Functional requirements and conceptual design of the Feature-Based Modelling System

by Jami J. Shah and Mary T. Rogers
Arizona State University

The databases of contemporary CAD systems cannot be used to drive most CAE and CAM systems in an automated manner since CAD systems do not produce a complete product definition. The information that is supported is in the form of low-level details, from which higher-level information such as form features cannot be obtained easily. This paper describes the functional requirements of a system that will be capable of overcoming these deficiencies. FBMS consists of an advanced solid modelling shell and a feature mapping shell, offering richer data structures and the application of artificial intelligence techniques to geometric modelling.

Introduction

The current generation of computer-aided design and manufacturing (CADCAM) systems has made available tools for partial automation of some of the tasks related to product development and manufacturing. However, data transfer between these tools is limited to a few applications. Examples include finite-element mesh generation from geometric models and tool path generation for numerical control (NC) machines from geometry data.

Lack of integration is a major obstacle in the widespread use of CADCAM in routine engineering work. Current technology has not only created islands of automation, it has created islands of optimisation, too. It is highly desirable to develop tools that promote interaction between groups of specialists, which will eventually lead to global optimisation of products. For example, designers will be able to get feedback on manufacturability, allowing them to redesign less expensive products that are comparable in functionality.

The reasons for the lack of integration range from the inability to create complete product databases to a mismatch in the level of abstraction of information. In this paper the functional requirements for an integrated system are discussed and a conceptual design is presented. The system, designated as the Feature-Based Modelling System (FBMS), works in association with solid modellers to produce a complete product definition as a multi-tiered database that can be used by many automated applications.

Background

A solid modeller is the central element in contemporary CAD systems. It is used for defining the geometry of the product. If a complete product definition can be created and stored in a database, only then is it possible to drive applications software in an automated fashion. However, at present this is not possible because of the data structure of current solid modellers, as outlined below.

Two types of representation schemes are most popular: boundary representation (B-rep) and constructive solid geometry (CSG). B-rep modellers store the evaluated geometry as a redundant hierarchy of topological entities (faces, edges, vertices) with pointers to geometric entities (surfaces, curves, points). Euler-Poincaré equations are used to ensure the validity of objects. Point sets are classified as inside, outside or on the object, using a convention for the direction of surface normals.

CSG modellers store an unevaluated model in terms of the construction procedure. The data structure is a binary tree whose leaf nodes are primitive objects and whose interior nodes are Boolean operators. If valid primitives are used with regular Boolean operations, then no validity checking is required. A boundary evaluation program is used to determine the active boundaries of the constructed solid.

Solid modellers have two major deficiencies:

- *Incomplete product definition:* Only the nominal geometry can be defined; tolerances and surface finish cannot be represented. Material specifications, surface treatments, designer's instructions related to certain features etc. cannot be stored.
- *Low-level product definition:* Product definition is in terms of low-level details, i.e. geometry and topology for B-rep and primitives and operators for CSG. Only when an image is displayed from the above data can one comprehend the meaning of the data; i.e. the interpretation is done by the viewer. The database cannot be queried: for example, 'Is the part rotational machined or sheet metal?' cannot be answered. This is because higher-level information, such as form features, is missing.

It is clear that solid modellers cannot be used to drive applications such as process planning or manufacturability evaluation because some of the information needed by these tasks is totally absent from the solid modeller database. In addition, the entities on which, for example, process planning is based, such as form features, require a higher level of abstraction than that which is available.

Features

In this section the types of information and their level is examined. Let us first define a feature to be a set of information related to a part's description. The description could be for design purposes, or manufacturing and inspection or even for administrative purposes. Thus the nature of

these sets can be quite different. We find it convenient to classify features or information sets related to product engineering as follows:

- form features (nominal geometry):
  - ☐ functional
  - ☐ aesthetic
  - ☐ assembly aids
- material features (material composition and condition):
  - ☐ properties/specifications
  - ☐ treatment, applied to materials and surfaces
- precision features (allowable deviations from nominal geometry):
  - ☐ tolerances
  - ☐ surface finish
- technological features (information related to part performance and operation):
  - ☐ performance parameters
  - ☐ operating variables
  - ☐ design constraints.

Designers and application engineers reason in terms of these features, and so they must be supported in any integrated product engineering system. At this stage we are focusing our attention on the first three types, which can be used for complete documentation of a product's geometry and for supporting manufacturing applications.

Features and solid modelling

Three fundamental approaches have been proposed and tried for associating features with solid models. These methods are summarised below:

- *Human-assisted feature recognition:* This approach has been used in preparing input for process planning systems. Chang and Wysk (Ref. 1) created a system in which a planner could interactively mark surface types on a two-/three-dimensional image of the part, which allowed geometry data from the modeller to be grouped as form features and to be combined with tolerance information. Other developers of process planning systems have also used this approach. It is clear that this method is neither convenient nor efficient, but was necessary in the absence of a better product definition method.
- *Feature recognition and extraction:* In this approach a feature recognition program examines the database produced by a solid modeller and makes deductions about the types of features present. This approach has been applied to B-rep modellers (Refs. 2 – 5) as well as to CSG modellers (Ref. 6). The method amounts to making explicit what is implicit in the solid model. However, it cannot derive information that does not exist, such as tolerances and finish. There is also room for misinterpretation. The algorithms for recognising even simple features are fairly complex and are generally modeller-specific. Feature recognition and extraction is redundant effort which would be unnecessary if a method could be devised for retaining in the modeller all the information available to the designer.
- *Feature-based modelling:* This approach provides a means for building a complete database at multiple abstraction levels, right from the start of product development. Feature-based modelling systems will allow users to build models using features stored in libraries and to network these together as needed. This paper will elaborate this approach.

Literature survey of feature modelling

Pratt and Wilson (Ref. 7) laid down the functional requirements for the support of form features in a solid modelling environment. They created a hierarchy of feature types, with major classification done under explicit and implicit categories, the former being evaluated geometry and the latter unevaluated. Implicit features were further subdivided into modifiers and generic types. Both B-rep and CSG representations were considered for the underlying solid model. Pratt and Wilson also looked at the manipulation functions needed for creating models with feature. All the work was of a conceptual nature, although some concepts have since been reported to have been implemented at Cranfield Institute of Technology (Ref. 8).

Shah (Ref. 9) developed Pratt and Wilson's scheme further. He examined both the representation and the interpretation issues. Parameters used for defining design features were compared with parameters for manufacturing features. Miner (Ref. 10) developed a prototype feature modeller with limited geometric capabilities (planar surfaces only).

Dixon, Simmons and associate researchers have developed a number of feature-based applications (Refs. 11 and 12). Surfaces are not explicitly represented; features are application-specific and solid modelling functions such as Boolean operations were not supported. The database built has been applied successfully to manufacturability evaluation of castings and design of injection moulding.

Several studies have been done for supporting dimensional tolerances in solid modelling. Requicha (Ref. 13) discussed parametric and non-parametric approaches to tolerancing. The non-parametric approach is based on offset surfaces to define tolerance zones. Neither approach has been explored fully yet.

Faux (Ref. 14) reconciled earlier work done within CAM-I in the area of form features (Ref. 7) and dimensional tolerances (Ref. 15). A feature hierarchy was defined along with sets of rules. The lowest-level indivisible units are termed primitive features. These can be used to define connected features, which are sets of connected faces obeying certain declared rules. These, in turn, could be used for defining compound features and patterns. This decomposition into low-level elements is suited to tolerancing.

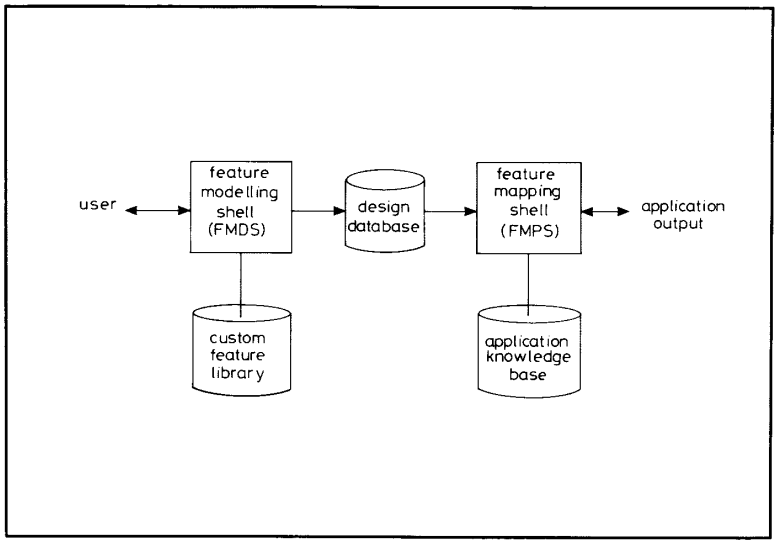

Fig. 1 The shell concept applied to FBMS

Functional requirements of FBMS

The requirements for implementing a system that will support form, precision and material features can be categorised into three groups: data support (structure and database), functions and environment. A system that is suitable for product definition and that creates a complete database required by engineering applications must satisfy the following general requirements:

- The data set supported must include all elements of the vocabulary of the tasks to be automated (Ref. 16). It is clearly

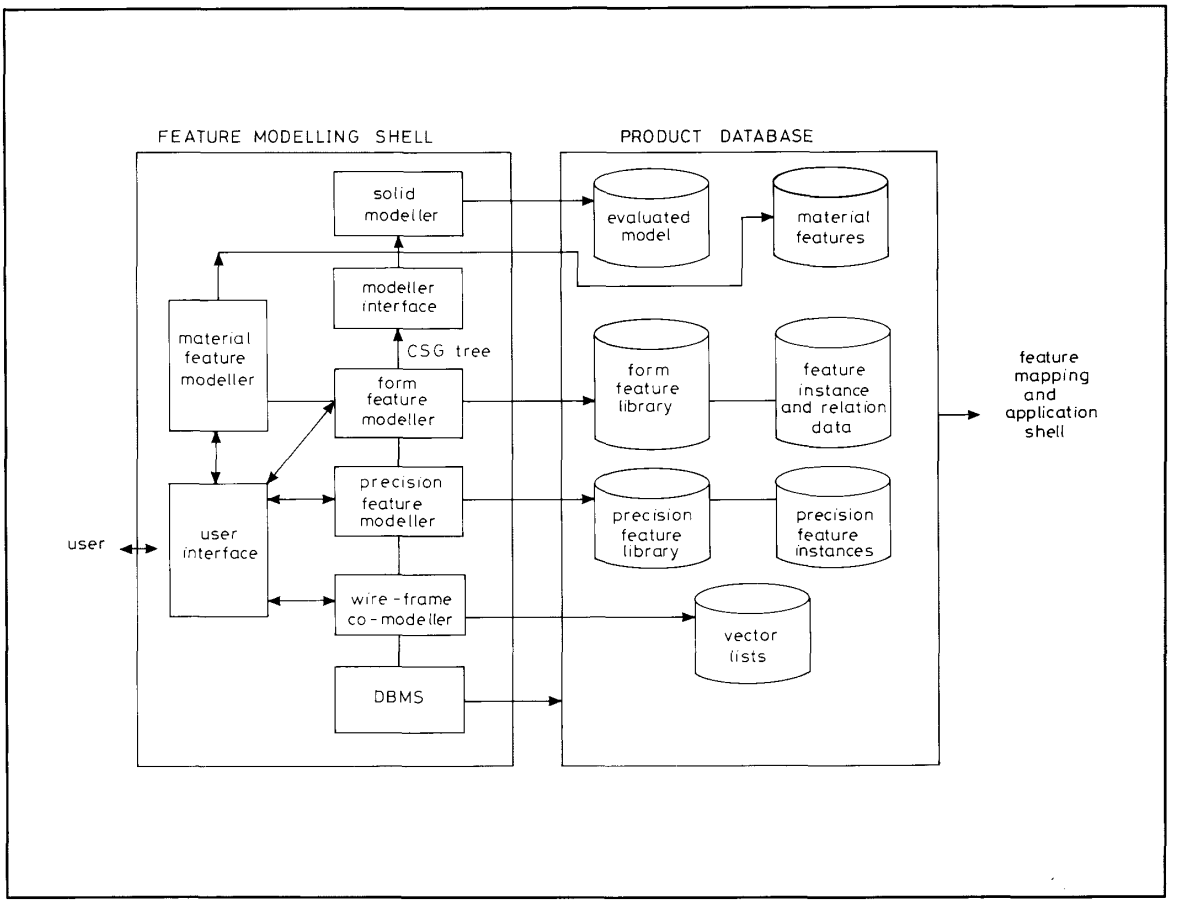

Fig. 2 Schematic diagram of the modelling system and the product database

necessary to incorporate all information needed by applications that will use the product database. The fundamental information sets have been identified as form features, precision features and material features.

- The mechanism for feature definition must be immensely flexible to allow designers to define features in any form, at any level, and in any combination, as appropriate to their needs. In order to use these feature definitions repeatedly, it should be possible to store the generic properties in a library so that features can be instanced.
- The product definition system (the feature modeller) must provide an attractive environment for creating, manipulating, modifying and deleting product models. It should be possible to define and instance form, precision and material features individually and to define relationships between form features as well as intra-feature relationships. For example, defining tolerances after creating the nominal geometry involves creating a network between form and precision features.
- Recognising the fact that the interpretation of features is application specific, it is necessary to create a feature mapping system that is also immensely flexible, so that it can be set up to support any application that uses the product database.
- It is useful to have product data at multiple levels of abstraction: generic parameters and feature relationships at the high level and conventional solid model data at the low level. These parallel representations must be managed in a manner such that consistency and equivalency are guaranteed.

FBMS overview

The requirement of producing a flexible product definition system as well as a flexible interpretation system can only be met by providing only the mechanism for supporting these objectives but not the actual knowledge: in other words, by providing a shell for definition (modelling) and a shell for applications (mapping). This concept is illustrated in Fig. 1. The feature modelling shell (FMDS) contains all the necessary facilities for creating a product database except the actual definition of features. FMDS can be customised by the organisation using it to define the features needed by their designers. Thus the feature 'knowledge' is added by the organisation to get a complete feature definition system. Once the customisation is complete, designers can start using FMDS to define products.

In order to use the design database, applications must extract relevant information from it. The information to be extracted will depend on the application as well as the direction that the solution/search takes; this is influenced by the contents of the database. A feature mapping shell (FMPS) is provided for the purpose of extracting and reformulating product data as needed by the application. The mapping process is intimately intertwined within the application: the mapping system must combine the extraction process with the application. The application knowledge can be incorporated in FMPS using frames and custom syntax to set up the application computation and extraction procedures. Again, customisation is left to the user, making the system immensely flexible.

In the following sections we will describe briefly each of the major components of FBMS.

Feature modelling shell — conceptual design

It was determined that the requirements called for the design of a distributed system, each subsystem supporting one type of

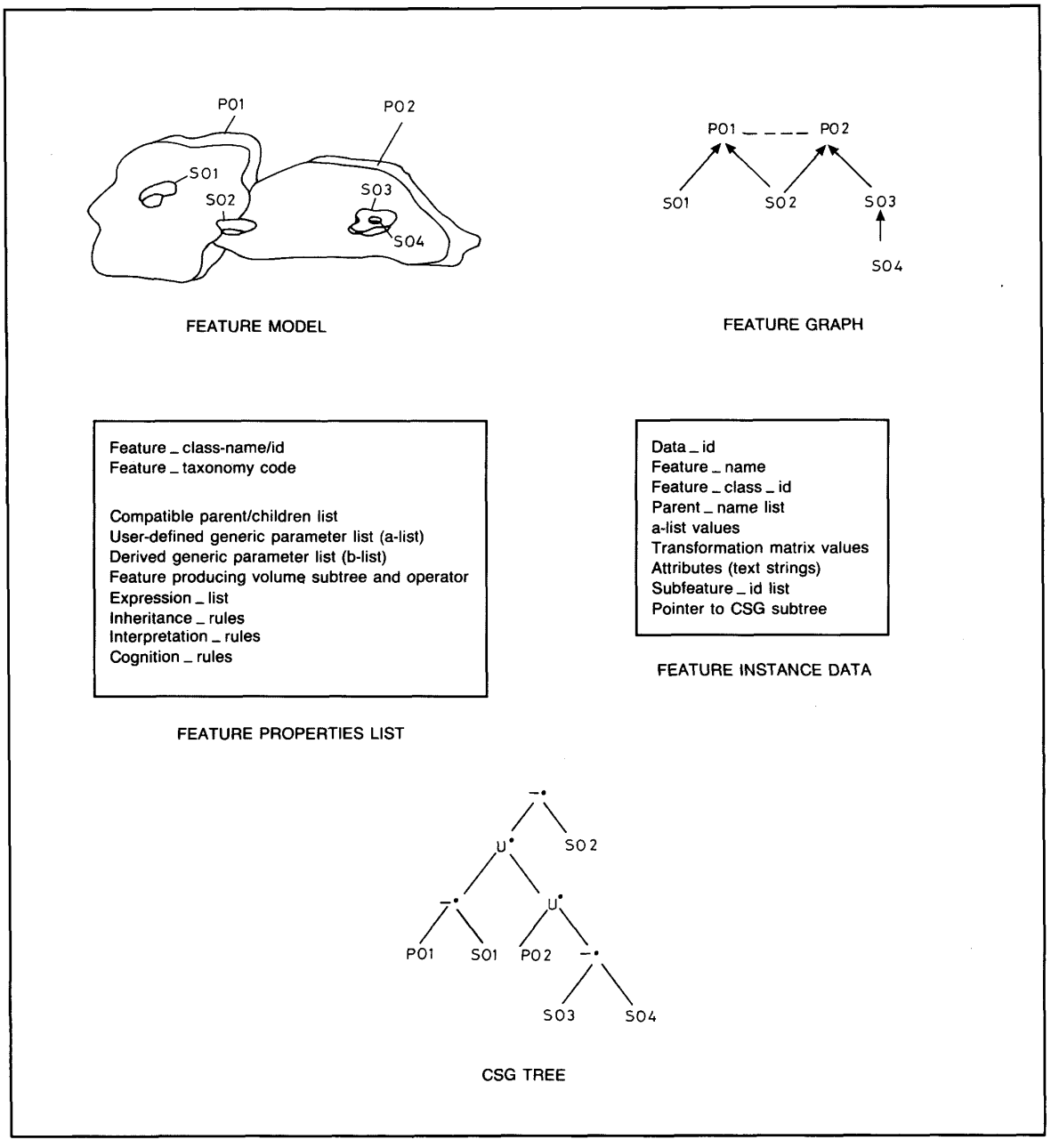

Fig. 3 Elements of the form feature modeller database

feature. Thus there are three subsystems:

- form feature modeller
- precision feature modeller
- material feature modeller.

Fig. 2 shows the system schematically. Each modeller operates in two modes:

- *Set-up mode:* This allows the definition of generic features by specifying feature properties in a frame-based system. This includes specification of user-defined parameters, inherited parameters, expressions, solid representation and sets of rules.
- *Modelling mode:* This allows generic features to be instanced and attached to the model, as well as related to other features.

The modelling mode is controlled by the user interface, which determines the modeller to be activated.

The wire-frame co-modeller has been incorporated to provide quick response to the user when interacting with the model. A parallel vector list representation of the model allows unevaluated wire-frame images to be displayed quickly. When the evaluated image is required, the user can send the form feature data to the solid modeller to examine evaluated active boundaries. In the following sections, the major characteristics of each type of modeller are explained.

Form feature modeller

The form feature modeller allows the nominal part geometry to be defined. Form features are treated as positive or negative volumes or implicit modifiers, which do not have an FPV — feature producing volume, considered as an independent solid — representation. Form features are used as the fundamental representation; precision and material features can be defined independently, and then networked to form features, as explained in later sections. It is convenient to divide form features into primary and subfeatures: primary features can be thought of as major shapes, while subfeatures are alterations to the major

shape. In terms of object-oriented programming, primary features do not inherit any parameters from any other features, while subfeatures inherit parameters from one or more features.

All types of features are considered to be frames or objects whose generic properties are defined in the set-up mode of the appropriate modeller. For form features, the feature property list is shown in Fig. 3. Features are defined by two groups of parameters: those to be supplied by the user (independent parameters), and those to be derived from other features (inherited parameters). These are stored as two ordered lists, referred to as the a-list and b-list, respectively.

Interpretive languages have been devised for writing rules and expressions as part of a feature's definition. These rules and expressions are used for creating an inheritance network and cognition rules. Inheritance rules and expressions are used for determining the values of the parameters in the b-lists using the parameters in parent features. Cognition rules are used for checking if the feature is being used in a valid way. This imparts a degree of 'intelligence' to the modeller and provides a high-level environment for the designer.

The solid representation of the form feature is stored as a feature producing volume (CSG subtree) and Boolean operator. This is expressed in a generic CSG syntax, which is later converted to solid modeller specific commands in the solid modeller interface. Sets of interpretation rules and taxonomy codes can also be incorporated if needed by the application.

The database associated with the nominal geometry has four tiers. At the top level is a feature relationship graph which records both adjacency links (broken lines in Fig. 3) and child-parent links (solid lines). In Fig. 3 the prefix P is used to label primary features and prefix S is used for subfeatures. The second level in the form feature modeller database is the definition of each form feature, consisting of generic properties and instance data. The third level is a CSG tree constructed from the FPVs of all features in the model. This global solid model can be translated into solid modeller specific commands and evaluated to give active boundaries, representing the lowest level of geometric and topological information. Fig. 3 illustrates these features of the database. Fig. 4 is a print-out of an example feature database showing generic parameters, FPV definition, and inheritance and cognition rules.

Precision feature modeller

The purpose of the precision feature modeller is to enable the definition of allowable deviations from nominal form and size, i.e. dimensional tolerances and surface finish. Even though there is an ANSI standard for specifying geometric tolerances (Ref. 17), many companies continue to use their own informal scheme, commonly referred to as the conventional method, which is also supported in the ANSI standard.

In a manner similar to that of form features, it was decided to leave the definition of precision features up to the user and to allow storing of such generic precision features in a library. These generic features can then be instanced and attached to form features and their geometric entities as appropriate. The definition of generic precision features involves specification of user-defined parameters and their meaning (for example roundness, or positive/negative deviations in size etc.), specification of datums if appropriate, and the entity type(s) to which the precision feature must be attached. Geometric tolerances also involve the specification of datum precedence as well as marking the datum planes with labels. Also needed are condition attributes such as the maximum-material condition or least-material condition.

We have classified generic precision features into four major groups on the basis of the location of the geometric entity data in the feature database. The four categories are as follows:

- *Type 1:* Precision features based on a parameter exclusively in the property list of a form feature:
  - size (length, diameter, radius)
  - shape (roundness, straightness, flatness, cylindricity).
- *Type 2:* Precision features based on relations between geometric entities that are used in defining form features:
  - real entities such as edges, surfaces etc. (angularity, squareness, parallelism)
  - imaginary entities such as centre of gravity, axis of symmetry etc. (coaxiality).
- *Type 3:* Precision features based on relations between form features (location and orientation):
  - when features are defined in the local datum of the parent
  - when features are defined in other datum systems.
- *Type 4:* Precision features based on one geometric entity of form feature (surface finish).

The reason for this classification is as follows. Precision features of type 1 can be incorporated in the property list of form features by means of pointers to parameters since the parameters are not influenced by other contents of the feature database. Types 2 and 4 must be attached to entities in the solid model.

Type 3 refers to relationships between features that are supported at the feature level. For primary features, this relates to the global position and orientation vectors; for subfeatures it relates to inherited as well as user-defined elements of the position and orientation vectors. Since subfeatures are positioned in the local co-ordinate system of the parent, if that system is also used as the datum, the tolerances can be specified directly in the subfeature property list. However, this will only be a special situation. Therefore it is necessary to provide a way of using any datum system; this is done by a facility called the 'transfer-through' mechanism, which can relate the feature co-ordinate system to any specified datum. It is essential to set up the system this way to maintain consistency.

The separation of form and precision feature modellers is also convenient for

```
**********************************************************************
Feature _ id          i   21202
Feature _ name        c   countersunk
Feature _ type        c   sub
Compatible _ parnt    c   hole _ 1r
Compatible _ sub      c
User _ def _ param    c   entity _ num      diameter      angle
Expressions           c    # (ab - c2/tan2.*/)
                           # (a2/)
                           # (ab - 2/)
Derived _ param       c   depth
FPV _ subtree         c   CON(a1,f1,f2,b1,0,0,0,0,0,0)if(a1 = 1)(0,0,0,0,
                          0,0)if(a1 = 2)(0,0,f3,180,0,0) -
FPV _ parameters      c    # (e2(s1,a2))
                           # (e2(p1,a1))
                           # (p2,a2)
Blank                 c
Blank                 c
Inherit _ rules       c    # ((s1,b1)e1((s1,a2),(p1,a1),(s1,a3)))
Cognition _ rules     c    # ((s1,b1)<(p1,b1))
                           # ((s1,a3)>0)
                           # ((s1,a3)<180)
                           # ((s1,a2)>(p1,a1))
                           # ((s1,a2)<(p2,a1))
                           # ((p1,a2)<e3((p2,a1),(s1,a2)))
Interpret _ rules     c
```

Fig. 4 Example form feature (countersunk)

designers, since they can work with nominal geometry in the early part of the design and specify precision features later only when and if needed.

Material feature modeller

The purpose of the material feature modeller component of the program is to associate material information with form features. Material information can be of the following types:

- material name
- material composition code (standard or proprietary)
- material composition, physical and mechanical properties
- heat treatments to be applied to entire part
- surface treatments.

Some of these are attributes of volumes, others of surfaces. Since the form feature modeller uses volumes as the basic representation, volume-related attributes can be linked directly with form features that correspond to positive volumes, i.e. when the Boolean operator in the FPV is a union operator. Surface attributes must use a greater level of detail than is available in the form feature database. These cannot be supported at present, but work is planned on devising a scheme for supporting such information as well.

Like form and precision features, material features can be defined and stored in a library. Material features can then be attached to form features.

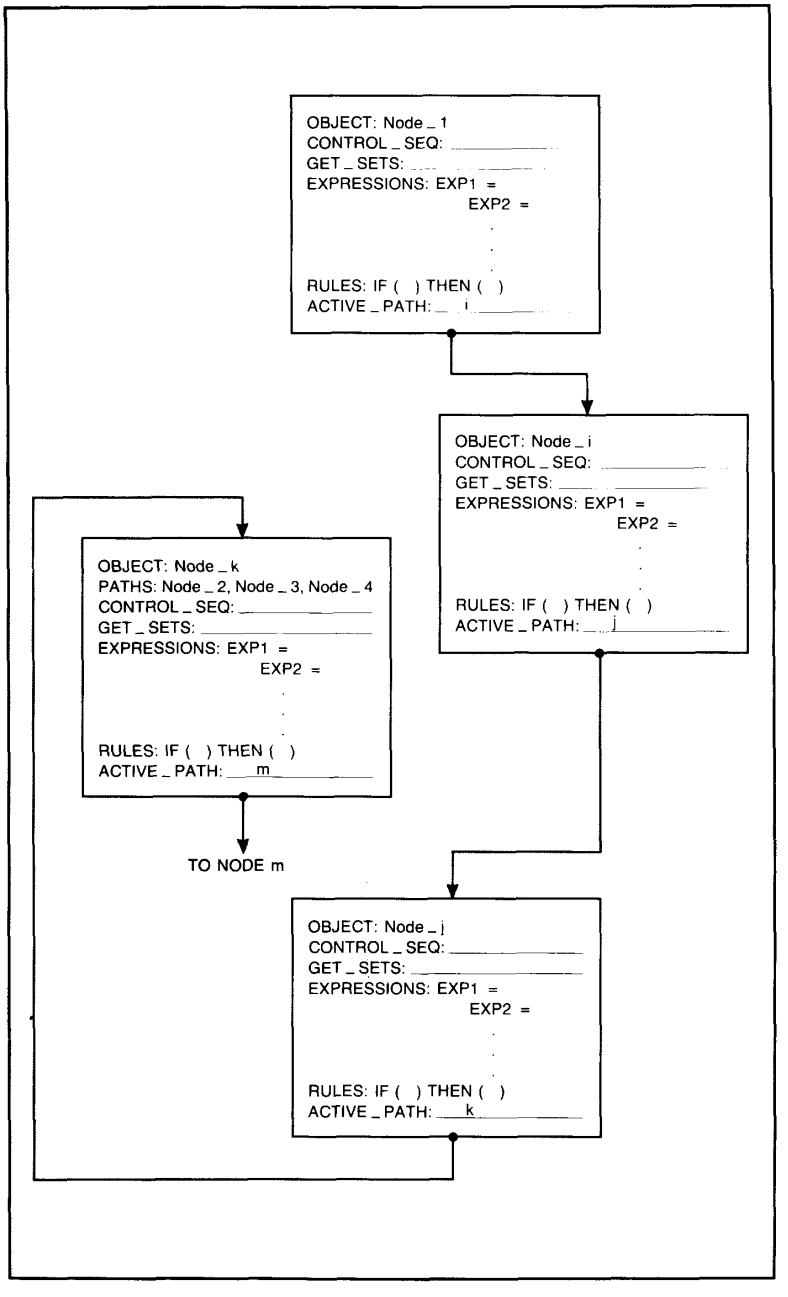

Fig. 5 Object-oriented search

Feature mapping

Interpretation of feature data is application-dependent: for example, finite-element automated systems will need information that is very different from that needed by a group technology (GT) classification expert system. Application-dependence goes even further in the case of GT: one GT scheme can be substantially different from another, thus making feature interpretation scheme-specific. However, a mapping mechanism that is generic is highly desirable, so that the system can be customised to different applications with minimal effort. With this motivation, we classified the design parameters required into three generic groups on the following basis:

- *Type 1:* This kind of parameter is available explicitly in the feature database; it is independent of other data in the model (model-invariant).
- *Type 2:* This kind of parameter can be calculated from those that are explicit in feature data of one feature. Thus parameters of this type are implicit but model-invariant.
- *Type 3:* This kind of parameter depends upon a relation between two or more features; i.e. it must be calculated from type-1 or type-2 parameters of several features. Such parameters are model-dependent.

Thus the mapping scheme must have mechanisms for extracting all three types of information. Type-1 parameters can be obtained through simple GET functions used in relational databases. Type-2 and type-3 parameters require computations and logical operations on type-1 parameters. Thus one of the requirements for creating generic shells is an interpretive language which can be used for writing specific extraction procedures. We have already built and tested a language for cognition and inheritance rules. Because the requirements are similar, we decided to use a similar language for interpretation rules.

Two other fundamental questions arise:

- Where the interpretation rule sets should reside:
  - ☐ in the feature database
  - ☐ in the mapping program, or
  - ☐ in the application program.
- What the nature of the relationship between the mapping and application programs should be:
  - ☐ sequential — the mapping program

processes *all* data to produce *all* parameters needed by the application program, the latter being activated only after the mapping process ends

- parallel — the application program interrogates the mapping program only when it encounters parameters that it must obtain from the product data.

As is apparent, these two questions are related. The sequential relation is inefficient because it will produce more information than needed by the application, since it has no advance knowledge of the direction in which the application will proceed. Also the sequential relation will have to be geared to an application, thus preventing the mapping shell from being generic. However, it is simpler to design and implement a sequential system than it is to obtain real-time interaction between the mapping and application programs. Because of the advantages of a parallel system, we decided to take this route. We are implementing the system along the following lines:

- Assume that applications to be supported can be modelled as a decision process. Many applications fall into this category: GT classification, process planning, manufacturability evaluation, finite-element mesh generation etc.
- Create a search system in an object-oriented environment. This concept is shown in Fig. 5. Each node is treated as an object; when the control sequence at the node is executed the path to the next node is determined, in addition to part of the solution being obtained. The control sequence involves firing of rules, extraction of parameters, computation of expressions and output to data files.
- Information is obtained from the product database as follows. Type-1 parameters are already in the slots of feature description. Type-2 parameters are defined as sequence of expressions that can be used regardless of the information in other instance data sets. Thus parameter type-2 expressions can be placed in feature property slots. However, since there are going to be many different applications that use feature data, many application-specific expression sets are required. Consequently, it is best to place parameter type-2 expression sets in the mapping program. Parameter type-3 expressions depend upon the model itself, not just the generic properties of individual features. These should be placed in the slots of node objects where they are to be used.

This mapping/application shell can be set up to support a variety of applications. Customisation is done by the user organisation. After new features have been defined in the form feature modeller library, each application engineer will examine the definition and create parameter type-2, parameter type-3 expression sets which will be attached to nodes in the application decision tree. If the feature is modified, a flag will warn application engineers to update the interpretation procedures. Application engineers will use the interpretation language interactively to define the nodes, the rules and expressions, and the control sequence needed to support the application. Parameter extraction functions are built into the shell so that these can be used with ease.

Conclusion

We have presented the design of a system that can be used both for product definition in association with solid modelling and for driving engineering applications in an automated manner.

The FBMS being developed can be customised by user organisations to support all elements of the definition of mechanical components, including form, precision and material features. It can also be set up to support a variety of data-driven applications by using a mapping shell that can automatically extract design parameters from the product database when needed by the application. With FBMS it will become possible to integrate software for design, engineering and manufacturing to a far greater extent than is currently possible.

To date, we have implemented the major components of the form feature modeller and the mapping shell, and we are in the process of implementing the rest of the system. We are also working on two specific applications of the feature database: automatic GT classification, and process selection for machining. The needs of these applications have had a major influence on the design of FBMS.

Acknowledgments

We would like to acknowledge the support of General Electric Corporate Research and Development Center and Texas Instruments, Defense and Electronic Information Systems. Their financial and technical support is helping to continue these projects.

References

1 CHANG, T. C., and WYSK, R. A.: 'An introduction to automated process planning systems' (Prentice-Hall, 1985)
2 GRAYER, A. R.: 'A computer link between design and manufacturing'. Ph.D. Dissertation, University of Cambridge, Cambridge, England, 1975
3 HENDERSON, M. R., and ANDERSON, D. C.: 'Computer recognition and extraction of form features: a CAD/CAM link', *Computers in Industry*, 1984, **5**, (4), pp. 329–339
4 KYPRIANOU, L. K.: 'Shape classification in computer aided design'. Ph.D. Dissertation, University of Cambridge, Cambridge, England, 1976
5 WOO, T. C.: 'Feature extraction by volume decomposition'. Proceedings of Conference on CAD/CAM Technology in Mechanical Engineering, Cambridge, England, 1982
6 LEE, Y. C., and FU, K. S.: 'Machine understanding of CSG: extraction and unification of manufacturing features', *IEEE Computer Graphics & Applications*, 1987, **7**, (1), pp. 20–32
7 PRATT, M. J., and WILSON, P. R.: 'Requirements for the support of form features in a solid modelling system'. Report R-85-ASPP-01, CAM-I, Arlington, TX, USA, 1985
8 PRATT, M. J.: 'Current status of form features research in a solid modeling context'. Draft Document, General Electric Corporate Research and Development Center, Schenectady, NY, USA, Dec. 1986
9 SHAH, J.: 'A scheme for CAD-CAPP integration'. Report, Automation Systems Laboratory, General Electric Corporate Research and Development Center, Schenectady, NY, USA, Aug. 1986
10 MINER, R. H.: 'A method for the representation and manipulation of geometric features in a solid model'. M.S. Thesis, Mechanical Engineering Department, Massachusetts Institute of Technology, Cambridge, MA, USA, May 1985
11 LUBY, S. C., DIXON, J. R., and SIMMONS, M. K.: 'Creating and using a features data base', *Computers in Mechanical Engineering*, 1986, **5**, (3), pp. 25–33
12 VAGHUL, M., ZINSMEISTER, G. E., DIXON, G. E., and SIMMONS, M. K.: 'Expert systems in a CAD environment: injection molding part design as an example'. Proceedings of 1985 ASME Conference on Computers in Engineering, Boston, MA, USA, Aug. 1985
13 REQUICHA, A. A. G.: 'Representation of tolerances in solid modeling: issues and alternative approaches'. General Motors Solid Modelling Symposium, Detroit, MI, USA, 1984
14 FAUX, I. D.: 'Reconciliation of design and manufacturing requirements for product description data using functional primitive part features'. Report R-86-ANC-GM/PP-01.1, CAM-I, Arlington, TX, USA, Dec. 1986
15 JOHNSON, R. H.: 'Dimensioning and tolerancing — final report'. Report R-85-GM-02.2, CAM-I, Arlington, TX, USA, 1984
16 SHAH, J. J., and WILSON, P. R.: 'Analysis of knowledge abstraction, representation, and interaction requirements for computer aided engineering'. Report, Department of Mechanical and Aerospace Engineering, Arizona State University, Tempe, AZ, USA, 1987 (under consideration by *Transactrons of ASME. Journal of Mechanisms, Transmissions & Automation in Design*)
17 'Standard for dimensioning and tolerancing'. ANSI Y14.5M, American National Standards Institute, New York, NY, USA, 1982

J. J. Shah is Assistant Professor and M. T. Rogers is Graduate Research Assistant with the Department of Mechanical & Aerospace Engineering, College of Engineering & Applied Sciences, Arizona State University, Tempe, AZ 85287, USA.