3-D Constraint-based Modeling: Finding Common Themes

Eric N. Wiebe
North Carolina State University

ABSTRACT
The move of constraint-based 3-D modeling into the mainstream of engineering design and manufacturing has been coupled with an increase of publishing activity in 3-D modeling instructional texts. Unfortunately, there is a lack of a clear overarching framework for teachers to understand what are the common themes which tie all of these different modeling tools together. This paper uses Shneiderman's (1998) Object-Action Interface model, along with the engineering design process, as a framework for understanding software interface elements which are common across four popular 3-D modeling software tools. The goal is to provide an educational framework from which instructional materials can be developed, independent of any one software tool, but which still address the fundamental functionalities of these new, powerful tools.

Introduction

Three dimensional modeling, especially constraint-based modeling, has broken into mainstream instruction in the past couple of years. In as much as book publishing is an indicator of instructional activity, in the past two years a number of reference and tutorial texts have been published about constraint-based modeling systems such as Pro/ENGINEER (Pro/E) and Mechanical Desktop. In the research arena, much of the interest to date has been the relationship of 3D instruction and visualization ability (c.f., Gorska, Sorby & Leopold, 1997; Leach & Matthews, 1992; McWhorter & et al., 1990; Ross & Aukstakalnis, 1993; Shah, 1994; Sorby & Baartmans, 1994; Wiebe, 1993). Another important issue is a general knowledge of how 3D constraint-based modeling software functions. In addition to a general ability to visualize 3D form is the ability to transform that mental form into a usable electronic geometric database. Particularly useful is a higher level understanding of this class of software, transcending any one particular software package. Just as with visualization skills, these software skills can be taken into the professional sector and applied to whatever modeling software is being used.

A model for instructional design

The goal of this current line of research is to look at the best way to impart a robust, general understanding of 3D constraint-based modeling. Related to this goal is facilitating the teaching of the specific software being used in an engineering/technical graphics class. Stepping back from simply looking at the computer interface, understanding how computer software will be used requires an understanding of the knowledge the user currently possesses and the tasks which they are attempting to complete using the computer software. The tasks the user brings to the computer can be divided between 'objects' and 'actions' (*Figure 1*). Objects may be pieces of information or physical entities representing how the task might be achieved if it were done exclusively apart from the computer. Actions represent how the objects will be manipulated. These objects and actions associated with the task have a parallel in the interface of the computer software. In this case, the objects are

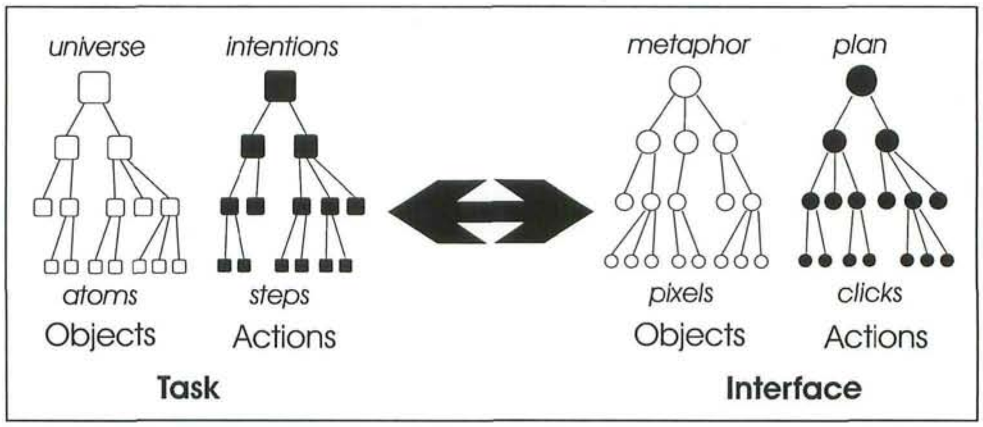


Figure 1 *- Object-Action Interface model (after Shneiderman, 1998).*

visual entities such as buttons, menus, text, and cursors, which make up the interface. Objects on the interface can be manipulated through actions taken with input devices such as a mouse or keyboard. The objects and actions of the interface should serve as metaphors for the objects and actions of the actual task which need to be accomplished. The degree to which the mapping between the task and the interface can be successfully bridged is a measure of how effectively the software can and is being used. The model that has been described is Shneiderman's Object-Action Interface (OAI) Model (Shneiderman, 1998).

Taking 2-D CAD software as an example, the tasks to be performed have many parallels with techniques executed with traditional drafting instruments. Understanding how well students can transition from traditional drafting to 2-D CAD packages can be explained, in part, by how directly techniques such as geometric construction and view layout can be translated to the commands and visual representations provided by the CAD software. Unfortunately, students learning 3-D modeling packages do not necessarily have real world experience in building 3-D models from wood, clay, plastic, or metal. Even if they did, there is some question as to how well these model building tasks translate to the interface metaphors used in modern 3-D modeling software.
Is there then, any use in the OAI model in understanding the use of 3-D modeling software? Another approach to understanding the object-action components of the 'tasks' students bring to the software is to think of the tasks as being more conceptually formed tasks rather than ones rooted exclusively in physical objects and actions. Rather than thinking in terms of using a real compass to strike an arc, a conceptual understanding of how a line (the object) can be swept out in a circular path (the action) by constraining it to be a fixed distance from a point. In this context, tasks that the students bring to the software (and their understanding of these tasks) will be based on the instructional materials presented by the teacher in addition to previous experiences.

Related to the OAI model is a more general notion of the 'mental model' the user has of the software tool (Carroll & Olson, 1990; Kuhn & Egenhofer, 1991; Norman, 1987). One's mental model can be considered semantic knowledge about a system - knowledge beyond the memorization of commands. Combined with syntactic knowledge - specific knowledge of a software interface - a user is able to predict how the system will respond to command inputs. A user can also formulate strategies of how to approach problem-solving using a particular system. Successful use of metaphors and concepts in instruction which reinforce a correct mental model of the system will enhance a student's ability to grasp the intricacies of the system (Norman, 1987; van der Veer & Wijk, 1990)

In summary, the OAI model and the application of mental models to human-computer interaction predict that arming students with strategies for model building which closely parallel the interface of 3-D modeling software should enhance the students' abilities to use the software to complete modeling tasks. The question then is, can a generic set of instructional strategies be developed which support instruction on a wide variety of 3-D constraint-based modeling software? This paper will begin exploring this issue by first looking at the feasibility of defining a set of generic objects and actions universally used in popular modeling software packages. From this understanding, instructional strategies can be developed which support the use and understanding of these interface elements.

Understanding the design process

The beginning point for developing a set of generic interface objects and actions is to understand the software's context in the mechanical design process (e.g., Wiebe, 1997; Wiebe, Howe, Summey & Norton, 1997). Implicit in this is an understanding that there are limitations as to how much of the design process is generic to most mechanical design environments and how much of this process is typically covered in engineering and technical curriculums. Looking at this process, most final products involve multiple discrete parts that must be coordinated in an assembly. It is also important to recognize that there are activities, which do not directly relate to the design process but are important for the management/use of the system. Examples of these auxiliary activities include: file management, correction of mistakes, and changes in modeling strategies.

At the broadest level, the mechanical design process can be thought of consisting of three phases (Bertoline, Wiebe, Miller & Mohler, 1997):

- Ideation
- Refinement
- Implementation

These phases happen both concurrently and cyclically towards a final design solution, and 3-D modeling software plays a role in all three phases.

In the ideation phase, design requirements are embodied in potential geometric forms and material specifications. In some cases, the design is a derivation of an existing design that may already exist as a computer model. In other cases, the design is created from scratch. The geometric form, which represents the design goals, can be decomposed into individual or groups of features. It is these features, representing the 'functionality' of the design, which need to be embodied in individual parts and assemblies. Here we can think of functionality as representing not only the design's function as a finished product, but also how each feature might map to the functional processes applied by the machines used to manufacture the part(s). There is also a need for geometric relationships to be established within and between the features in order to bring them together to become a whole design. These geometric relationships, along with algebraic relations, are meant to embody the design intent of the model.

Successful creation of the model requires the designer to translate the geometric features of the final, physical model into geometric features that can be created in the modeling software. Though the final, virtual computer model of the design may look and behave much like the proposed physical design, its construction probably differs from how the physical design is fabricated. It follows that the strategy developed for modeling the design depends on the user's semantic knowledge of what tools are available in the modeling software for creating and manipulating geometry. In terms of the OAI model, this strategy marks a critical mapping of the user's task (build a virtual model of this design) onto the software interface (use these commands in this sequence to build the model). Since the user is using off-the-shelf software, not designing it, the interest is less in proposing new interface elements as it is in how best to let the user/student know what is available and how best the tool can be used.

In the refinement stage, the modeling strategy developed in the ideation stage is applied to the actual construction of the model. The transition now has to be made from general semantic knowledge of software capability to specific syntactic knowledge of software commands and interface elements. Through a series of actions, geometric features are created and related to other features, both within and between parts in an assembly. This process entails both bottom up and top down strategies. In the former, individual parts are modeled from features and brought together into assemblies. In the latter, an overall assembly strategy is defined and parts modeled and assembled to meet this design. Through iterative analysis, decisions are made as to how the model should be modified to meet specific design goals. These analyses range from informal methods such as visual inspection of the fit of parts in an assembly to more rigorous methods such as finite elemental analysis. Changes to the model are also made in the way the geometry behaves to modifications in size and location of features. If it is determined that the design intent has not been properly embedded in the relationship of features, then these alterations of underlying relationships are made at this time.

In the implementation stage, the geometric database, representing the design, is transformed in ways that help support the manufacture, sale, and support of the product. These transformations might include the creation of traditional working drawings from the model or the creation of CNC code for machining molds. Similarly, technical illustrations can be created from projections of the model for use in brochures or in service manuals.

Of most interest in this paper is the activity taking place during the ideation and refinement stages. More specifically, planning and then creation of individual part models. This decision is not meant to diminish the importance of the other ways in which the modeler is used in the mechanical design process. Rather, part modeling represents what is probably the most common use of modeling software in the academic setting (c.f., Barr & Juricic, 1992; Howell, 1995; Clark & Scales, 1998). With that said, a students' understanding of the part modeling process will have considerable impact on their application of the modeling software in other areas of the design process. For example, the initial modeling strategy developed in the ideation phase will determine how the model is manipulated during iterative analysis.

Defining generic interface objects and actions

A deeper understanding of the common interface objects and the actions taken with them - as defined by the most popular constraint-based modeling packages - will provide a means for helping students develop effective model planning and construction techniques. It is worth noting that an exploration of generic modeling tasks has previously be done at a higher level and with an older generation of modeling software by a number of researchers (c.f., Barr & Juricic, 1992; Bertoline, Wiebe, Miller & Mohler, 1997; Howell, 1995). The goal here is to both use the most current generation of software available and to approach the definition of the tasks in a more systematic and detailed manner. The OAI model gives flexibility to the task definition by recognizing the hierarchical nature of most tasks and the similarly hierarchical nature of many software interface elements. There are, however, practical limits regarding the extent to which the full depth of the hierarchy can be addressed in this paper.

Initially, a checklist of generic objects and actions represented in modeling interfaces was developed based on personal experience and a selection of software-specific texts currently available:

- Pro/ENGINEER (Bolluyt, 1998; Toogood, 1998; Utz, Cox & Steffen, 1997)
- Mechanical Desktop (Howell, 1998)

This checklist was used while building a simple assembly *(Figure 2)* with a range of modeling packages:

- Mechanical Desktop
- Pro/ENGINEER
- SolidEdge
- SolidWorks

The next step was to evaluate how these software packages varied in their implementation of the initially defined interface objects and actions. Questions were asked, such as:

- Are objects visually or metaphorically represented differently between packages?
- Are actions missing (e.g., automated by the software) or organized differently between packages?

From the initial checklist, a new set of common interface object/action elements were developed which apply to all of these software packages and represent real-world application of the software.

Ideation - Modeling Strategy

As stated previously, the model building strategy should ideally be conceived during the ideation stage and prior to the actual construction of the model. More so than working with 2-D CAD systems, careful planning is central to the construction of all but the simplest parts in a 3-D constraint-based modeler. Planning requires knowledge of the basic methods used by the modeler to generate and constrain feature geometry. For most modelers, the primary method of creating feature geometry is to *(see Figure 3)*:

- Define a sketch plane in 3-D space
- Sketch a 2-D profile on the sketch plane
- Dimension/constrain the profile to other

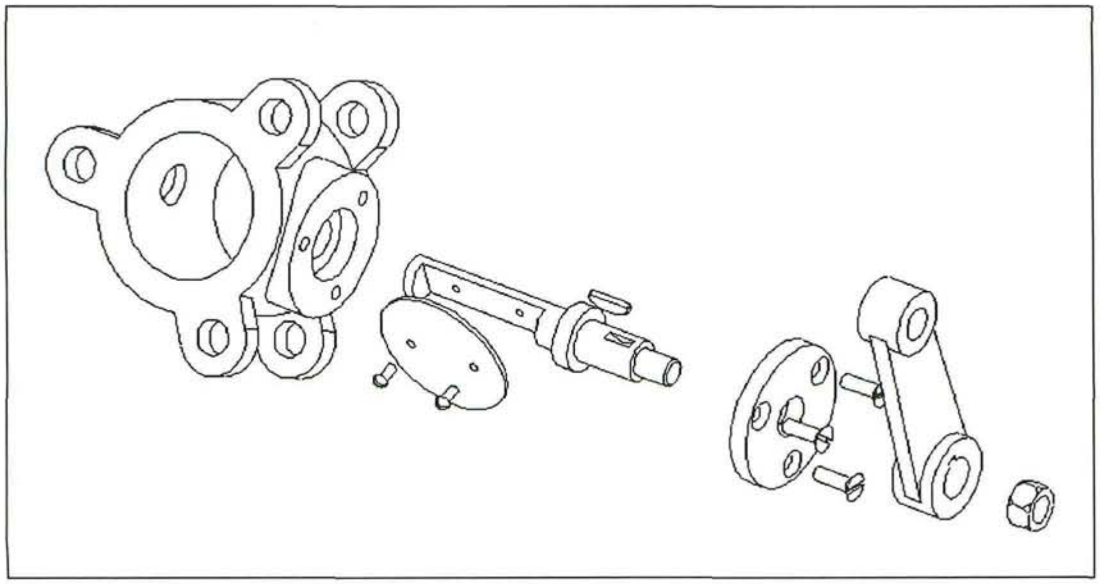


Figure 2 *- Test assembly.*

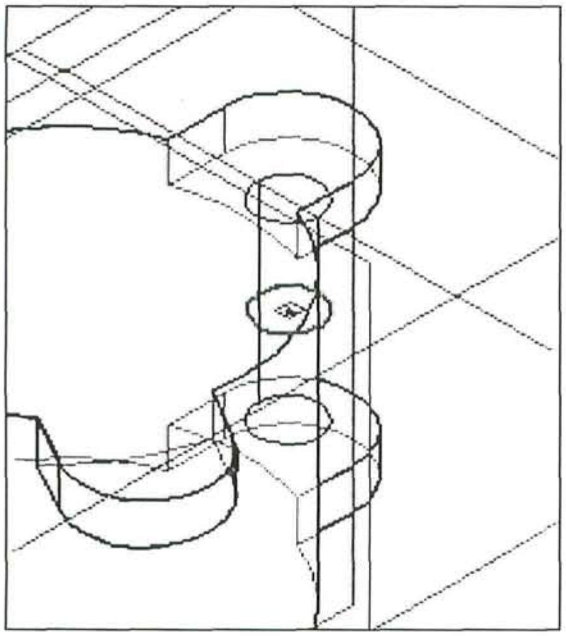


Figure 3 - Generalized sweep operation (SolidEdge).

construction or part geometry

- Define how the profile is swept away from the sketch plane to define a 3-D solid form
- Define the Boolean relation to the existing part geometry

Strategies that might be considered by the user include:

- Can/should multiple features be contained within a single sweep operation?
- Can/should a single feature be defined by multiple sweep operations?
- What is the appropriate sequence for the sweeping operations? For example, should all of the positive operations adding material be done first before subtractive features?
- Is there feature geometry that can be reused through mirroring or copying operations?
- Are there lines of symmetry that can be aligned along construction (datum) planes?

All of the above strategies are largely confined within a part. As part of a top-down design process, one may also be concerned with how the part features and construction geometry will interact with other parts in an assembly. For example, one may want to define features about construction planes that can be aligned between parts. Similarly, one may also want to try to have as many sweeps as possible mimic the actual machining operations being used to manufacture the part.

Through an iterative cycle of progressively more difficult lab exercises, students can explore the capabilities of these generalized sweeping operations and develop metaknowledge about the software capabilities. This knowledge can then be used to deconstruct existing geometric representations of parts into a series of sweeping operations. It is not unreasonable to expect students to sketch their plan of action on grid paper prior to actual execution of the model on the modeling system.

Refinement

Profile sketch process

With a basic strategy in place, the modeling being done in the refinement stage can commence. Since all geometry must be anchored relative to some coordinate system, some decision has to be made as to how this will be achieved. The approach is determined by a combination of system functionality and strategies determined in the ideation stage. A common approach is to define three mutually perpendicular construction (datum) planes intersecting at a global coordinate system origin *(as seen in Figure 4)*. For modelers such as Pro/E, SolidWorks, and SolidEdge, these planes can be created by default when a new part is created.

The first sweep operation creates what can be considered the 'base feature' of the part. The first step is to choose one of the construction planes in order to define an orientation and location of the profile used to create the first feature. Once a plane is chosen, an X-Y coordinate system is oriented on the

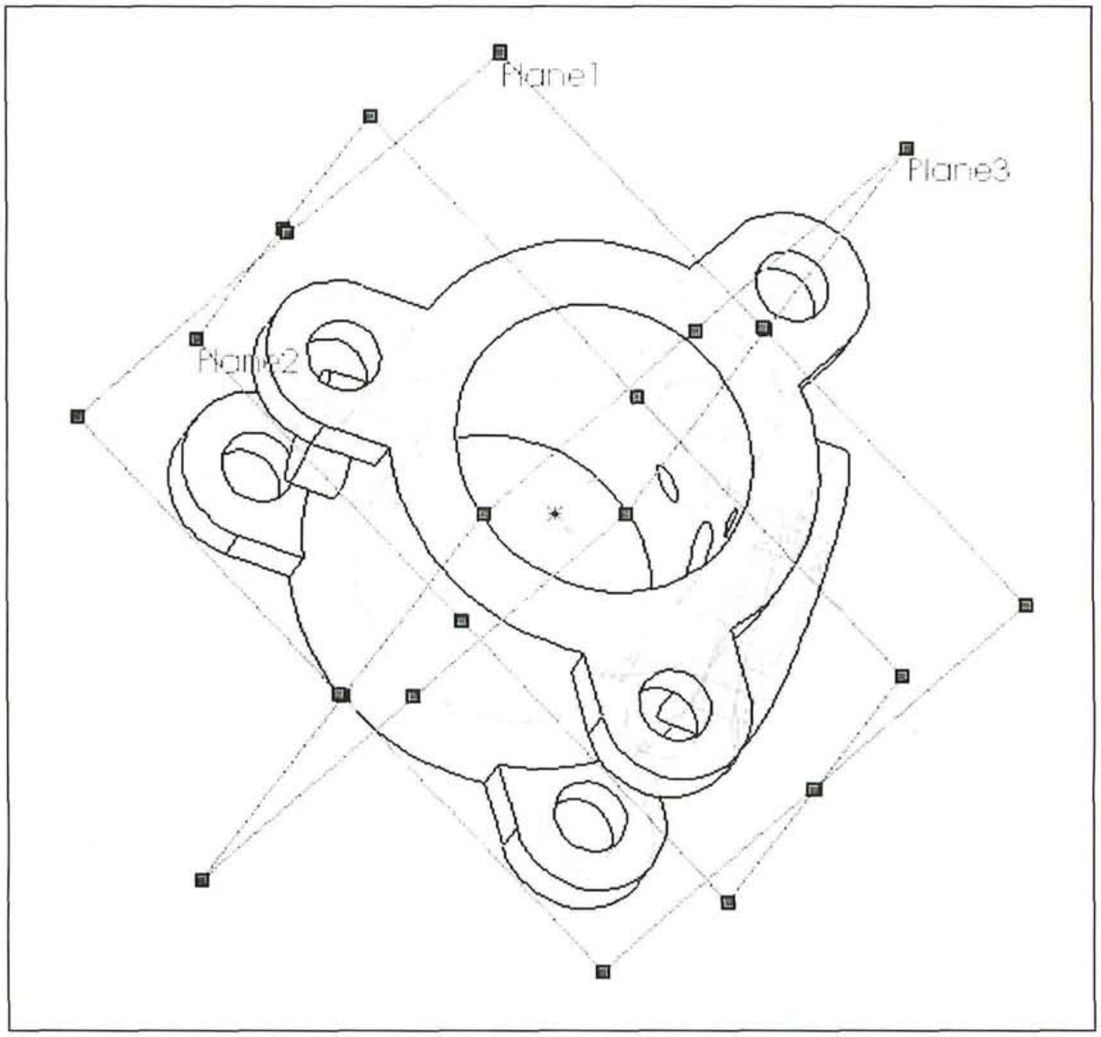


Figure 4 *- Three mutually perpendicular construction planes used to define the initial geometry of a part (SolidWorks).*

plane in a number of ways: 1) by systematically rotating/flipping a set of coordinate axes (Mechanical Desktop), 2) based on an established local coordinate system on the plane (SolidEdge), or 3) choosing a mutually perpendicular plane and indicating its X-Y orientation (Pro/E). Because it is the first feature created in the modeler, it is not formally a Boolean operation and therefore simply adds geometry to the void. This feature is often used as a basis for orienting and locating new features.

A key technique needed in using modelers is the sketching of the 2-D profile used in all sweeping operations. Though the profile definition can be automated for simple geometries (e.g., a circle for creating a swept cylinder), the user has to sketch and create constraints for most profiles. After a sketch plane is defined, 2-D drawing tools are used to create a profile. These tools consist of both creation tools, such as line and arc tools, and editing tools, such as trim/extend and copy. The sketching takes place in either a pictorial or orthogonal view. Some systems default to an orthogonal view (e.g., Pro/E) while others hold the existing, usually pictorial, view (e.g., SolidWorks). The profile sketch consists of as little as a single line or as complex as multiple closed loops of straight edges, circular curves, and spline curves *(Figure 5)*. The requirements of the profile depend on the modeler one uses. Some require single closed loops while others allow open loops or multiple loops. Typically, loops cannot overlap and, if there is more than one loop, none can be open.

The loop that is drawn, combined with the sweep path, defines the topology of the eventual solid feature. The geometry of the solid feature is defined through a combination of explicit and implicit geometric constraints. Within the profile, implicit constraints are relations of geometric elements to each other. Examples include:

- Parallelism
- Perpendicularity
- Collinearity
- Similar size/length
- Symmetry
- Closure

How implicit constraints are implemented depends on the modeler. With most modelers, there is a set (or customizable) range of

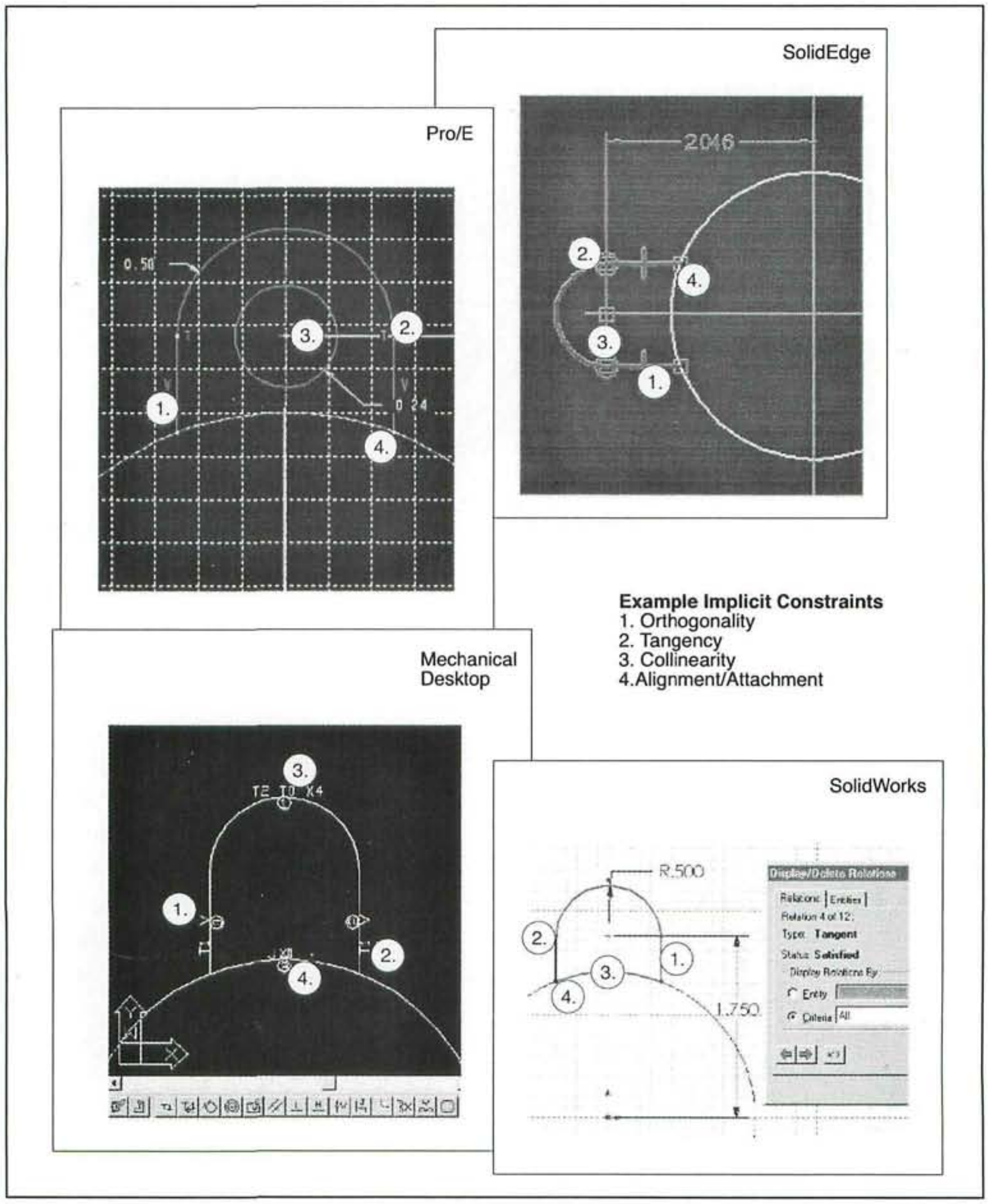


Figure 5 *- Example profile sketch interfaces.*

Parameter	Notes
Boolean operation to apply	Union, Subtraction, or Intersection
Distance of the sweep	Can be set as a scalar unit or defined rela tive to other geometry (e.g., through next surface)
What type of sweep path	Linear, circular, a defined curve, or a curve connecting a series of profiles
Direction of the sweep	Linear or circular sweeps can come one direction or from both sides of a profile

Table 1 *- sweeping parameters.*

size or orientation variation in which a profile element will be considered: the same size as another element, horizontal, connected end to end with another element, etc. If the element is within range (e.g., within 5 degrees of vertical), the appropriate constraint is applied. With some modelers, this application of constraints happens 'on-the-fly', dynamically cleaning up the sketch as one draws. With other modelers, the user finishes the sketch and then instructs the modeler to apply appropriate constraints. Most modelers will provide visible icons indicating which implicit constraints were applied and with what other elements *(see Figure 5).*

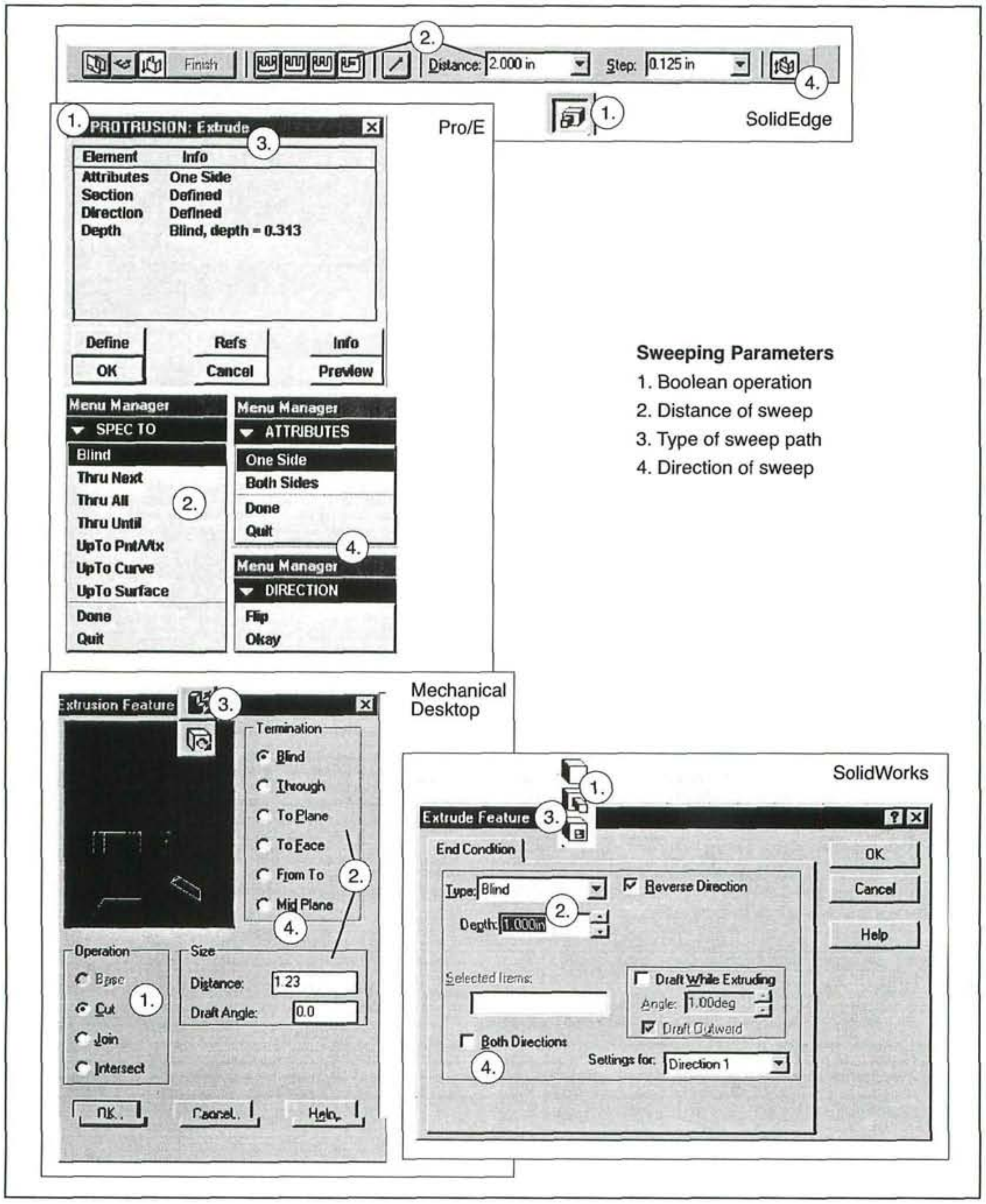
Explicit constraints are typically the size and location of geometric elements relative to other elements in the profile or to existing geometry. How these explicit constraints are established depends on the modeler. In some, such as Pro/E, visible size and location dimensions are placed to fully define (in combination with the implicit constraints) the geometry. In other modelers, size and location are determined exclusively by the act of sketching the profile geometry -- no other dimensioning is needed. One can, however, add dimensions to provide for control of the geometry. This difference is often used as the defining element between parametric and variational constraint-based modelers.

Sweeping operations

The completed profile is combined with sweeping parameters to create the solid feature. The common sweeping parameters are outlined in *Table 1.*

Where and how these parameters are defined depends on the modeler *(Figure 6).* In some cases, parameters are defined before the profile sketching (the Boolean operation frequently is). In other cases, all of the parameters can be defined after the profile sketch has been created (e.g., SolidWorks). It is also worth noting that the specific language used to define the parameters varies from package to package. For example, none of the packages refer to the Boolean operation by its formal name. Instead, the Booleans are packaged in 'feature-based' commands such as protrusion, cut, and hole. With some modelers, the feature defines both the Boolean and other sweeping parameters. For example, Pro/E uses the Shaft feature to define a Boolean Union created with a revolved sweep.

There are, of course, exceptions to this generalized sweep description of feature creation. In some cases, parameter definition is automated: hole features may always use a linear sweep. In other cases, the feature definition bears essentially no resemblance to profile sweeping (e.g., Chamfer and Round feature commands typically only require the



Distance: 2.000 in Step: 0.125 in

Element	Info
Attributes	One Side
Section	Defined
Direction	Defined
Depth	Blind, depth = 0.313

Distance: 1.23
Draft Angle: 0.0

Type: Blind
Depth: 1.000in
Angle: 1.00deg
Settings for: Direction 1

Sweeping Parameters
1. Boolean operation
2. Distance of sweep
3. Type of sweep path
4. Direction of sweep

Figure 6 *- Generalized sweep interface examples.*

user to select edges on the model to operate on and then specify dimensional constraints).

Construction geometry creation

Construction geometry is generally defined as geometry used in support of the definition of the solid model geometry. Construction geometry exists in the model database but does not explicitly represent 'visible' components of the model. This geometry is typically created as geometry in 2-D space used in support of profile sketching or as geometry in 3-D space used in support of profile sweeping, feature copying, or other feature operations. The most common 2-D geometry created in support of profile construction is center lines used to indicate symmetry.

Geometry	Example Definition	Example Use
0-D Point	Intersection of a construction axis with a construction plane	Define the limiting extent of a linear sweep as a point along a construction axis
1-D Line	Intersection of two construction planes	Define an axis of revolution for a revolved sweep
2-D Plane	Parallel with a construction plane and tangent to a curved face	Define a sketching plane for a profile
3-D Cylinder	Collinear with a construction axis at a set radius	Define a series of curves (via intersections with planes) for radial patterning

***Table 2** - Examples of construction geometry in 3-D space, how they might be defined, and how they might be used.*

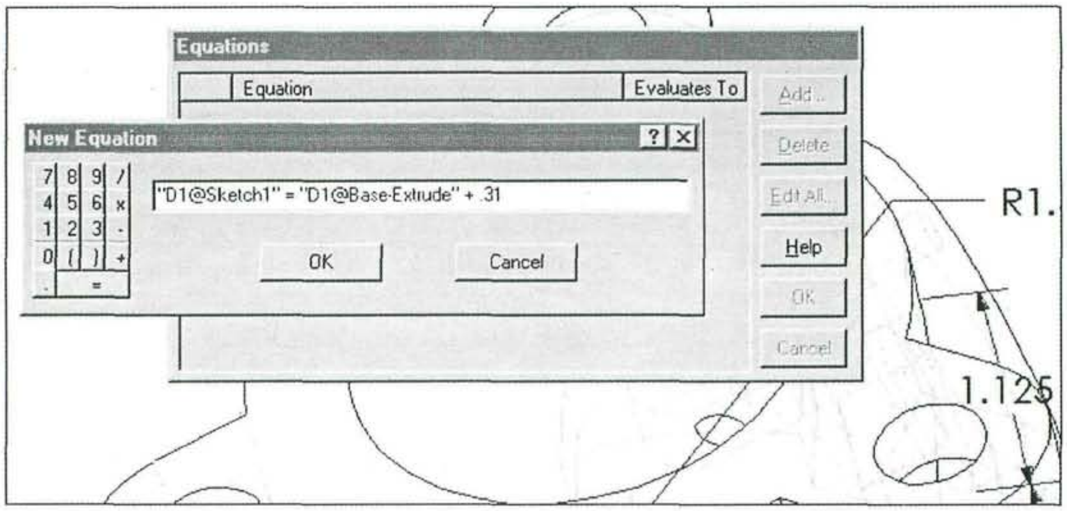
Two-dimensional center lines can also be used either in the sketch plane or 3-D space to define an axis of revolution for revolved sweeps. Definition of geometry in 3-D space, in particular, demands a basic knowledge of descriptive geometry and how existing geometry can be used to define new geometry. *Table 2* gives examples of construction geometry.

Duplication of features

Once a feature is created, it can be propagated in a number of ways. Most systems support some type of patterning. Linear patterning can be defined as a 1-D or 2-D array of copies of a feature. Both the number of copies in each orthogonal dimension and the spacing between copies needs to be defined. With a radial pattern, an axis of rotation is defined, along with a radius, the number of copies, and the angular displacement between copies (either as an angle or distribution about 360 degrees). These patterns are not unlike patterned arrays in 2-D CAD systems, except they can be defined on any 3-D plane in space and are used to propagate 3-D features. In addition to patterning, most systems also support creating a singular copy, either mirrored about a plane of symmetry or translated some offset from the original. With all types of duplications, varying levels of dependencies can be established between the original and the copies. These dependencies often include independence or dependence of the copy to the original's sweep and profile parameters. For example, dependency can be established such that a change in the original hole's diameter induces a similar diameter change in all of the copies.

Redefinition of the model

Using various interface mechanisms, all of the modelers allow for extensive redefinition of features. The modelers provide fairly direct mechanisms for modification of data values assigned to constraint dimensions. These changes are typically applied to provide for fairly localized changes to individual features, but can also involve topological change throughout the model. Various error-checking mechanisms warn of the creation of illegal geometry, but usually not before constraint values are modified and the model attempts the reconfiguration. With all of the modelers, constraint values can be indirectly modified by linking their values to the values of other constraints *(Figure 7)*. Using



"D1@Sketch1" = "D1@Base-Extrude" + .31

***Figure 7** - A user defined relational constraint (SolidWorks).*

algebraic equations and dimensional constraints as variables, relationships are defined between constraints. Typically there is a hierarchy where only a single constraint variable is allowed on the left hand side of the equation and the constraint variable(s) on the right would then 'drive' the left side constraint. In *Figure 7*, Dimension 1 (D1) of the feature Base-Extrude drives Dimension 1 of (profile) Sketch 1.

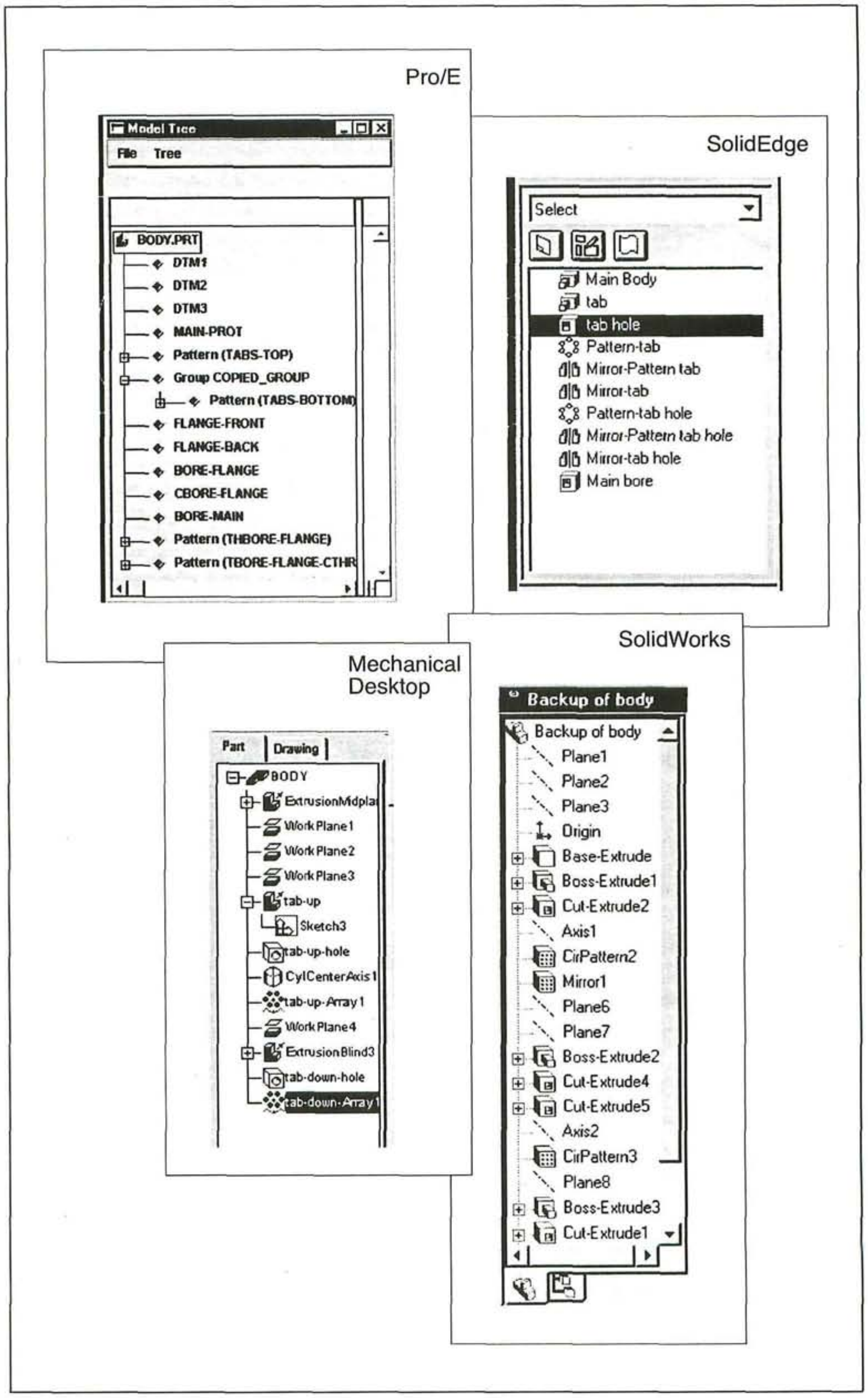
In addition to modifying dimensional constraint values, the parameters defining the sweep or the geometry of the sweep profile can also be changed. Typically, a redefinition command allows one to choose the feature to redefine and then takes the user back through dialogues/menus that were used to define the feature in the first place. Again, error checking is used to assure the redefined feature does not violate geometry rules.

All of the modelers examined ordered the features in a history-based tree whereby all features act on the model based on what features precede it in the tree but ignore what proceeds it. This history basis has a number of implications. First, construction geometry used to define features must exist in the tree prior to the feature(s) which depend on it and cannot be deleted at a later time without also redefining the dependent feature. Next, the Boolean operations incorporated into features will only operate on geometry created prior to it. For example, a through all hole operation will not cut through a flange in the path of the sweep if the flange was created after the hole operation. Finally, based on the previous statement, the overall geometry of the part can be altered simply by reordering features in the tree (a capability shared by all of the modelers). *Figure 8* shows how these feature trees are represented in the modelers. Notice that icons are used to represent different types of elements in the tree (e.g., type of feature, the profile sketches used by features, parts versus assemblies) and how the tree branches.

View control

Integral to all model construction is viewpoint control. In addition to the controls found in 2-D CAD systems, pan and zoom, these modelers also allowed the 3-D viewpoint to be changed either to predefined orthogonal or pictorial viewpoints, user-defined viewpoints, or through free rotation. View controls are usually easily accessible and/or available through hot keys.

In addition to controlling point of view, the modelers also allow geometry to be represented in numerous ways. Typical options included:



Model Tree
File Tree
BODY.PRT
DTM1
DTM2
DTM3
MAIN-PROT
Pattern (TABS-TOP)
Group COPIED_GROUP
Pattern (TABS-BOTTOM)
FLANGE-FRONT
FLANGE-BACK
BORE-FLANGE
CBORE-FLANGE
BORE-MAIN
Pattern (THBORE-FLANGE)
Pattern (TBORE-FLANGE-CTHR

Select
Main Body
tab
tab hole
Pattern-tab
Mirror-Pattern tab
Mirror-tab
Pattern-tab hole
Mirror-Pattern tab hole
Mirror-tab hole
Main bore

Part Drawing
BODY
ExtrusionMidpla
WorkPlane1
WorkPlane2
WorkPlane3
tab-up
Sketch3
tab-up-hole
CylCenterAxis1
tab-up-Array1
WorkPlane4
ExtrusionBlind3
tab-down-hole
tab-down-Array1

Backup of body
Backup of body
Plane1
Plane2
Plane3
Origin
Base-Extrude
Boss-Extrude1
Cut-Extrude2
Axis1
CirPattern2
Mirror1
Plane6
Plane7
Boss-Extrude2
Cut-Extrude4
Cut-Extrude5
Axis2
CirPattern3
Plane8
Boss-Extrude3
Cut-Extrude1

***Figure 8** - Example feature tree interfaces.*

- Wireframe
- Shaded
- Hidden lines removed
- Construction geometry hidden or visible
- Hidden lines grayed/dashed

The common interface standard seems to be to have construction geometry, such as planes, appear in wireframe and not be affected by the rendering of the part model. This has both its advantages and disadvantages: it always makes construction geometry accessible (when set to visible), but also leads to orientation/location confusion on the part of the user.

Evaluation of the design

Going hand in hand with the creation of a (virtual) model in the modeling system is analyzing it to see if it meets design criteria. Though central to the use of modelers in industrial settings, this area will only be briefly outlined here. The modelers evaluated all have the capabilities of doing mass properties calculations on the parts. In addition, density values can be assigned to parts so that assemblies can also be appropriately evaluated. In addition to visual inspection of parts and assemblies, the modelers also have measuring tools to measure linear and angular distances between features in real or projected dimensions. Interference (overlap) between features is evaluated either visually or with Boolean operations. Usually separate from the base modeler package are finite element analysis, kinematic/dynamics, and rapid prototyping (for physical part analysis) tools. Depending on the level of integration, information is exchanged between the modeler and analysis tools either through the native file format of the modeler or through a neutral format such as IGES. Results of the analysis then inform the designer how the model might be modified to better meet the design specifications. After a round of model modifications, the analyses can then be repeated and, as necessary, the entire cycle repeated until the designer is satisfied with the results.

Implementation - Documentation

Once a design is finished, 3-D model information is converted into a form usable by the latter stages of the product realization process. Historically, paper documentation in the form of working drawings was central to the documentation of a design for manufacture. Paper documentation is created via the modeling system in much the same way it is with a 2-D CAD system, with some important differences. A typical process for producing virtual drawings, which can then be printed, might be:

- Establish a paper size for printing
- Create or retrieve a titleblock and border for the drawing
- Create views of the model within the titleblock
- Detail the views with appropriate dimensional, symbolic, and text information
- Store and print the document

This process usually takes place within a module separate from where the modeling is done, with a separate set of tools used for creating the drawing. This module has much of the look and feel of a 2-D CAD package, but usually has much less emphasis on tools used to create linework representing model geometry. This is because linework is largely created by capturing projections of the model. Using viewing tools similar or identical to the tools used to establish views within the modeling module, views of the model are laid out within the drawing border. For orthographic views, the user often establishes a 'base view' with other views projected based on their location relative to the base view. Scale is established for the base view and all associated views. View parameters can typically be established for each view to indicate how hidden edges and tangents should be represented.

Another important distinction from traditional 2-D CAD is that much of the dimensioning, per se, has already been done when dimensional constraints were associated with features within the model. In theory, the same constraints used to define model features should also be appropriate for defining the geometry in the working drawing. In reality, there will almost always be a divergence from how a model is constructed and how a part will actually be manufactured. For that reason, there are tools available to create additional dimensions (reference or otherwise) and to suppress existing dimensional constraints.

The close tie between view generation and dimensional detailing and the model is because a dynamic link is preserved between the working drawing and the 3-D model. In all of the packages evaluated, changes in the model are reflected in the drawings and vice versa. This bi-directional associativity is both a tremendously powerful tool for keeping drawings accurate and up to date, but also tends to limit the liberties one can take in detailing a drawing. Since the drawings are updated with no human intervention, drawing conventions which violate true projection or use subjective geometry representation are typically not allowed.

It is worth noting that other information contained within the model database can also be dynamically linked to the working drawings. Information, which might be used in a bill of materials or parts list, can be linked via variables within the drawing.

Even with the limitations imposed by the bi-directional linkage between the drawings and the model, there tends to be considerable flexibility in establishing the classic view types used in drawings, including:

- Sections
- Auxiliary
- Partial or broken
- Removed

Sectional views are typically established by defining a cutting plane - either as an existing construction plane or as an extruded edge - and then indicating the appropriate view to section and show a cutting plane line(s). Partial or broken views can be created by indicating a region that should be visible in a standard view. Auxiliary views simply have the user indicate a base view and an alternate edge (e.g., construction plane/axis or model edge) to act as a folding line to revolve a view about. Removed views can either be existing projected views moved from their standard locations, or new base views established independent of any other base view. These view representation capabilities are often combined. For example, a detail might be created by establishing a removed, partial, section view at a different scale. In addition to the flexibility of creating multiple view types, there is the ability to bring in multiple parts–either independent or as part of an assembly.

Conclusion

The results of this evaluation clearly show that there are significant commonalties between the modelers in the objects and actions used to define primary interface elements. It is also clear by reviewing the interface screen capture figures in this paper that the syntactic level surface details of the interfaces differ markedly between modeling systems. Still, at a semantic level, there were clear common themes that could be mapped between all of the modelers and used as the basis for instructional material independent of particular software packages. It is hard to extrapolate whether these commonalties will hold across all modelers on the market, but the four systems surveyed represent a significant percentage of the modelers currently being used in four and two-year engineering and technical graphics programs in the United States (Clark & Scales, 1999). Further work will needed to be done by this author or others to see whether these commonalties hold for other popular industrial modelers, such as I-DEAS, Unigraphics, and Catia.

By using the Object-Action Interface (OAI) model applied to the design process as an overall structuring methodology, the following interface elements were identified:

- Profile sketching

- Implicit and explicit profile constraints
- Sweeping operations
- Construction geometry creation
- Duplication of features
- Dynamic dimensional constraint modification
- User-definable dimensional constraint relations
- History-based feature trees
- View control
- Bi-directional links between model and detail drawings
- View creation and detailing options for detail drawings

These interface elements encompass much of what might be used in part creation, modification, and documentation; central activities in both academic and industrial use of 3-D constraint-based solid modeling. These activities in no way cover all of the activities performed with modeling systems in academic and industrial settings. For example, assembly modeling was not included in this evaluation. Still, this work validates the potential usability of the OAI model as an approach to develop instructional materials and methods to teaching modeling which spans the majority of software tools currently on the market.

References

Barr, R. E., & Juricic, D. (1992). A new look at the engineering design graphics process based on geometric modeling. *Engineering Design Graphics Journal, 56*(3), 18-26.

Bertoline, G. R., Wiebe, E. N., Miller, C., & Mohler, J. L. (1997). *Technical graphics communications*. (2 ed.). New York, NY: McGraw-Hill.

Bolluyt, J. E. (1998). *Design modeling with Pro/ENGINEER*. Shawnee-Mission, KS: Schroff Development Corp.

Carroll, J. M., & Olson, J. R. (1990). Mental models in human-computer interaction. In M. Hellander (Ed.), *Handbook of human-computer interaction*, (pp.45-65). Amsterdam: North-Holland.

Clark, A. C. & Scales, A. Y. (1999). *Taking the pulse of the profession*, Paper presented at the ASEE Engineering Design Graphics Division Mid-Year Meeting, Columbus, OH.

Gorska, R., Sorby, S., & Leopold, C. (1997). *Gender differences in visualization skills - An international perspective*. Paper presented at the ASEE Engineering Design Graphics Division Mid-Year Meeting, Madison, WI.

Howell, S. K. (1995). The use of a parametric feature based CAD system to teach introductory engineering graphics. *Engineering Design Graphics Journal, 59*(1), 27-33.

Howell, S. K. (1998). *Mechanical desktop: Parametric solid and assembly modeling*. Albany, NY: Autodesk Press.

Kuhn, W., & Egenhofer, M. J. (1991). CHI '90 workshop on visual interfaces to geometry. *SIGCHI Bulletin*, 23(2), 46-55.

Leach, J. A., & Matthews, R. A. (1992). Utilization of solid modeling in engineering graphics. *Engineering Design Graphics Journal*, 56(2), 5-10.

McWhorter, S. W., & et al. (1990). *Evaluation of 3-D display techniques for engineering design visualization*. Paper presented at the ASEE Engineering Design Graphics Division Mid-Year Meeting, Tempe, AZ.

Norman, D. A. (1987). Some observations on mental models. In R. M. Baecker & W. A. S. Buxton (Eds.), *Readings in human-computer interaction: A multidisciplinary approach*, (pp. 241-244). Los Altos, CA: Morgan Kaufmann.

Ross, W., & Aukstakalnis, S. (1993). *Virtual Reality: Implications for Research in Engineering Design Graphics*. Paper presented at the ASEE Engineering Design

Graphics Division Mid-Year Meeting, San Francisco, CA.

Shah, M. J. (1994). *The use of a parametric feature based CAD system to teach introductory engineering graphics*. Paper presented at the ASEE Engineering Design Graphics Division Mid-Year Meeting, Houston, TX.

Shneiderman, B. (1998). *Designing the user interface: Strategies for effective human-computer interaction*. (3rd ed.). Reading, MA: Addison-Wesley.

Sorby, S. A., & Baartmans, B. G. (1994). *An introduction to 3-D spatial visualization - A pre-graphics course*. Paper presented at the 6th International Conference on Engineering Computer Graphics and Descriptive Geometry, Tokyo, Japan.

Toogood, R. (1998). *Pro/ENGINEER tutorial: A click-by-click primer.* Shawnee-Mission, KS: Schroff Development Corp.

Utz, J., Cox, W. R., & Steffen, D. (1997). *Inside Pro/ENGINEER*. Santa Fe, NM: OnWord Press.

van der Veer, G., & Wijk, R. (1990). Teaching a spreadsheet application - visual-spatial metaphors in relation to spatial ability, and the effect on mental models. In P. Gorny & M. J. Tauber (Eds.), *Visualization in human-computer interaction*, (pp. 194-208). NY: Springer-Verlag.

Wiebe, E. N. (1993). Visualization of three-dimensional form: A discussion of theoretical models of internal representation. *Engineering Design Graphics Journal*, 57, 18-28.

Wiebe, E. N. (1997). Adding agility to CAD: Integrating product data management tools into an organization. *International Journal of Human Factors in Manufacturing,* 7, 21-35.

Wiebe, E. N., Howe, J. E., Summey, J., & Norton, J. J. (1997). Computing and organizational assessment in the furniture industry. In G. Salvendy, M. Smith, J., & R. J. Koubek (Eds.), *Design of computing systems: Social and ergonomic considerations*, (Vol. 21B, pp. 921-924). Amsterdam: Elsevier.