Development Of A Feature Based Rapid Design Environment

Utpal Roy, Daniel Panayil
Syracuse University

Abstract

Computer based realistic projects are required in all of our undergraduate design related courses. Students are encouraged to design their individual projects (related to some real-life design and manufacturing problems) using state-of-the-art computer-tided design (CAD) tools. However, due to some inherent inflexibilities of current CAD tools, students cannot use them efficiently. It is now recognized that feature-based modeling holds the promise of developing requisite design tools for rapid and efficient design systems. This paper discusses the feasibility of using a design-by-feature approach in the students' design projects. It reports the development of a library of macro, micro, and functional features for establishing a prototype, rapid design environment in SDRC's I-DEAS solid model based design environment (using I-DEAS's programmability).

1.0 Introduction

Designing an object using a currently available CAD tool is a very tedious task for a designer. Making the component of his/her design from the primitives, extruded or revolved profiles takes up more time than he/she spends on designing that object. The designer also faces the problem of redesigning most of the time due to cost, manufacturing and assembly reasons. The current design systems do not provide any user--friendly design environment where the designer can modify his/her design easily according to the desired design constraints. We have had such experiences in one of our undergraduate design classes, where a group of five (5) students was assigned to design a fire truck. The group had to design several major and minor parts. Each part is again made of several small, intricate design components. For example, the door-knob on the door has many components inside it like springs, lever, screws and pins. The screws that were used for the hinges and the door-knob handle were the same except that they were of different sizes. So the students had to make the same screws of different dimensions several times. It took them more than a week to figure out how to make these parts using the basic primitives and geometry. As the dimensions of the chair and the door depended on the width and length of the truck which was designed by the other members of the group, they had to modify the design several time as others made changes in their design. The same problems arise in other design projects as well. These are the reasons what motivated us to make a feature based design library so that other students wouldn't have to spend so much time in making similar parts.

The objective of the feature based design system is to accelerate the design process and to express the student's (user's) intention in design and manufacturing right on the feature itself. This feature based design not only makes it easier for the user to make his/her design but also provides a better representation of the design both for the designer and the manufacturer. This paper discusses how to create and use the features using the Feature Definition Task in I-DEAS solid modeler in order to develop a feature based rapid design environment. It reports the development of a library of macro, micro and functional features, and demonstrates the creation of several complex part using the design-by-feature approach.

2.0 Development of the System: Feature-Based Geometry Construction

In order to develop the feature-based rapid design system, we develop a library of three (3) kinds (i.e., Macro-, Micro- and Functional) of generic features, instead of providing a feature set which is only useful to a particular application domain. These generic features are callable at any phase of the design and are instantiable to produce any application-dependent features as required. Our set of macro features includes regular bolt, stud bolt, spring, shaft, spur gear, bevel gear, helical gear, roller bearings with option of cylinder or ball rollers and pulley with arms or web option. The micro features include counterbore, countersink, elbow, bend plate, wedge, seam, wireedge, round-end slot, cap, flange, rib, offset plate and

parallelogram. Currently the functional feature includes only the "twist" function which can be used to twist any part of an object placed in the workplane. Each type of the stored features is represented by an object class. Each feature class contains attributes which describe the characteristics and behaviors of its members. In order to instantiate, the user needs to supply values for all required attributes. The new instantiated feature thus generated, can be positioned and oriented as required to build the desired design object. The feature library has been implemented in I-DEAS programmability. A sample feature program file has been shown in figure 1. Each feature module typically consists of the following four steps: (1) parameter extraction, (2) additional parameter request, (3) feature creation, and (4) feature positioning (and orienting) in the design object. Following is a brief description of the development of those features that are stored in the feature library.

2.1 **Development of Macro Features**

All macro features have been coded in I-DEAL (1-DEAS's programming language) and stored as program files in the library. The programs use I-DEAS's object modeling facilities (specifically its object modeling task and construction geometry task) liberally to create the features. In the following subsections, we discuss some of the macro features and their development processes.

2.1.1 *Bolt-Regular and Stud*

The bolt program file allows the user to make a regular bolt or a stud bolt with a square or hexagonal head and a nut. It asks the user to input the diameter, the height of the bolt and the pitch of the thread. To create the bolt (Figure 2(a)), a cylinder is created first from the radius and length entered by the user and it is stored in a directory. A thread is then created to join with the cylinder. It is done by creating a cross-section of the thread in the construction geometry. This is dimensioned as a function of the pitch of the thread and diameter of the bolt, and is created using the 'profile from points' option in the construction geometry. This cross-section is placed at a distance of the radius of the cylinder (that has been created for the bolt) from the y-axis so that when this cross-section is revolved in the I-DEAS object modeling to form the thread, the inner diameter of the thread equals the diameter of the cylinder. This cross-section is then stored as a profile and called in the object modeling to revolve around the y-axis. The length of the thread is made to 4/5 th of length of the cylinder. The profile is revolved around the y-axis by an angle which is equal to 360* (4/5 *length of cylinder)/pitch of the thread. The thread and the cylinder are then joined together.

To create the head of the bolt, a profile of its cross-section is created and then extruded. This program has an option of creating square or hexagonal head as requested by the user. The square head can be made using the 'profile from points' option in the construction geometry. It is then extruded in the object modeling. However in the case of hexagon, a wireframe of one side of the hexagon is created first. The dimension of this side is found out by finding the internal angles of a hexagon(360/6) and applying the Pythagorean theorem. Five other copies of this side is made by rotating this line by 60 degrees having the origin as the pivot point and thus completing the hexagon. The wireframe is then turned to a profile by 'Autochain Profile' and stored. It is extruded in object modeling and joined with the cylinder and the thread. The nut for the bolt is made just by calling the head of the bolt from its directory and cutting it with the threaded cylinder.

2.1.2 *Gear-Spur and Helical*

This gear (feature) program asks the user to enter his/her choice of spur or helical gear, the pitch diameter, the axle diameter, height of the teeth, gear thickness and the pitch of the teeth as the inputs. The number of teeth for the gear is calculated as the integer of (pitch diameter *pi/pitch). Since the pitch entered by the user usually results in a fractional number of teeth, the pitch is corrected by the program by changing the number of teeth to the nearest whole number and recalculating the pitch. A tooth is created using splines and lines as a function of the base pitch, the pitch diameter and the tooth height. This tooth wireframe is then copied by rotating it round the z-axis to equal the total number of teeth in the gear. Due to minor inaccuracies in the calculations and round-off errors, there would occur tiny spaces between each tooth as they are rotated and copied. This problem is corrected by changing the tolerance of point coincidence from the default .002 to a slightly larger value so that the disjoint ends are coincided. After creating the autochain profile from the wireframe, it is stored. It is then extruded in the object modelling task with the specified thickness. The axle is created by calling the shaft feature from a universal file and cutting it with the gear (Figure 2(b)).

2.1.3 *Roller Bearing with Cylinder or Ball Roller Option*

This program asks the user to input the outer diameter of the bearing, its width, thickness of the outer and inner rings, the diameter of the bore and the option of cylinder or ball rollers. The outer and inner rings **of** the bearing are made by creating two tubes from dimensions entered by the user (Figure 3(a)). When the cylinder roller option is chosen, the program makes a cylindrical race which guides the cylinder rollers. This cylindrical race is made by cutting the outer and inner rings with a tube. The middle ring which holds the cylinder rollers in place is also made from a tube. The program then makes holes in the middle ring where the

cylinder rollers are going to be placed. One hole is made by cutting the middle ring with a rectangular block. The other holes are made similarly by a loop in the program which rotates the block by an angle (the angle rotated depends on the diameter of the bearing) and cuts a hole in the middle ring after it rotates. The loop terminates when the block has been rotated by 360 degrees. The cylinder rollers are created in the same loop and placed in each hole in the middle ring.

When the ball roller bearing option is chosen, the program makes a ball race which guides the ball rollers. This ball race is made by cutting the outer and inner rings of the bearing with a torus. The torus is made by creating a circular profile in the construction geometry and revolving it by 360 degrees in the object modeling. The middle ring and circular holes for the ball rollers are made in the similar fashion as they are made in case of cylinder rollers.

2.1.4 *Pulley with Arms or Web Option*

The program asks the user to provide the diameters of the driven and driver pulleys, the diameters of the holes (in the hubs) of the driven and the driver pulleys, the width and the distance (center to center) between the pulleys and the number of arms if the arms option is chosen (Figure 3(b)).

The face of the pulley is created frost by making a tube. Then the hub is made out of a cylinder. The diameter of the hub is made to be twice the diameter of the hole. The arms for the pulley are created in the construction geometry using skin groups in the z-direction. Since the top width of the arm is three quarters of the base width, two sections for the skin groups are made, one with the base width and the other with three-fourths of the base width. These two sections are placed apart a distance equal to the length of the arm in the z-direction. Then they are skinned in the object modeling task. The other arms are created recursively by using a loop and are then joined to the pulley face and the hub. The driver pulley is made from a cylinder. The holes for both the driver and the driven pulleys are made by calling the feature "shaft" and cutting both the hubs of the pulleys with it. The "belt" is also called from the feature library.

2.1.5 *Springs*

This simple program of making compression or extension helical springs (Figure 2(c)) is very similar to the program that is used to make the threads for the bolts. The user has to input the diameter of the spring, the cross-sectional diameter, the spring length and the pitch of the spring.

First the workplane is set to the xy- plane (in the I-DEAS construction geometry) and a circle with the specified diameter is created at a distance equal to the radius of the spring from the y-axis. A profile is created out of this wireframe by using the I-DEAS's 'Autochain Profile' command and it is stored. It is then called in the object modeling and is revolved around the y-axis. The number of turns in the spring is calculated by dividing the length with the pitch of the spring. The revolution angle around the y-axis is then equal to (360 * number of turns). This program only creates springs with circular cross-sections. However other cross-sections can also be created.

2.2 **Development of Micro Features**

All the micro features except "elbow" were made using the IDEAS's feature definition task and can be found in the feature library. Some of the general construction techniques which were used to make all these features are described below.

The object to be used in the feature definition task could be made either in the object modeling task using the primitives or by creating a constrained profile and then extruding, revolving or skinning it in the object modeling. It is only necessary to draw a rough sketch of the profile used in feature task than drawing it accurately with all the correct dimensions. This rough sketch can be corrected by constraining the profile (parallel, perpendicular, horizontal ground, etc.) and the dimensions can be modified to the actual ones using the modify command. There are a few guidelines to be followed when constraining a profile. At least one point and the orientation of a line are needed to be grounded using the ground constraints. A profile does not have to be fully constrained to create an object; however, it is recommended to do so. A profile can be manually constrained or auto constrained. The profile after extruded in the object modeling, has to be stored to be used in the feature definition task.

In feature definition the 'create from object' option is chosen first and the object that was stored in the object modeling is selected. It would then ask if this feature is to be cut or joined with another object. Appropriate parameters have to be created from the dimensions to control the feature. There are two options to parametrize this object: one is a prompt and the other is an equation. A prompt is used so that the user can input the dimensions of the object to create the feature. An equation on the other hand, controls a dimensional constraint as a function of the other dimensions. When the prompt command is chosen, the program asks for the parameter name, the prompt, the entity to control, the feature label, and a default value. If the feature doesn't work because of over or under constraining, the profile has to be modified and a modified object has to be made out of that modified profile. It needs to be stored and the prompted and equational parameters

require to be created all over again for that modified object. The feature created can then be used in the object modeling using 'create-use feature' or 'construction-use feature' option. Using the 'create-use feature' command, only the feature is created from the prompted parameters; while using the 'construction-use feature' command, one can create the feature as well as use the feature for orienting or positioning with respect to other object or, for cutting or joining the feature with other objects. A brief description of some of the micro features is attached in the following subsections.

2.2.1 *Seam*

This feature can produce seams (Figure 4(a)) of variable length, thickness, width and radius of bend. The profile is created first using profile from points. The profile is then extruded in object modeling and stored as an object. In feature definition this object is called and prompted parameters (such as the length of the two places, the distance between them, the thickness and the width) are created for the feature definition.

2.2.2 *Counterbore and Countersink*

These are the two features that weren't made out of profiles. The counterbore is made in the object modeling using two cylinders. One of the cylinders is made shorter and wider than the other one. Then these two are oriented and joined.

In the feature definition, the entity to control now becomes the two cylinders instead of the profiles in previous cases. The radius and the height of this counterbore can be controlled by the radii and the heights of the two cylinders.

The countersink (Figure 4(b)) is also made similarly except that a cone is attached with the cylinder. The upper radius of the cone and the radius of the cylinder controls the radius of the countersink and the bore respectively. One equational parameter is created for the bottom radius of the cone. It is set to be equal to the radius of the cylinder.

Other than the above five micro features, we have also created features like wedge,rib, flange, round-end slot, parallelogram, semicircular plate, wireedge, offset plate, etc. and these features have been stored in the feature library.

2.3 **Development of the Functional Feature**

The only functional feature that we have made is the "twist" feature. This twist program can twist any object of any shape along the xy, yz or xz plane to a limit of 180 degrees (Figure 5). The only limitation of this program is that the user has to place the end of the object to be twisted at the origin and the rest of it has to lie in the negative z-direction. This is because the program is set to extract the cross-sectional profile from the object in the negative z-direction.

3. Creating Objects with Features: A Rapid Design Environment

The initial prototype system which has been already implemented, is intended to meet the requirements of our immediate use: geometry construction of a rapid design part (needed for the student design projects). Instead of using low-level entities (geometric entities), a part being designed with high-level entities (design features, i.e. macro, micro and functional features) which has a direct and significant meaning from the perspective of the designer (student), helps the rapid design process (especially in its conceptual design phase). The macro, micro or functional features that have been used in the prototype system imply features which achieve certain (explicit) functions, e.g. transmitting the motion (torque), binding for motion, joining (fastening) two components, etc. These design features support the students (the designers) in establishing an easy and meaningful communication with the design system to fulfill their creative intentions. The students construct apart geometry by sending proper messages to the particular feature class. For example, to create a block (in order to create the desk as shown in figures 6, one would send a message to the "create method" of the block feature class and instantiate its variational constraint variables to create an "instant" of the block feature. (Note: The block feature is already available in the I-DEAS as one of its primitive features. We did not create it specifically for the prototype system). After the generation of the feature, it is then positioned and oriented as required to create the desired part. In another example, we created a gearbox (the figure is not included) by using the macro and micro features such as spur gears, flanges, seam, shafts and semicircular plates. The gears are made by running the gear program and entering the dimensions of the pitch diameter, pitch, axle diameter and thickness, and height of teeth as the inputs. All the gears that are made have to be stored in directories because when other macro features are executed like the gear and bearing program, it erases all the objects in the workspace. The shaft is made by inputting its radius, its length and the size of the keyway. It is then oriented to the-axle of the gear. The casing of the gearbox is made out of a flange and a seam.

4. Conclusion

Engineering design related projects have been successfully integrated with all of our design courses. It helps combine creativity, technology and design methodology and eventually fosters the development of an

interdisciplinary attitude among the students. Since solid modeling is touted as the CAD/CAM tool of choice for the next century in industry, we strongly encourage the use of the solid modeling tool (SDRC's I-IDEAS) in the problem solving process. Though the students are truly enthusiastic about applying this tool in their works, proper and efficient use of the tool is difficult to achieve because of the inherent inflexibility associated with the tool itself. More difficult is to express the design intent at the conceptual stage of a design. The feature-based modeling approach provides the user with a natural way of expressing his/her design intent. In the SDRC's I-DEAS design environment, we have developed a prototype feature based design system to facilitate an easy use of domain-specific features and help students create their designs rapidly. Further work is necessary to develop more domain-specific feature libraries, and flexible assembly procedures to position and orient the features on the design parts correctly and quickly.

6. Bibliography

[1] Chung, J. C. H., Patel, D. R., Cook, R.L., and Simmons, M. K., "Feature-Based Modeling for Mechanical Design," Comput. & Graphics, Vol. 14, No. 2, PP. 189-199, 1990.

[2] LaWry, M.H., I-DEAS™ Student Guide, Structural Dynamics Research Corporation, 2000 Eastman Dr., Milford, OH 45150, 1993.

[3] Richards, R.A. and Sheppard, S.D., "Classifier System Based Structural Component Shape Improvement Utilizing I-DEAS," 92 ICCON Users' Conference Proceedings, I-DEAS™ and CAEDS® Cooperative Network, PP. 111-120, May 5-7, Cincinnati, Ohio, 1992.

[4] Roy, Utpal, "Design Project Experience in a CAD/CAM S ystem Course," Computers in Education Journal, Vol. IV, 'No. 1, PP. 6-10, January - March, 1994.


```
CM: ..*CREATES CYLINDER RACE*..

K :/CR T
K:BRAD+(4*THI/5),ORAD-(4*THI/5),WID/1.5
K :/OR R
K :D
K :90,0,0
K :/MA STO
K :CUT
K :/CO CU
K :L
K : WORK2
K :/MA G ?
K :CUT
K :/CO CU
K :L
K : WORK1
K : #GOTO MIDDLE

CM:***CREATES SALL RACE"**

K :# BALL:/TA CG CR C

K :R
K :K
K :BRAD+4*THI/5+SPC/2,0
K :SPC/2
K :D
K :/CR AC
K :*
K :/MA PR STO
K : PR01
K :/TA O CR REV
K : PR01
K :360
K :/OR R
K :D
K :90,0,0
K :/MA STO
K:BALL BACE
K:/CO CU
K :L
K: WORK1
K :/MA G
K:BALL RACE
K :/CO CU
K :L
K :WORK2
```

Figure 1. Part of a program File

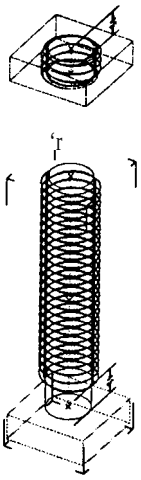

Figure 2(a). A Typical Regular Bolt With Square Head and a Square Nut

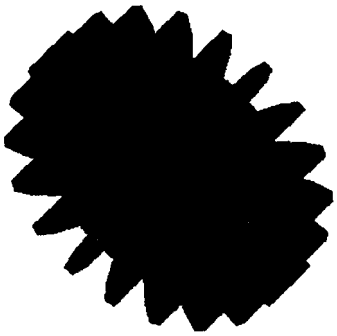

Figure 2(b). A Typical Spur Gear

Figure 2(c) .A Typical Helical Spring

Figure 2. Some Macro Features

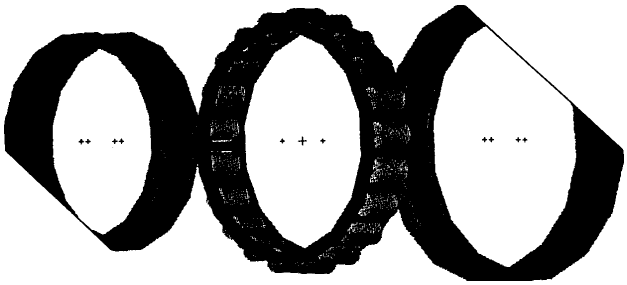

Figure 3(a). A Typical Roller Bearing With Cylinder Roller

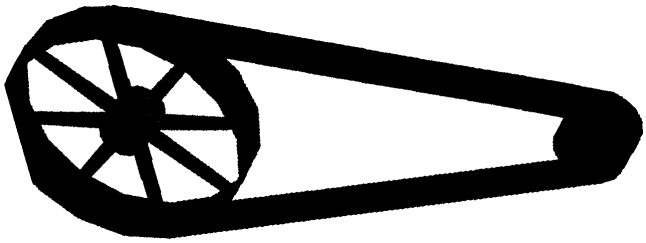

Figure 3(b). A Typical Pulley With Arms

Figure 3. Some Complex Macro Features

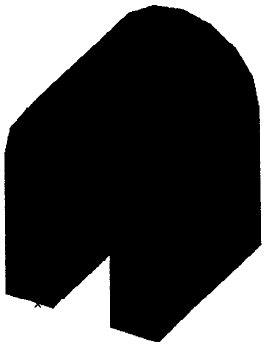

Figure 4(a). A Typical Seam

Figure 4(b). A Typical Countersink

Figure 4. Some Typical Micro Features

Figure 5. The Functional Feature, Twist

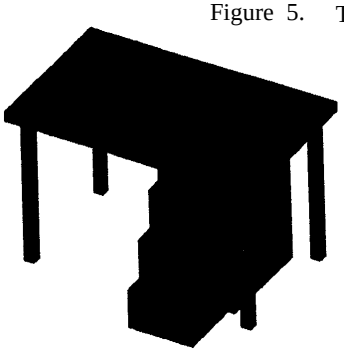

Figure 6. A Typical Office Table

1996 ASEE Annual Conference Proceedings