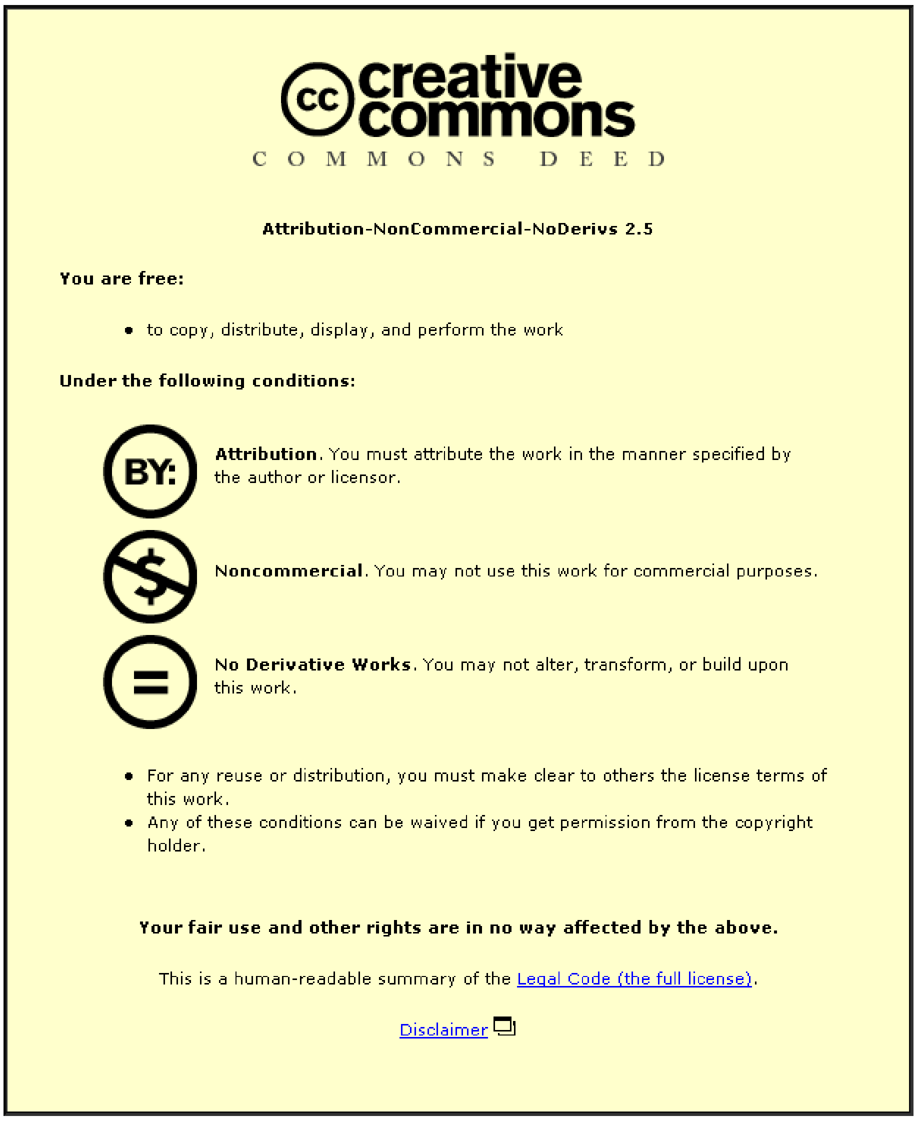

In A. Bramley, & A. Mileham (Eds.), ‘*Advances in Manufacturing Technology VII* ‘
the Proceedings of the Ninth National Conference on Manufacturing Research, NCMR 1993
(pp. 272-276). University of Bath, Bath, UK: Bath, UK: Bath University Press.

Feature Validation in a Feature-Based Design System

K Case, J.X. Gao, N Gindy

Dept of Manufacturing Eng, Loughborough University, Loughborough, Leics
LE11 3TU

1. Abstract

The Loughborough University of Technology Feature-Based Design System (LUT-FBDS) allows detail design to be carried out in a computer aided design (CAD) environment by the addition of form features to stock material or part-machined components. An iconic user interface assists in the description parts in terms of a set of primitive features such as holes, pockets and slots or higher level compound features such as patterns of holes and counterbored holes. This feature representation is generated in parallel with the geometric data structure of the underlying boundary representation solid modeller. The feature representation is useful for a range of downstream manufacturing activities, but our research focusses on the integration of CAD with process planning. LUT-FBDS functions allows the designer or manufacturing engineer to progressively construct the final geometry of a part, and facilities are provided for the designer to modify parameters which relate to feature dimensions, location, orientation and relationships with other features. These changes may result in changes to the feature representation and hence there is a need for feature validation to ensure the integrity of the model.

2.Introduction

Features can be described as groups of geometric entities with associated attributes related to product life-cycle activities, and general descriptions of the approach can be found in the literature (e.g. Case and Gao, 1993). LUT-FBDS (Loughborough University of Technology Feature Based Design System) is an example of the design by features approach (Gao et al, 1992} and provides the designer with a library of features with which to construct a component representation. It has been used for process planning and process capability modelling research (Gindy and Ratchev, 1992). Several issues are raised by a design by features approach but this paper is particularly concerned with problems of *Feature Validation* in an iterative and interactive detail design environment.

3.The Design by Features System

LUT-FBDS provides the designer with a library of primitive features which are used to construct a geometric model within a proprietary solid modeller and to generate a complementary feature data model containing information for process planning. Figure 1 shows a typical example that has been defined in this way. A full description of the feature taxonomy is given in Gindy (1989), but each feature class is classified by its number of External Access Directions (EAD's). The through slot shown in figure 2a has three EAD's, each of which (in this simple case) is perpendicular to the imaginary face created by removal of the feature volume from the component.

The concepts of entry face, exit face and a depth profile allow a generic description of the topological characteristics of the feature, and a realisable entity is produced by the addition of geometric shape and dimensionality to 'instance' the feature. Figure 3 shows the features available.

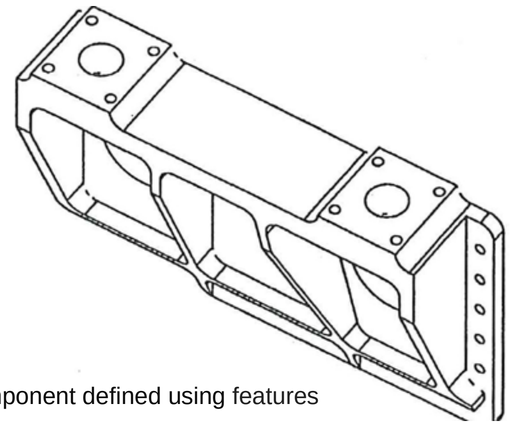

Figure 1. A typical component defined using features

Specification of a feature's class (and hence topology), shape, dimensions, location and orientation are sufficient to define the geometric model, but additional information is required for manufacture. The feature class may provide some indication of manufacturing method although this is more properly a process planning decision. Additional information is primarily in the form of relationships within and between features.

Ownership relationships such as that between the hole and slot in figure 2b are important for two reasons. During design such relationships are used to express and control design functionality so that for example the slot may be defined as the logical child of the hole so as to maintain the spatial relationship on moving the hole. At the conclusion of the design stage these relationships serve no further purpose, but further relationships that are useful to process planning may be determined. The hole now becomes the child of the slot to record a potential machining access direction and precedence. (As the hole is a through hole it will also be a child of the 'bottom' surface of the block).

Manufacturing aspects which need to be controlled to meet design functionality are traditional conveyed from design to process planning by the annotation of drawings with dimensional and geometric tolerances and attributes such as surface finish. LUT-FBDS provides for a similar 'annotation' of the feature model and in particular uses relationships to describe tolerances for dimensions (linear and angular), form (straightness, flatness, roundness and cylindricity), attitude (parallelism, squareness and angularity) and location (position, concentricity and symmetry).

4. Feature Validation

Design is an iterative process where the final geometric form is the result of extensive creation, deletion and modification of the geometry. Maintaining the integrity of the model during this process is one of the important characteristics of an effective solid modelling system. The purpose of feature validation is to handle feature information in an analogous way to avoid the possibility of generating ambiguities, inconsistencies and false information. The validation should accommodate the activities of feature (a) creation, (b) deletion, and modifications to (c) dimensions, (d) location, (e) orientation, (f) parent-child relationships, (g) dimensional and geometric tolerances, and (h) attributes. The circumstances to be handled consist of (a) changes to feature class, and violation of (b) parent-child relationships and (c) tolerances. The objectives are to monitor the outcome of design activities, detect the occurrence of changes to the feature model and update the feature model under designer or automatic control. Some examples will demonstrate the general principles. Dimensional changes may cause a feature class change by geometric interaction with other features or by exceeding the stock material. Thus increasing the width of the slot in figure 2b may cause it to become a step (figure 2d). Changes to feature class can be generated by changes to the positional parameters so that the slot of figure 2b becomes the step of figure 2c or a notch of figure 2f.

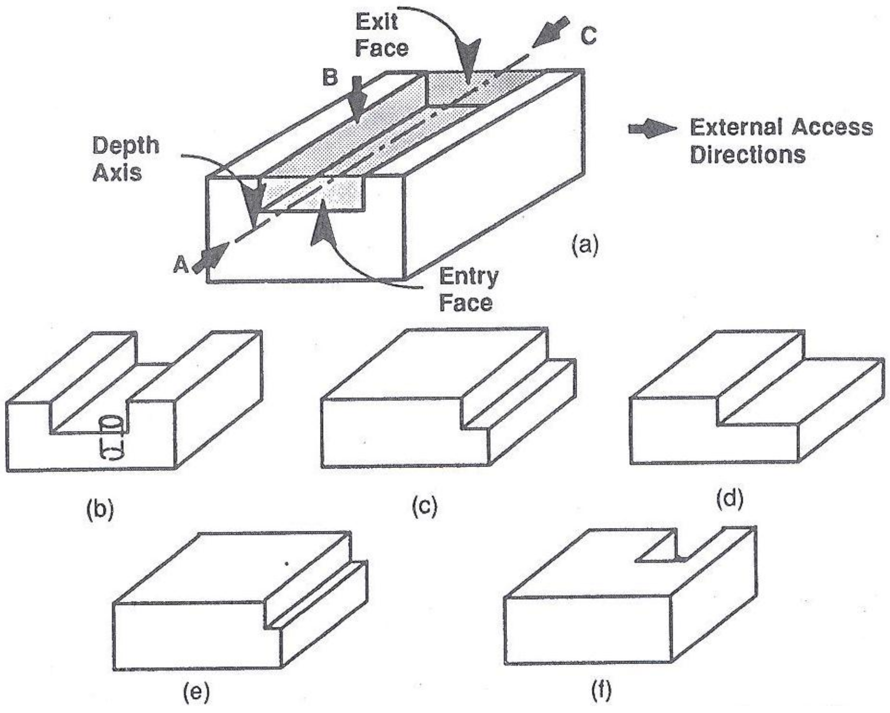

Figure 2. The External Access Directions and modifications to a Through Slot

Detection of the occurrence of a change in feature type is straightforward with a rigorous feature taxonomy based on a topological knowledge of primitive features. In the above simple example the component model in 2b has three external access directions (EAD's) whereas 2d and 2e have four and 2f has two. The modelling system must be capable of reporting the change, and the close relationship between the EAD's of the feature model and the imaginary faces in the geometric model make this a simple task.

Detection of a change in feature type must be followed by a determination of whether it was intentional or represents user error and, as design by feature systems have active human involvement, it would seem appropriate to let the designer make this decision. Any changes need to be reflected in the feature model, and again it is attractive to call upon the designer again so that in going say from 2b to 2c he would effectively delete the through slot and create a new step feature. In this simple situation this *may* be adequate, but in general some automatic procedure is required to ensure feature model integrity.

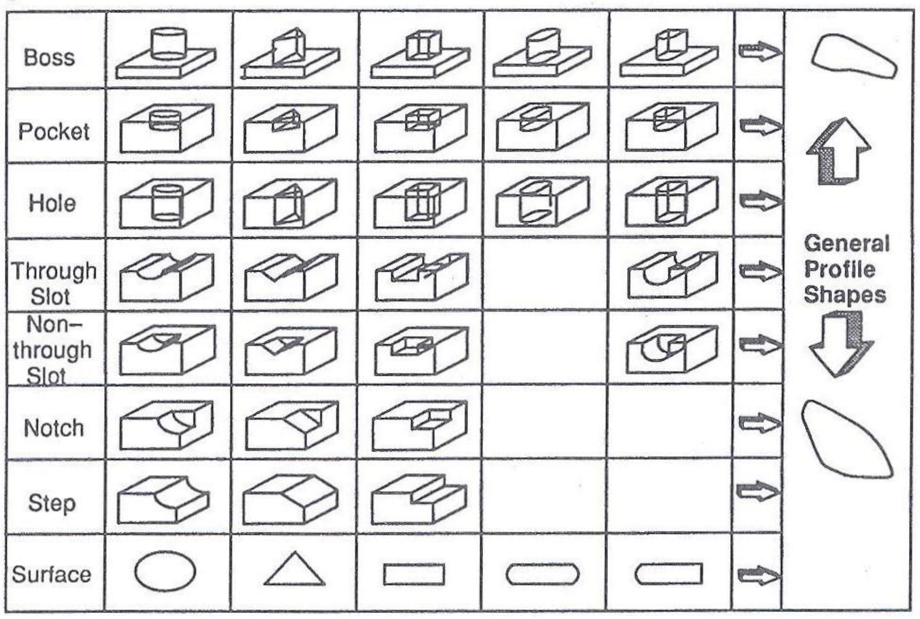

Figure 3: The library of feature primitives

The most complex aspect of validation is where compound features or relationships are involved. For example, when a through slot with a child hole is moved towards the edge of a component, the slot *may* change feature class to become a step, and child features *may* become non-existent or invalid (Figure 2b to 2e). Again, detection of the situation is the first concern, and a primitive feature system is well-suited to detecting the disappearance of the hole as it operates at a sufficiently low level. The action to be taken after detection can be highly problematic as it depends upon the explicit relationships established between the hole and the slot.

5. Feature Validation Algorithm

The feature validation mechanism first checks whether the created or edited feature exceeds the boundaries of the stock material. Should it do so, then reasoning about the geometry of the feature and the stock material results in the determination of the changes in feature class, size and relationships with other features. The designer must then decide whether to accept any changes and update the database. Acceptance of the changes causes a check to be made to determine whether the feature intersects with other features on the component, and resulting changes in feature class, size and relationships for all affected features are presented to the designer for acceptance or rejection.

The algorithm for feature validation is based on a set of rules for each feature primitive, and the underlying concept is to determine any changes in feature class and dimensions by establishing the axis direction of the LCS (Local Coordinate System) in which the feature exceeds the boundary of the stock material. The principle of the algorithm is best explained by reference to the example of a square pocket with the origin (LCS) at the middle point of its top face (figure 4).

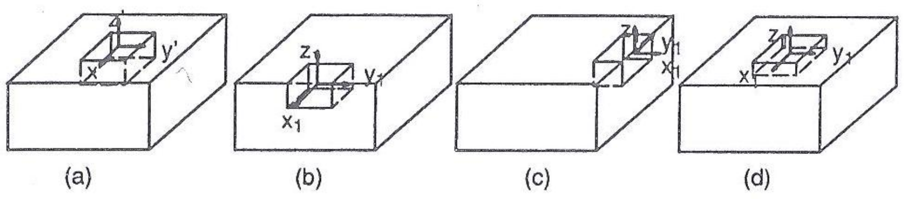

Figure 4. Examples of situations to be detected by the feature validation rules

The following situations must be detected. (1) If the pocket is moved in the +X direction and exceeds the boundary of the block (figure 4b) it becomes a non-through slot and the x dimension is reduced. If this is acceptable, the LCS for the new feature class is made to comply with the definition for non-through slots, i.e. the middle of the front edge as shown in the figure. Similar rules are applied for motion in the -X direction. (2) If the pocket is moved in the $+Y$ direction and exceeds the boundary of the block (figure 4c) the pocket also becomes a non-through slot, and the same considerations apply. (3) If the pocket is moved in the +Z direction (figure 4d), the pocket remains a pocket, but its z dimension is reduced. If the pocket is moved into the block (the -Z direction) it becomes a void. (4) If the pocket is moved in an arbitrary direction, the rules check the movement in the x, y and z directions separately and make decisions by considering the combined effect *of* the movements. Any movement that would place the pocket outside the block is detected as an error condition.

To detect the changes in feature relationships such as parent-child relationships and tolerances, the algorithm first retrieves the database to identify the existence of any relationships. The user is then responsible for controlling relationships associated with edited features. For example, when the pocket in figure 2b is moved, the feature validation mechanism searches the database and discovers that it has a child feature (the circular hole). The user can decide to maintain the parent-child relationship so that the hole is moved with the pocket or to break the relationship so as to leave the hole as a void in its initial position.

6. Conclusions

This paper has described the process of feature validation which is used to maintain feature model integrity whilst interactively creating component designs using a design by features system. The importance lies in ensuring that subsequent manufacturing planning activities are provided with enhanced information that can be guaranteed to not contain inconsistencies, ambiguities or false information. The algorithmic methods employed demonstrate that for this, and other activities, it is essential to describe feature primitives within a formal and rigorous topological taxonomy.

7. References

Case, K. and Gao, X., 1993, Feature Technology - An Overview, Int. J. of Computer Integrated Manufacture, Vol6, Nos 1 & 2, pp 2-12.

Gao, X., Case, K. & Gindy N.N.Z., 1992, A Design by Features Approach to the Building of Feature Data Models for Process Planning, Proc. of the 8th International Conference on Computer-Aided Production Engineering, Edinburgh University, pp 45-51.

Gindy, N.N.Z., 1989, A hierarchical structure for form features, Int. J. of Production Research, Vol 27, No 12, pp 2089-2103.

Gindy, N.N.Z. & Ratchev, T., 1992, Machine Tool Selection in Computer Aided Process Planning Systems. Int. J. of Integrated Manufacturing Systems, Vol 3, No 2, pp 32-36.