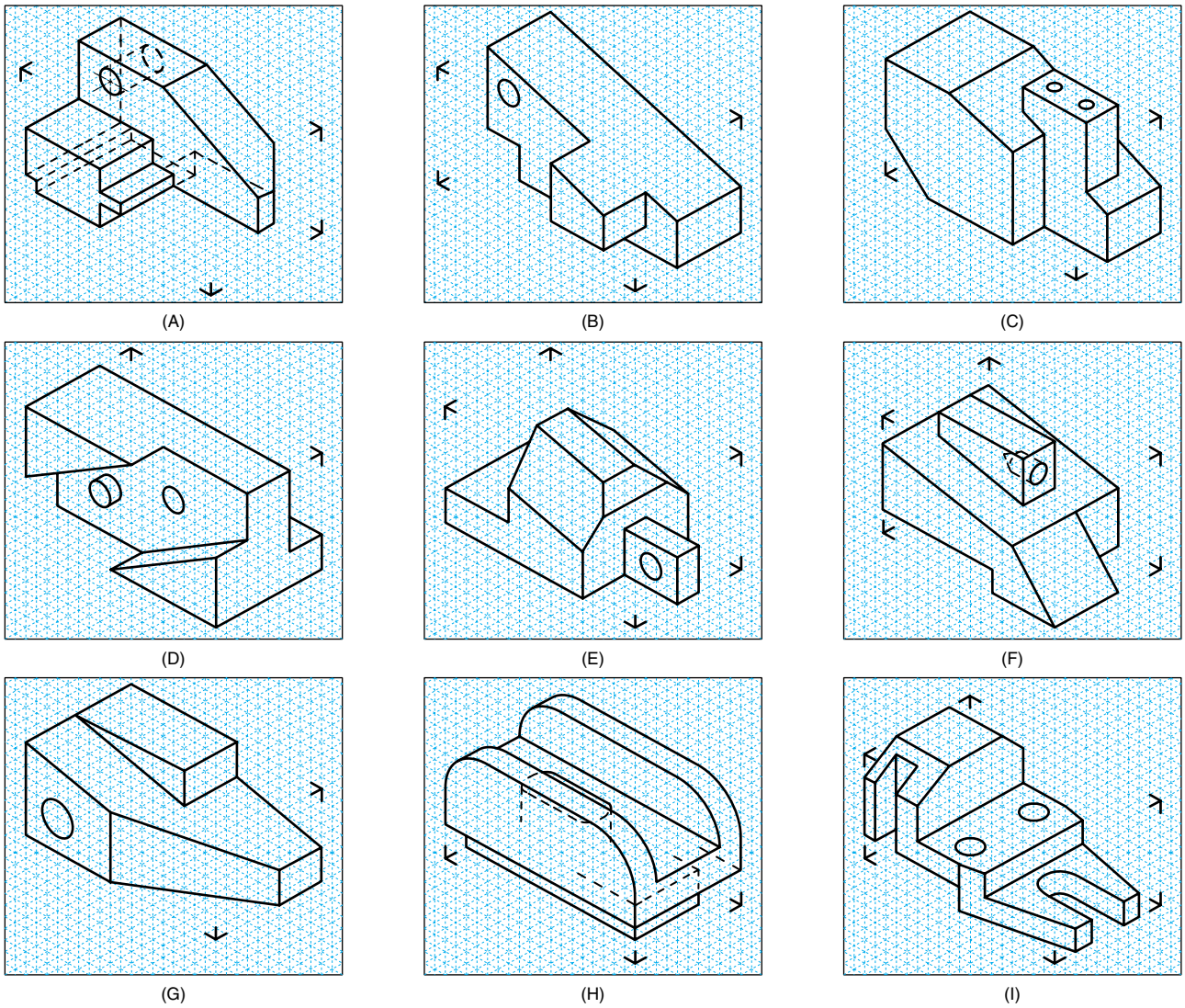
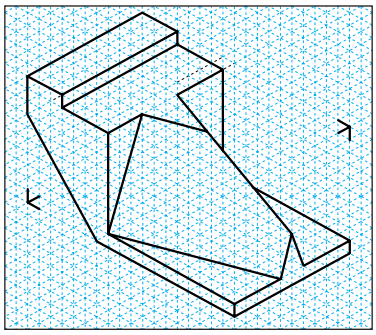
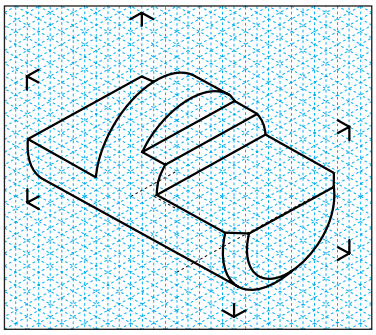
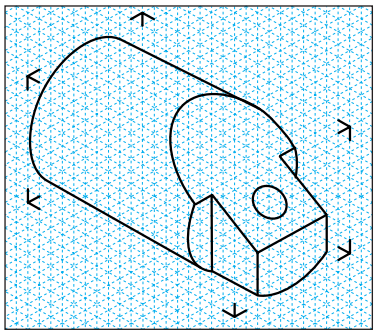


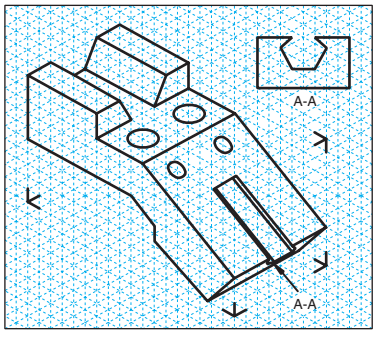
Figure 4.66

Create 3-D models of the objects. All holes are through unless otherwise indicated with dashed lines.

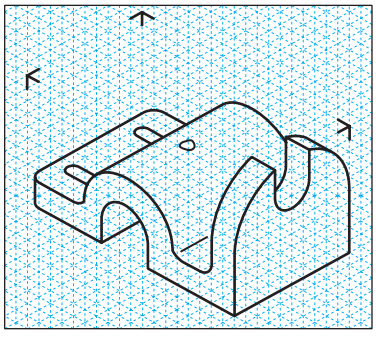

(J)

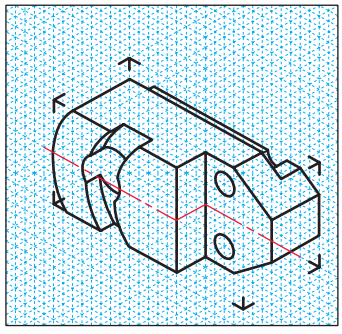

(K)

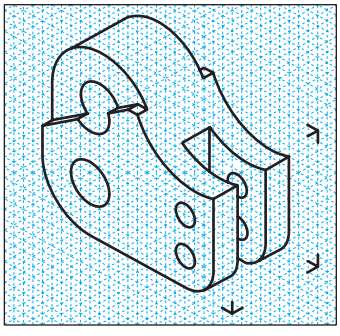

(L)

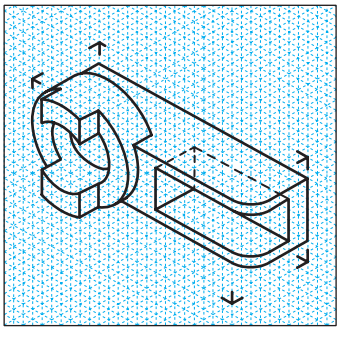


(M)


(N)


(O)


(P)


(Q)

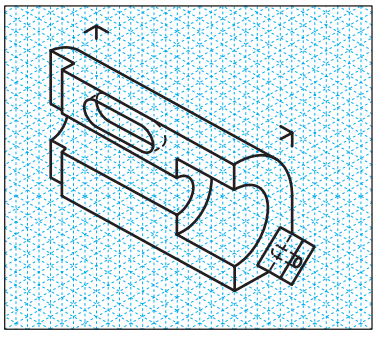

(R)

Figure 4.66 **Continued**

Create 3-D models of the objects. All holes are through unless otherwise indicated with dashed lines.

4.8 Model the assembly shown in Figure 4.67 on the next page.

a. Organize an exploded assembly from the parts. Capture the following views of the object, place the captured views in a standard border with a title block, and notate appropriately.
   (i) Axonometric view
   (ii) Axonometric view, rendered with color to code the parts

b. Organize the parts in their assembled position. Capture the following views of the object, place the captured views in a standard border with a title block, and notate appropriately.
   (i) Axonometric pictorial view.
   (ii) Front orthographic view, sectioned.
   (iii) Axonometric pictorial view, rendered and using transparency techniques to reveal interior detail.

c. With the front and back housing (parts 3 and 6) fixed, the shaft (1) rotates the cylinder (5), which, in turn, actuates the "L" pin (7). The screw (8) attaches the "L" pin to a vertical

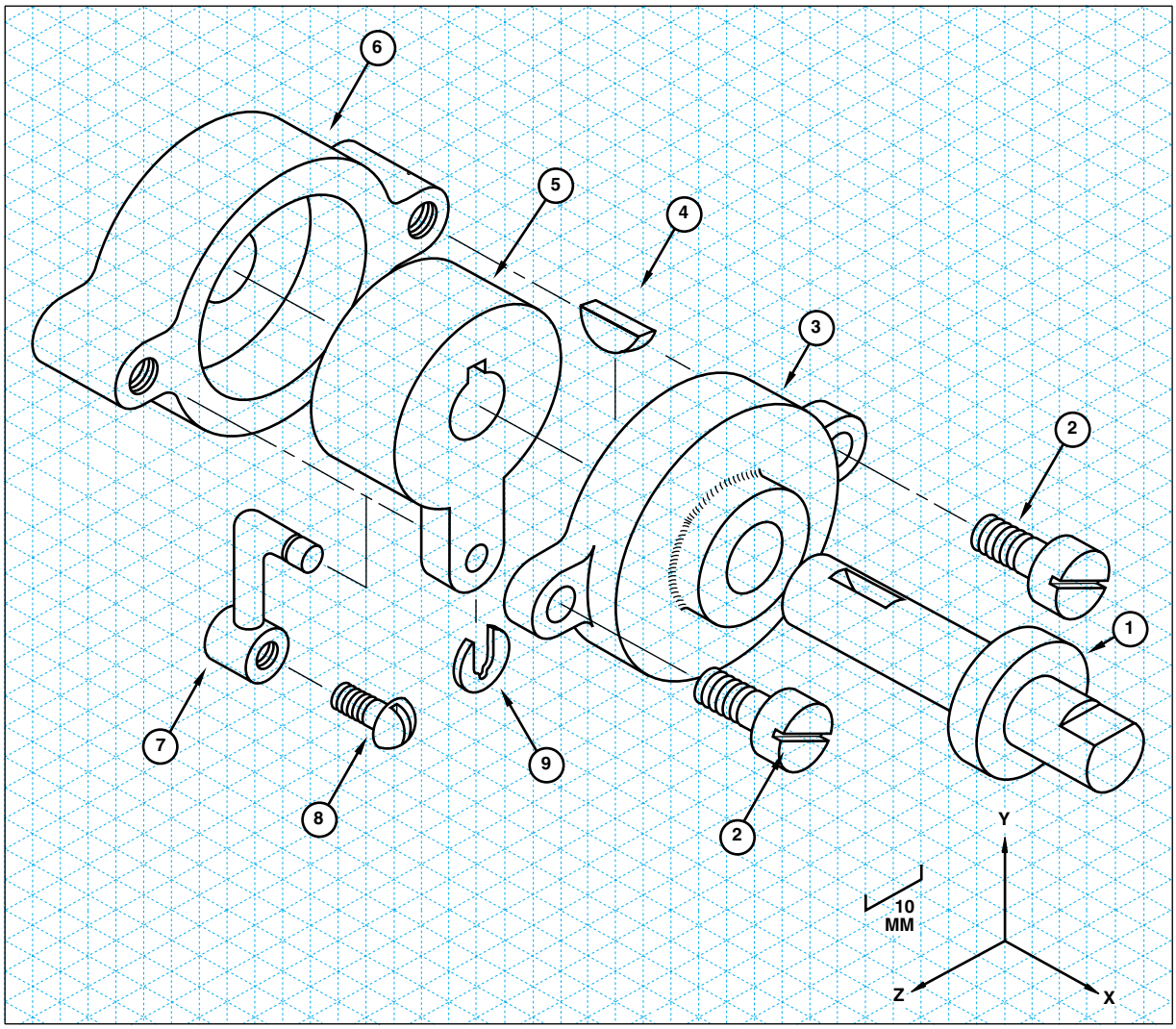


Figure 4.67 **Assembly to be modeled**

slider (not pictured). Analyze how far the shaft would have to rotate in order to move the vertical slider 5 mm. The "L" pin is vertical when the cylinder is in the position shown in the figure. Represent this analysis as follows:

(i) As an axonometric pictorial, using phantom lines to show the movement

(ii) As a pair of orthographic views looking down the primary axis, showing the mechanism in its two positions

(iii) As a computer animation

4.9 (Figures 4.68–4.81) Assign different Boolean operations to the eight assembled primitive parts; then do the following:

a. Sketch the resulting composite solids.

b. Use a solid modeling software to create the primitives with the given dimensions; then verify your sketches by performing the same Boolean operations on the computer.

4.10 (Figures 4.78–4.81) Using the given information for feature-based modeling, do the following:

a. Using a scale assigned by your instructor, measure the profiles and workpiece. On isometric grid paper, sketch the resulting workpiece after the feature-based modeling is performed.

b. Do the same operations with CAD and compare the results with your sketch.

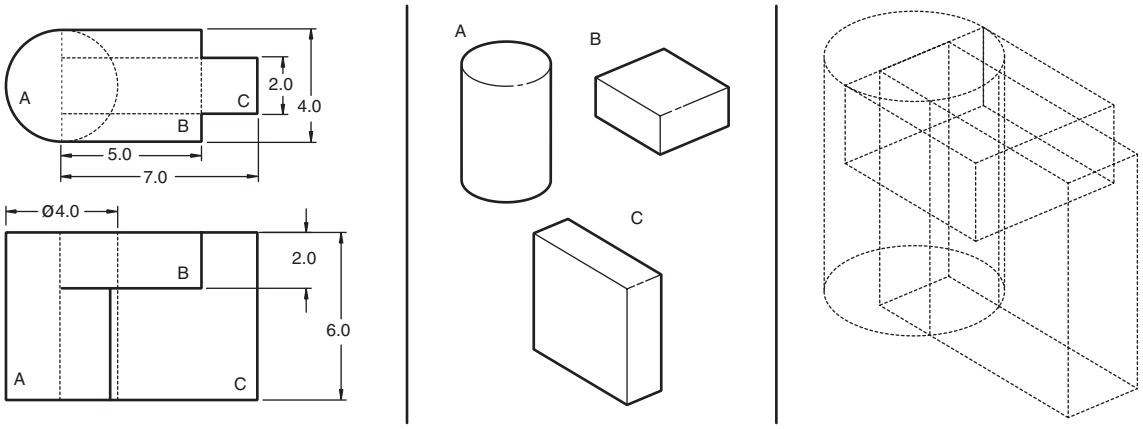


Figure 4.68 Assembled primitive parts for Boolean operations

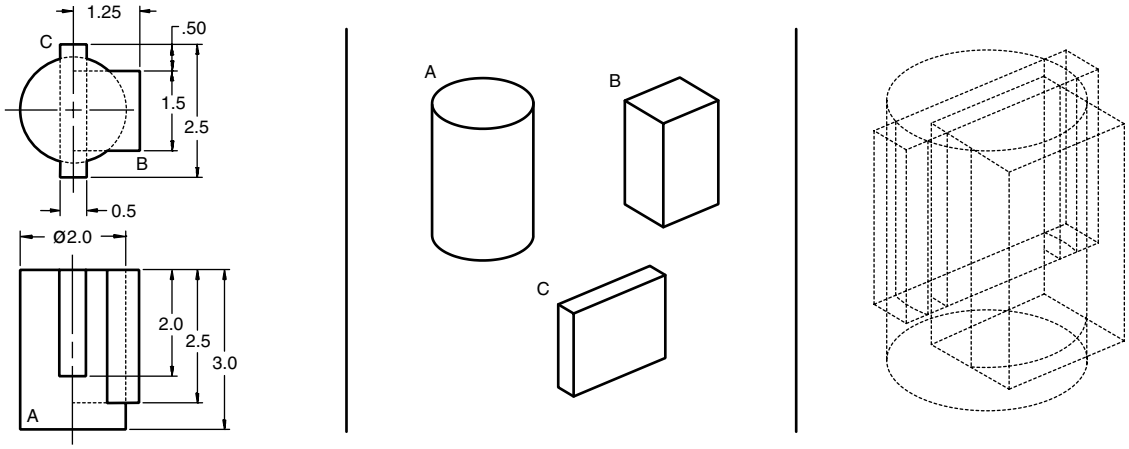


Figure 4.69

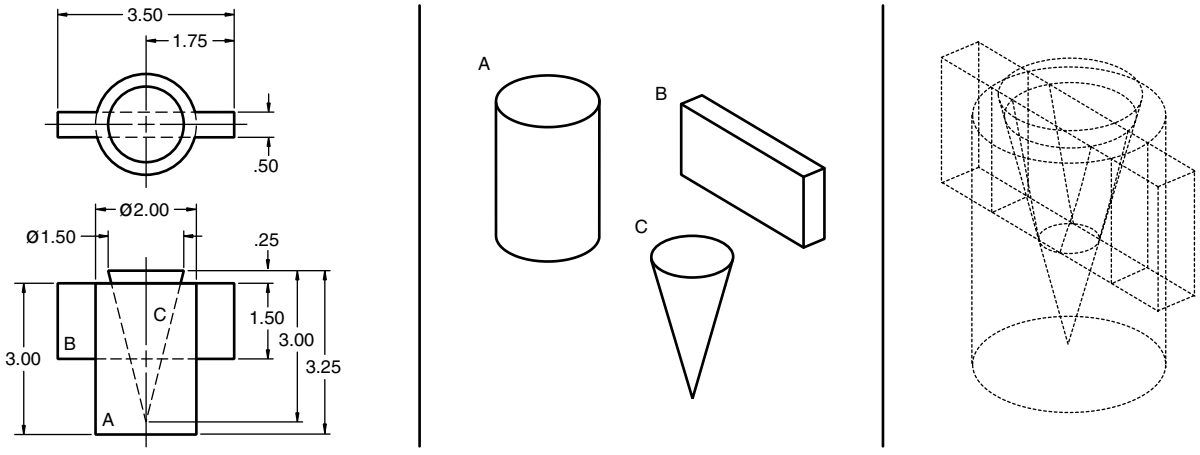


Figure 4.70

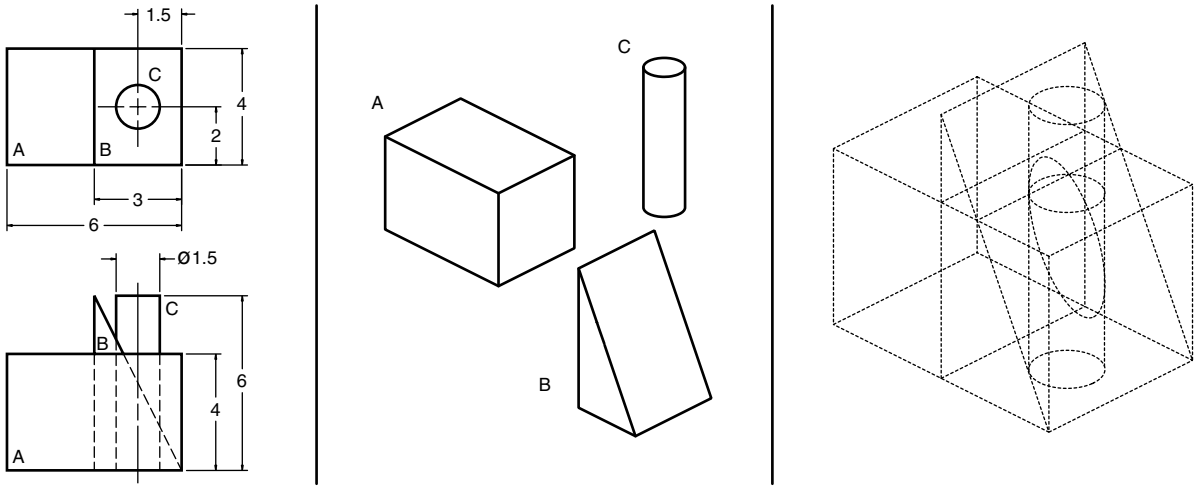


Figure 4.71

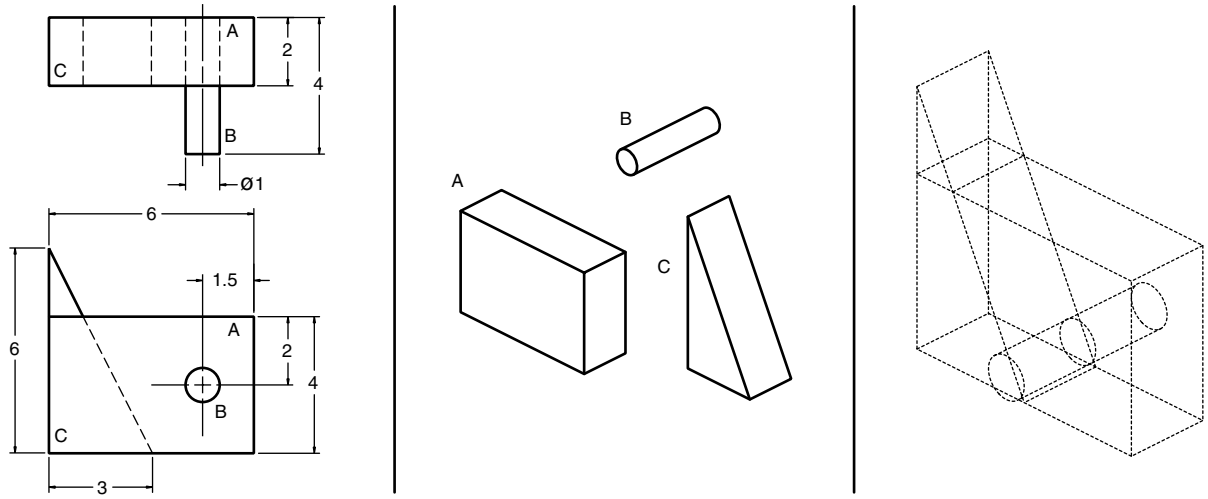


Figure 4.72

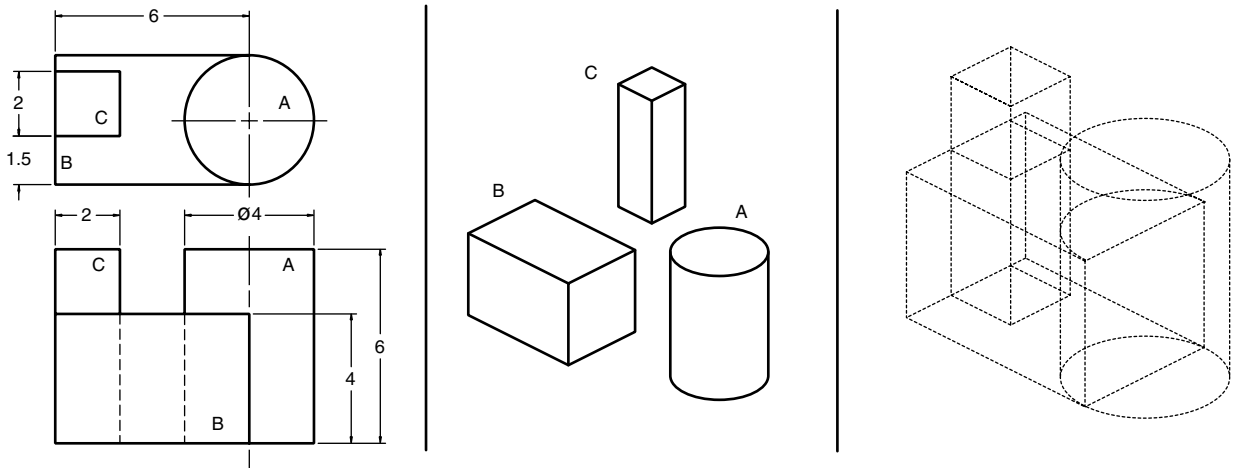


Figure 4.73

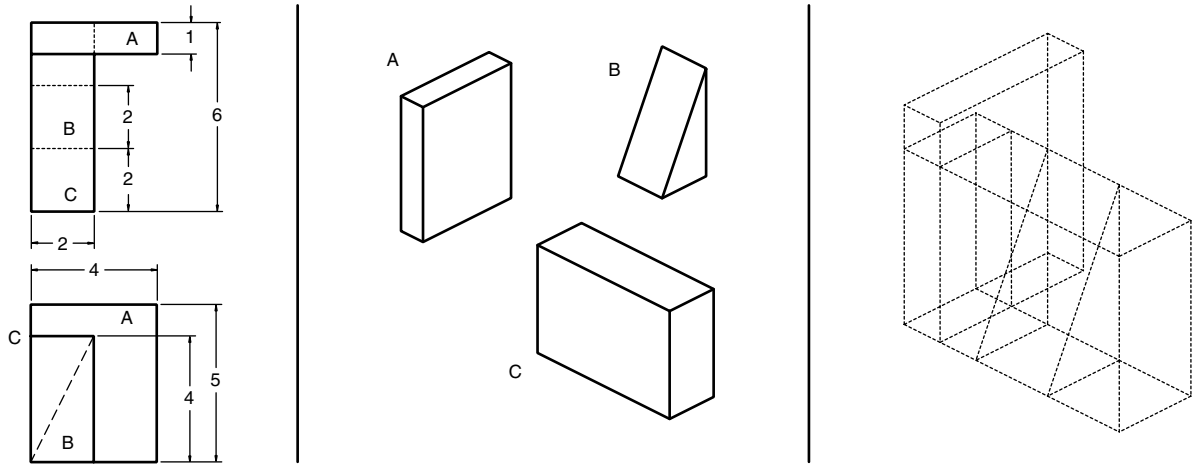


Figure 4.74

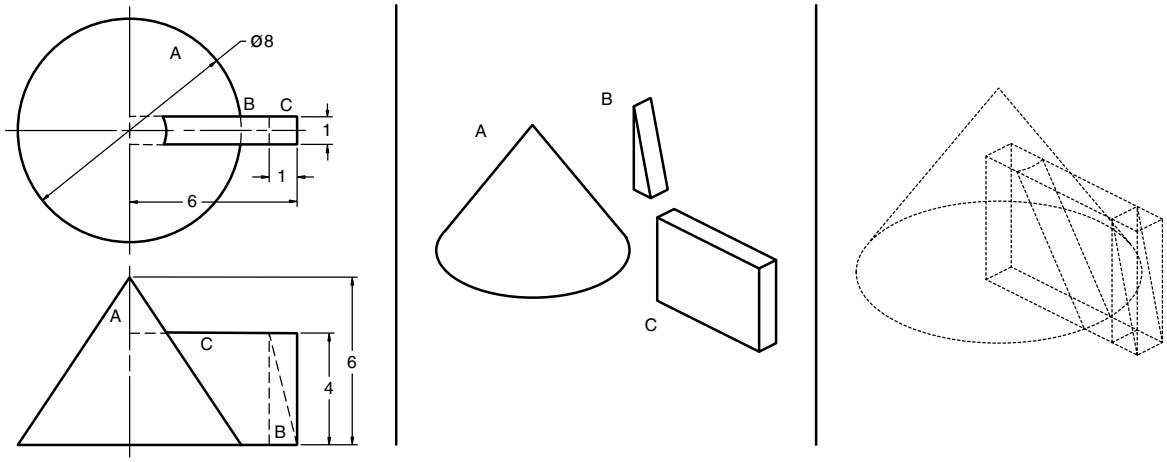


Figure 4.75

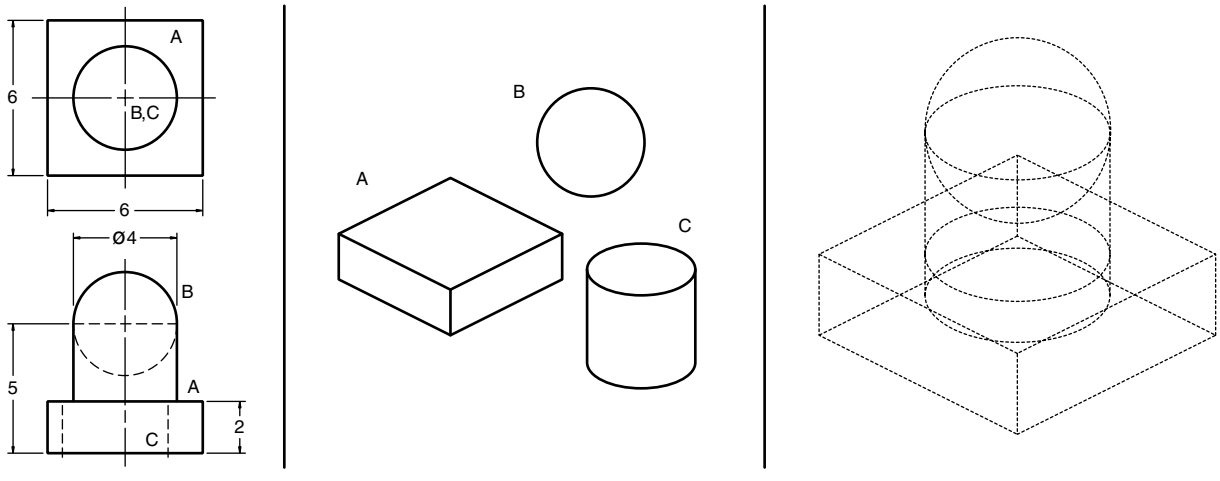


Figure 4.76

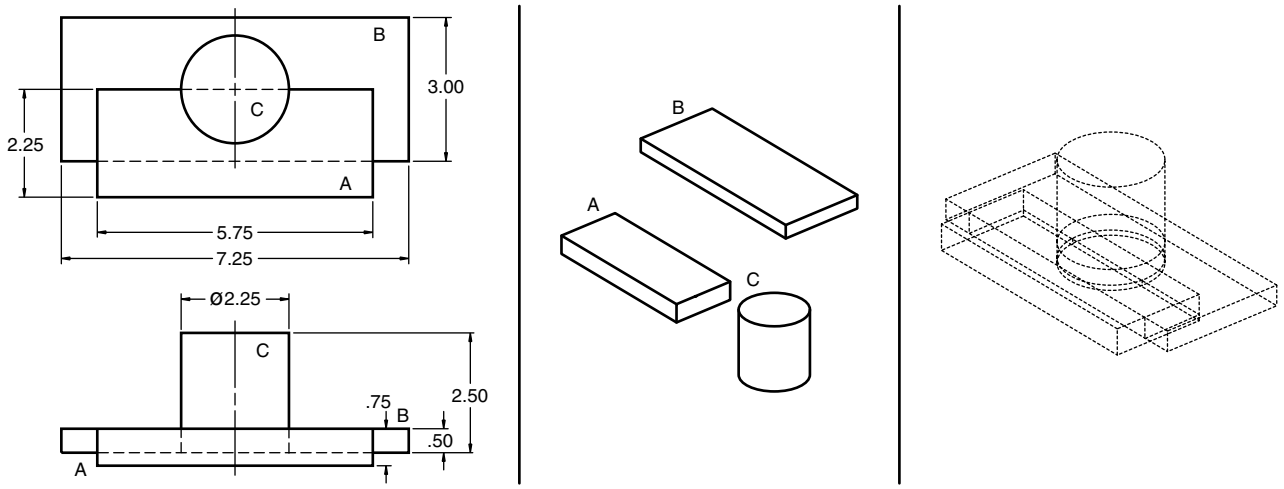


Figure 4.77

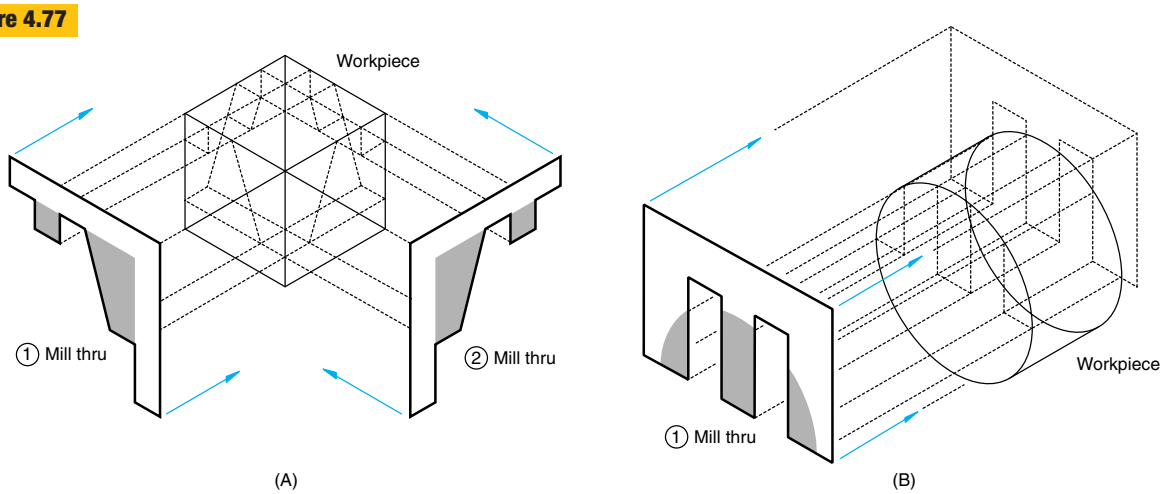


Figure 4.78 **Feature-based modeling information**

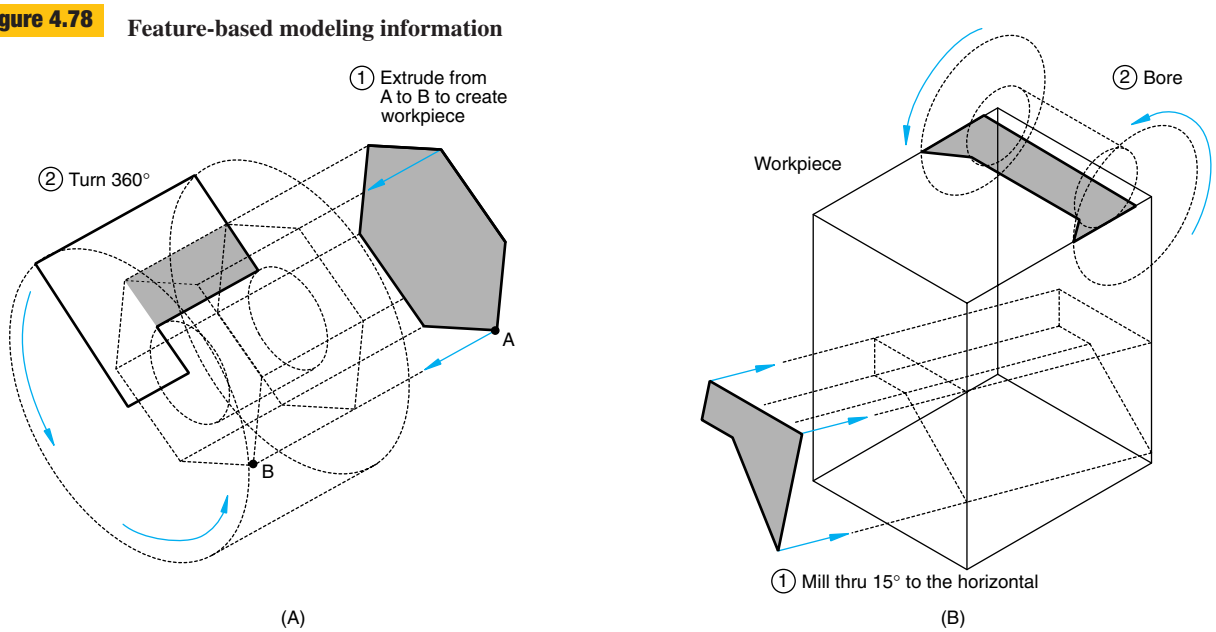


Figure 4.79

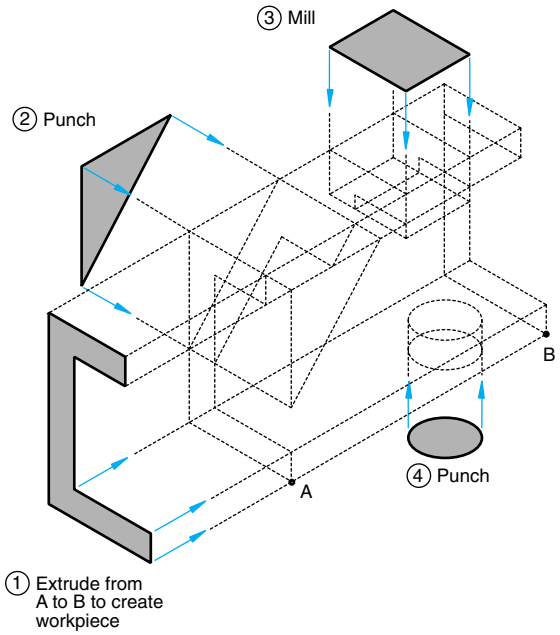


Figure 4.80

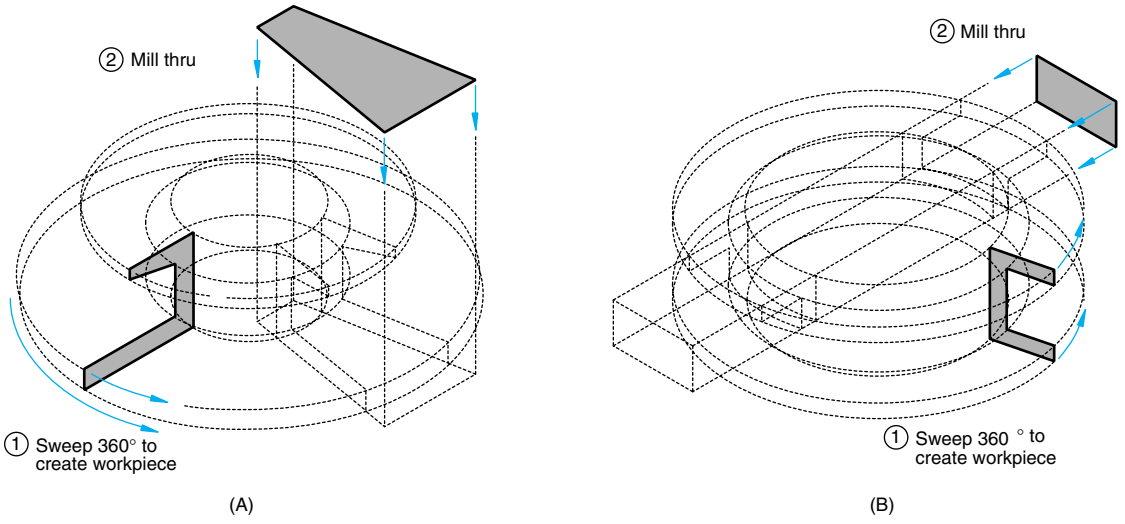


Figure 4.81