ELSEVIER

Computers in Industry 41 (2000) 3–24

COMPUTERS IN INDUSTRY

www.elsevier.nl/locate/compind

STEP-based feature extraction from STEP geometry for Agile Manufacturing

Mangesh P. Bhandarkar, Rakesh Nagi *

Department of Industrial Engineering, 342 Bell Hall, State University of New York at Buffalo, Buffalo, NY 14260, USA

Received 17 November 1997; accepted 10 May 1999

Abstract

Feature recognition, from low level geometric entities of product design representations within a CAD model to facilitate process planning and manufacturing activities, has been of significant importance in computer integrated manufacturing (CIM). However, the emerging paradigm of Agile Manufacturing has imposed additional requirements of ''neutral format'' so that form-feature information can be readily shared among multiple partners of a virtual enterprise. Recently, the STandard for the Exchange of Product model data (STEP) has emerged as the means for neutral form exchange of product related data. The ''STEP efforts'' have broken down the domain of manufacturing related activities in the form of application protocols (APs) target for specific functions which include drafting, configuration control and feature-based process planning to mention a few. Efforts are still on to increase the acceptance and use of this international standard (IS). This paper focuses on our efforts to support the STEP standard with the development of a standards-oriented form-feature extraction system. The developed feature extraction system takes as a input a STEP file defining the geometry and topology of a part and generates as output a STEP file with form-feature information in AP224 format for form feature-based process planning. The system can also be interfaced with a recent IGES to AP202 translator [M.P. Bhandarkar, B. Downie, M. Hardwick, R. Nagi, Migration from IGES to STEP: one-to-one translation of IGES drawing to STEP drafting data, accepted by Computers in Industry, July, 1999; M.P. Bhandarkar, Satisfying information needs in Agile Manufacturing through translation and feature extraction into STEP product data models, MS Thesis, State University of New York at Buffalo, 1997.] to allow conversion of legacy data. The feature recognition algorithm is boundary-representation (B-Rep) based and follows a sequential approach through an existing classification of features. Properties of each feature class are exploited to enable their extraction. The algorithm is currently developed for prismatic solids produced by milling operations and that contain elementary shapes such as plane and cylindrical surfaces (possibly using non-uniform rational B-splines (NURBS)). Special attention has been paid to implementation issues. We demonstrate the efficacy of the system using representative parts.

Keywords: STEP; Form feature; Feature extraction

1. Introduction

Computer-aided design (CAD) systems have been used for geometric modeling since the 1960s. Geometry is usually represented in the CAD systems in

* Corresponding author. Tel.: +1-7166452357; fax: +1-7166453302; e-mail: nagi@eng.buffalo.edu

0166-3615/00/$ - see front matter
PII: S0166-3615(99)00040-8

terms of low level geometric entities like vertices, edges and surfaces or in terms of solid entities like cubes, cylinders, etc. However, with the development of manufacturing technologies and the emergence of concepts like computer integrated manufacturing (CIM), there is a need for extensive cooperation between different engineering activities. More recently, the emergence of a new manufacturing paradigm called Agile Manufacturing has taken the concept of cooperation to a higher degree. Now, product data like design, manufacturing, utilization, maintenance and disposal is not only shared between various departments within a company, but is shared between various ''partner'' companies of a virtual enterprise. In this context, *feature extraction* from geometric representation and *feature-based modeling* have gained considerable amount of research importance as they allow the sharing of information between design and manufacturing process planning. Manufacturing world has seen a considerable amount of change, from the computer-aided drafting, computer-aided designing, computer-aided manufacturing to the new and emerging concept of Agile Manufacturing. In this new fragmented manufacturing world, where different companies share data and information about their products through a standard form of exchange, the representation of product data in multiple views through a standard exchange format (for multiple manufacturing applications) has become very essential. The ISO STandard for the Exchange of Product model data (STEP standard) has emerged as the means of neutral form for data exchange between companies.

This paper focuses on the extraction of feature information by converting low level geometry information into higher level manufacturing information. The extracted features are stored in the STEP AP224 format to facilitate their exchange between various ''partners'' and direct interface with an automated process planning system to create process plans for manufacturing.

The paper is organized as follows. Section 2 provides some background on the STEP international standard (IS) which is currently being developed, and an explanation of the STEP application protocol (AP) 224 for feature-based process planning. Section 3 presents a review of some of the work done in the field of manufacturing feature recognition. The characteristics and properties of features as they relate to the extraction process are detailed in Section 4 which is followed by the discussion of its implementation in Section 5. To demonstrate the efficacy of the procedure illustrative examples are presented in Section 6. Lastly, conclusions and recommendations for possible extensions of this work are discussed in Section 7.

2. Background

The manufacturing practices have undergone a lot of changes in the past few decades. The design process of developing drawings on paper and then converting them into blue prints has been replaced by CAD systems. The manufacturing process has also undergone change from single facility manual manufacturing to computer numeric control (CNC) machines and multi-facility manufacturing separated not only by location but also in the type of machines and systems that are used. Recently, a new trend in manufacturing called Agile Manufacturing has developed wherein separate companies come together to form a ''virtual enterprise'' to manufacture a product of the right quality with the least cost and at the right time. These new design and manufacturing trends have demanded a way for representing product related data in a single neutral format which can provide unambiguous data to support the various life-cycle stages of the product and also which can be understood by all the design (CAD) and manufacturing (CNC) systems. Emphasis on advanced techniques such as feature-based manufacturing and design have increased to reduce cycle time between design and manufacturing and to increase collaboration between design and manufacturing teams.

2.1. Motivation

The representation of design information as geometry and manufacturing information as form features in a neutral format is currently an actively researched area not only in academia but by industries too. Companies will be able to communicate electronic manufacturing information through simple media like electronic mail or over the Internet. Design and manufacturing of the part will not be restricted by geographic location any longer. The integrated product team (consisting of engineers from

various engineering disciplines) can develop the design in one part of the world, save this design as STEP data, and then pass the ASCII file to the manufacturing site where the same data can be viewed as per the manufacturing requirements.

This process essentially pushes all the input that the manufacturing and other ''post-design'' activities can provide into the design level of the life-cycle of the product. All the down stream activities (manufacturing, assembly, etc.) would now become views of the same standard data relevant to that particular application. Shifting all the intelligence into the design process leads to considerable cost saving. Industry estimates show that to make changes during design only cost a tenth of what it would cost in the later stages of the life-cycle of the product. This process also leads to the concurrent development of the design and manufacturing process plan of the product.

However, we are currently a long way from developing an integrated product. There is also the issue of legacy data which have not been developed with the concurrent engineering notion in mind. An IS STEP is being currently developed to address the issue of having standard data spanning the various stages of the life-cycles of product. Motivated by this effort the authors have tried to develop a mechanism for extracting features from the design data of the product. These features can act as input to an automated process planning system or also as input to a CNC machine for manufacturing the product directly. By extracting the features out of the design data the authors hope that the designer and manufacturer would have to work together and be forced to develop a ''manufacturable product'' at the design stage. Furthermore, the extracted features could be stored into a database and can act as design data for future design activities.

Section 2.2 provides some definitions for the term *feature* with regards its interpretation in different phases of product definition. Following this, an overview of the STEP standard and the AP relevant to the current effort are provided.

2.2. Feature definition

Without getting into the complex notation for a formal mathematical definition, a feature can be defined as characteristic of the part which carries significance or higher semantic meaning to a particular application. These various applications could be manufacturing, engineering, design, assembly, etc. The meaning of the term feature as it may apply to these disciplines is provided below.

In works of Shah and Rogers [31], the term *feature* was defined as a set of information related to an object's description. This description could be for design, for manufacturing or even for administrative purposes. The authors have classified features into sets related to product engineering applications as follows:

- form features: which identify the combination of geometric and topological entities in such a way that it makes practical sense during the various stages of the products life-cycle; for instance, shoulder and boss are examples of form features which are important during design and manufacturing;
- assembly features: which assist in the easy location/mating of parts for assembly, e.g., holes, slots, etc.;
- material features: which specify material composition and condition information such as properties/specification or treatment applied to materials and surfaces;
- tolerance features: such as geometric tolerances or surface finish; and
- functional features: such as performance parameters, operating variables or design constraints, e.g., the aerodynamic shape of the wing of an airplane.

In summary, *form features* are configurations on the object that may be for engineering-analysis during design, for process planning during manufacturing and during assembly. The same configurations could carry different connotation to different applications such as design and manufacturing. For example, holes, pockets and steps are types of form features that are represented as a set of surfaces during design and can be associated with manufacturing activities like drilling, end milling and slab milling.

2.3. Standard for the exchange of product model data

Recent efforts in the development of a standard mechanism for exchanging and sharing product data

have led to the development of the STEP standard. STEP provides a representation of product related information along with the mechanisms and definitions to enable product data exchange and sharing. The data generated is associated with the complete life-cycle of the product and can be exchanged between different computer systems and used by the various manufacturing stages such as design, manufacturing, utilization, maintenance and disposal.

STEP is based on a three layered architecture: (i) the reference model which develops a number of topical models specific to individual applications, (ii) the logical layer which specifies the format definition language called EXPRESS [27], and (iii) the physical layer which defines the communication file structure called STEP file. This file transfer mechanism represents the static aspect of STEP which allows exchange of product data. STEP also allows dynamic sharing of data between different systems through the standard data access interface (SDAI).

STEP is organized as a series of parts (shown in Fig. 1). These parts are published separately so that each can be independently developed and vendors need not implement all the standards into their systems. These parts fall into one of the following classes.

- Introductory: This class provides an introduction to the concepts and fundamental principles of STEP, e.g., Part 1 — Overviews and fundamental principles.

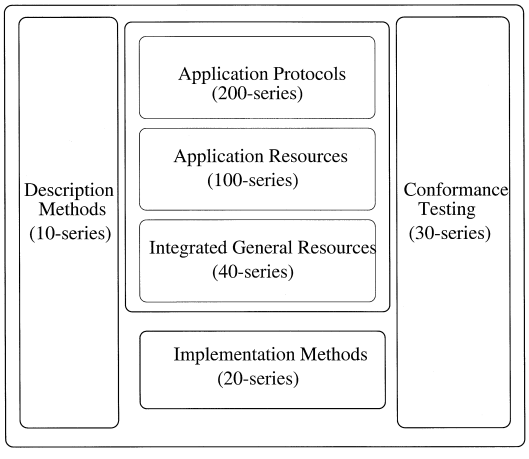


Fig. 1. Relationships between STEP parts.

- Description methods: The standardized methods to be used when describing STEP entities to ensure consistency and avoid ambiguity are described in this class, e.g., Part 11 — EXPRESS.
- Resource information models: This class defines the data content that is the basis for the development of the APs. These include models of general applicability and those that support a particular application or class of applications. The product data is represented in an application independent format, and is only implemented via an AP, described next, e.g., Part 41 — Fundamentals of product description and support.
- APs: This class defines the resources information models to provide specific functionality. APs state explicitly the information needs of a particular application, specify an unambiguous means by which information is to be exchanged for that application, and provide conformance requirements and test purposes for the conformance testing. APs are based on the resource information model but carry specific semantics in the application domain of the AP, e.g., Part 203 — Configuration controlled design.
- Implementation methods: This class describes the multiple implementation methods that are supported by the logically complete information model, e.g., Part 21 — Clear text encoding of the exchange structure and Part 22 — SDAI specification.
- Conformance testing methodologies: The standard procedures and tools required to undertake conformance testing of products are described in this class, e.g., Part 31 — Conformance testing methodology and framework: general concepts.

STEP uses a formal information modeling language, EXPRESS [27], to specify the product information to be represented. The use of a formal language enables precision and consistency of representation and facilitates implementation. APs are used to specify the representation of product information for one or more applications. APs are based on four main ideas: (i) scope and context of application, (ii) an application reference model (ARM) defining the requirements, (iii) an application-interpreted model (AIM) which satisfies the requirements given in the ARM using STEP constructs, and (iv) conformance requirements and test procedures for compliance with the AP. Fig. 2 shows the AP development process [36].

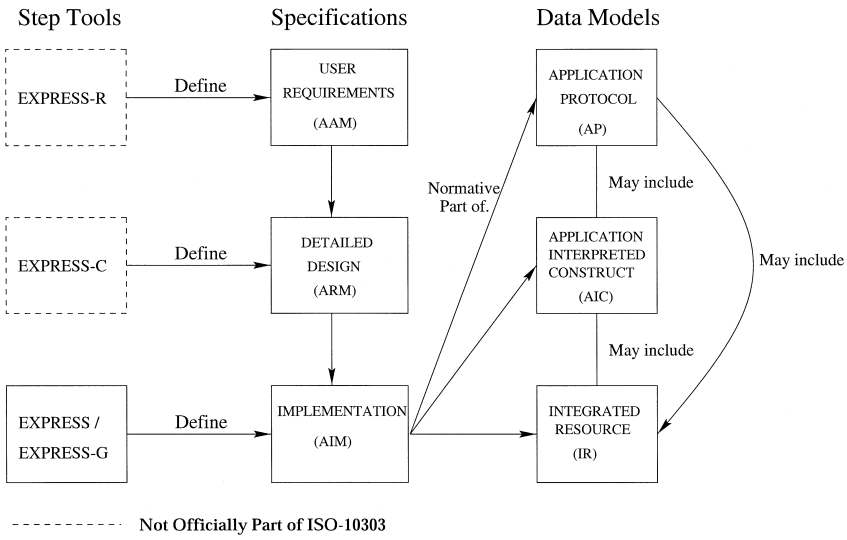


Fig. 2. AP development process.

STEP enables all people contributing to the design, manufacturing, marketing and supply of a product and its components to contribute to, to access, and to share information. STEP also attempts to unite manufacturing efforts among corporate partners, distant subsidiaries and suppliers across diverse computer environments. It is intended to fit in completely with the new emerging paradigm of virtual enterprises and agile manufacturing.

Several APs have already been developed to support different kinds of engineering applications and quite a few are currently being developed for various other manufacturing processes. Currently, the most widely used AP is the IS AP203 which is meant for representing design and configuration management information. Other APs like AP201 for explicit drafting, AP202 for associative drafting have also become IS. The AP for form feature-based process planning, AP224, which we have used in this paper for representing the form-feature information is currently at the final draft international standard (FDIS) stage. More details on STEP APs can be found in Refs. [21,33].

2.4. Application protocol AP224

AP224 is the STEP AP [23] which specifies the requirements for the representation and exchange of information needed to define product data necessary for manufacturing single piece mechanical parts. The product data is based on existing part designs that have their shapes represented by form features.

AP224 covers the following:

- product data that defines a single piece machined part to be manufactured;
- product data that covers parts manufactured by milling or turning;
- product data that is necessary to track down the customer order in the shop floor;
- product data necessary to identify the status of a part in the manufacturing process;
- product data necessary to track raw stock certification;
- product data necessary for tracking of a part design deficiency;
- form features that are necessary for defining shapes necessary for manufacturing.

This paper uses the feature representation capability of AP224 to give a higher level representation to the geometry and topology information of a part by way of form-features. The representation of product data in terms of form features helps in the fabrication of parts in the manufacturing environment. AP224 also contains the constructs to completely define the geometry and topology, attach tolerance values to the geometry elements and describe material specifications for the part. All this helps to provide the relevant information for the downstream operations like tool selection, speed and feed selection, etc. The

AP224 data can thus act as input to a STEP-based generative process planning system to create process plans consisting of routings, i.e., workstation selection, tool selection, feed and speed data along with complete manufacturing times, detailed bill of materials and manufacturing process plans.

The AP cannot be used for the representation of the following information:

- product data for the representation of assemblies (AP203 can be used for this kind of representation but it contains no form-feature information);
- product data for representation of composite materials;
- product data for the representation of sheet metal manufacturing;
- product data for the representation of part pedigree;
- data pertaining to the design phase of the product development; and
- product data necessary to schedule and track the progress of the part through the manufacturing process.

```
#38 = PRODUCT_DEFINITION_SHAPE('product shape','shape for product',#37);
⋮
#1690 = MANIFOLD_SOLID_BREP('',#1700);
#1700 = CLOSED_SHELL('',(#1710,#2080,#2450,#2540,#2620,#2700,#2770,#2860,
#2940,#3020));
#1710 = ADVANCED_FACE('',(#1720,#1900),#470,.T.);
#1720 = FACE_OUTER_BOUND('',#1730,.F.);
#1730 = EDGE_LOOP('',(#1790,#1830,#1870,#1890));
#1740 = EDGE_CURVE('edge_0',#1750,#1770,#520,.T.);
#1750 = VERTEX_POINT('vertex_0',#1760);
#1760 = CARTESIAN_POINT('',(0.,0.,0.));
#1770 = VERTEX_POINT('vertex_1',#1780);
#1780 = CARTESIAN_POINT('',(0.,10.,0.));
#1790 = ORIENTED_EDGE('',*,*,#1740,.T.);
#1800 = EDGE_CURVE('edge_1',#1770,#1810,#550,.T.);
#1810 = VERTEX_POINT('vertex_2',#1820);
#1820 = CARTESIAN_POINT('',(10.,10.,0.));
#1830 = ORIENTED_EDGE('',*,*,#1800,.T.);
#1840 = EDGE_CURVE('edge_2',#1810,#1850,#580,.T.);
#1850 = VERTEX_POINT('vertex_3',#1860);
⋮
#3090 = ADVANCED_BREP_SHAPE_REPRESENTATION('',(#1690),#460);
#3100 = ROUND_HOLE('','',#38,.T.);
#3110 = PRODUCT_DEFINITION_SHAPE('','',#3100);
#3120 = CARTESIAN_POINT('',(0.0,15.0,0.0));
#3130 = DIRECTION('',(0.0,0.0,1.0));
#3140 = DIRECTION('',(1.0,0.0,0.0));
#3150 = AXIS2_PLACEMENT_3D('',#3120,#3130,#3140);
#3160 = SHAPE_REPRESENTATION('',(#1050,#1070,#1080,##2000),#29);
#3170 = SHAPE_DEFINITION_REPRESENTATION('',#3110,#3160);
⋮
#3300 = HOLE_BOTTOM('bottom condition','flat',#3280,.F.);
#3310 = DESCRIPTIVE_REPRESENTATION_ITEM('blind bottom orientation','hole end');
#3320 = SHAPE_REPRESENTATION_WITH_PARAMETERS('',(#3310),#29);
#3330 = PRODUCT_DEFINITION_SHAPE('','',#3300);
#3340 = SHAPE_DEFINITION_RELATIONSHIP('',#3330,#3320);
⋮
```

Fig. 3. Part of a STEP AP224 file.

Fig. 3 is used to illustrate (in text form) the arrangement of form-feature entities in an AP224 STEP file. The form features have been divided into three major sub-categories, each subtypes of the *shape aspect* entity, namely: *machining feature*(s) [1], *transition feature*(s), and *replicate feature*(s). Other support entities (also subtype of shape aspect) such as profiles, paths and bottom conditions are defined to help completely describe the above form-features. The features are specified by a volume that is defined by a profile and by a path along which the profiles are swept. A feature is defined by creating an instance of the appropriate entity, for example, Fig. 3 describes a product containing a round hole. The feature's definitional parameters (say in the above example the diameter, orientation and maximum depth of the hole) are associated with the feature using the *shape definition with parameters* entity. The profiles, paths and bottom conditions for a feature are associated creating instances of the specified type such as *circular closed profile, hole bottom*, etc. The form-feature is associated with the profiles using the *shape definition relationship* entity.

3. Literature review

The field of feature recognition and feature-based design has been considerably researched. In feature recognition, the solid model for the part is first created and then the manufacturing features are identified or extracted from this model. On the other hand, in the design by feature approach, there is a database of manufacturing or other type (depending on interest for feature extraction) of features (e.g., holes, slots, grooves, etc.) which the designer can use to synthesize the part design. Other information such as manufacturing tolerances, datum information, surface attributes and material specifications can be specified by the designer. This section presents the review of some research work that has been done in terms of extraction of form features from CAD information models. There is also a review of some of the STEP-based feature extraction methods that have been developed recently. A more complete review can be found in Ref. [4].

3.1. Manufacturing feature recognition

In this section, we review some of the literature that is available for the field of manufacturing feature recognition. This particular field is extensively covered because it is of particular relevance to the current work especially considering that AP224 is used to describe machining features for manufacturing of a single piece mechanical part. A large number of techniques have been developed for features recognition. These techniques fall under the following categories [29].

3.1.1. Sectioning methods

This method is typically used for tool path generation and automatic process planning for 2.5D components encountered in the aerospace and avionics industries. Yue and Murray [40] have described a technique for automatic process planning of such types of parts. The technique involves validating the 2.5D components by testing the workpiece for approach direction, presence of non-planar faces, etc. The part is checked for presence of sharp corners, fillets, chamfers, hidden surfaces, tolerance requirements and cutter interferences with the fixture and with the component. The part volume is sliced in the X–Y plane to get the machining profiles and intersection curves. An NC tool path can be generated using the same.

Jung and Lee [17] developed a methodology for interfacing CAD and computer-aided process planning (CAPP)-based on automatic feature recognition for rotational parts. The methodology uses the IGES geometry representation of CAD data and identifies features using polygon decomposition and recognition of precedences.

3.1.2. Convex hull decomposition

The convex hull decomposition technique decomposes a volume by subtracting it from its convex hull and repeating the process for all the resulting volumes. Woo [38] developed this type of algorithm, in which a volume is decomposed into alternating sum

[1] Italicized text represents entity names and at times may not make grammatical sense.

of volumes (ASV) and the features are extracted from it. The algorithm however has a problem of non-convergence resulting in erroneous volumes of features.

Kim [18] developed a product shape recognition algorithm using convex decomposition, and extending the original ASV to avoid non-convergence. The algorithm called alternating sum of volumes partitioning (ASVP) can be applied to polyhedral solids. ASVP decomposition is a hierarchical decomposition of the boundary faces of the given solid. The ASVP is converted into a form feature decomposition (FFD) wherein all the components represent meaningful high level shape entities. However, this technique can only be applied to polyhedral solids. Pariente and Kim [22] have integrated an incremental update form feature recognizer with the earlier system. The system can update the ASVP structure if any changes are made to the design.

Ferreira and Hinduja [9] have also described a feature recognition technique that is applied to 2.5D components created from a boundary representation (B-Rep) solid modeler. The method involves determination of the convex hull for the component's faces. The approach works on a face-by-face basis and so it is not capable to dealing with complex features involving interactions between faces. The system can also determine possible directions of approach for the cutter and machining depths.

3.1.3. Boundary-based methods

The methods that take B-Rep data as input and use geometric and topological relations between the boundary entities fall under this category. The B-Rep tree is scanned and the features are identified by comparing the tree structure with the structure required for a feature. Joshi and Chang [16] use the concept of attributed adjacency graph (AAG) for the recognition of machining features from a B-Rep of a solid. The AAG is constructed in which every face becomes a unique node and every edge exists as a unique arc. The feature recognition procedure is called to compare the nodes of the graph to a pre-defined feature library. If the configuration of a particular pre-defined feature matches that of the graph, a feature is identified and extracted out. Interactions between features and virtual features are also recognized by the procedure.

Gavankar and Henderson [11] have developed a graph-based technique for the identification of protrusions and depressions from B-Rep of solids. Protrusions and depressions of the solid are found by examining the internal face loops of the faces. The technique cannot identify blind holes and pockets that open into more than one surface.

Ghad and Prinz [12] developed a shape-feature recognition system that is based on differential depth filter, which reduces the number of topological entities. The topological entities are transformed into entities of a higher level of abstraction called loops. Loops assist in reducing the number of entities which need to be searched for the feature to be extracted. This algorithm has been demonstrated to yield faster results with parts having a large number of faces and edges. Barber [2] has proposed a knowledge-based system for retrieval of relevant data for planning manufacturing applications. The system provides: (i) multi-level abstraction of part geometry, (ii) feature information in an object-oriented, semantic net representation, and (iii) a mechanism to retrieve previously stored process plans.

Gu et al. [14] use B-Rep solid model data to construct an attributed adjacency matrix (AAM). A feature recognizer uses fuzzy connectionist model, to convert the AAM into feature patterns. The system is capable of learning, and can recognize new features without any a priori knowledge. Brun [5] has presented an extraction procedure, where form-features are related to the modification of gross shape. The form features are classified based on the type of geometric element it modifies: a vertex, an edge, the interior of a face, or between two faces sharing a common edge, or between n faces sharing n edges and a vertex. The possibility of intersection between proposed features has been eliminated to avoid complexity.

Su et al. [35] have proposed a hybrid representation scheme called enhanced CSG tree of feature (ECTOF) which integrates the feature model with the solid model. The system can resolve feature interference by decomposing the intersecting features, removing redundant feature volumes, reclassifying features, and forming complex depression features.

Qamhiyah et al. [24,25] have presented a boundary-based procedure for the sequential extraction of

form-features. The authors have developed a classification of form-features based on their effect on the basic shape of the part. A loop-adjacency hyper graph (LAHG) is developed for the part from its B-Rep representation. The form feature extraction procedure is sequential and has five stages. Each stage goes through a particular type of form-feature class to identify the constituent form features. After a feature is identified, the set of loops constituting the feature are eliminated from the object. At the end of each stage, a valid B-Rep is reconstructed for each of the extracted components.

Liu et al. [20] have developed a PC-based system coded in C for form feature-based computer-aided process planning system. The automatic feature extraction system (AFES) for prismatic parts works off a CAD system. The AFES parses through the IGES B-Rep tree and extracts out the feature information. The algorithms proposed include concave edge test, oblique convex feature extraction algorithm, and the complex convex minimal enclosing box feature extraction procedure.

In summary, the boundary-based feature recognition is the most popular method for feature recognition. However, it does suffer from the lack of robust algorithms, particularly where feature interactions are present. Feature interactions remove portions of the features involved and the boundary-based methods fail to recognize partial or incomplete features. Also, other than Ref. [12], most of the B-Rep-based feature extraction methods are computationally expensive as they involve the traversal of the B-Rep tree to arrive at the face-edge relations.

3.1.4. Cellular decomposition

These methods have been applied for the determination of machining volumes from stock and part models. The Boolean difference between the volume of the stock and the volume of the final part yields the total volume to be removed. The volume is then decomposed into individual pieces corresponding to specific machining operations. CAM-I developed at General Dynamics was the earliest example of this type of feature extraction method. Sakurai and Chin [26] have developed a form feature recognition method called ''spatial decomposition and composition'' which decomposes the space surrounding a solid model into minimal convex cells. Various combinations of the cells are composed to determine if they are volume features. This process continues until all the features are identified. This process is however computationally very expensive.

Vandenbrande and Requicha [37] developed a method for feature recognition from a solid model of the part. The part is processed by production rules that generate hints for the presence of features by combining part faces and other attribute specifications about the face. These hints are further parsed to identify the form features. Dong and Parsaei [7] proposed the concept of general manufacturing features (GMFs) for feature extraction using the blank surface–concave edge (BS-CE) algorithm. The extracted features are then recognized using a rule-based system. Coles et al. [6] have developed a feature recognition method by generating volumetric feature representations from conventional B-Reps of parts. The features are recognized by decomposing the feature volume of the part into a set of smaller volumes through analytic face extension.

These systems are in the initial stage of development. Most of these systems lack generality and have been developed for specific use. Also, most of these systems can deal with simple prismatic solids with orthogonal features. There is also the obvious disadvantage of presence of multiple models, i.e., non-uniqueness for a required application. However, some of the cellular decomposition methods particularly Refs. [7,37] have been shown to deal with feature interaction to a great degree. These methods also have the advantage of being able to recognize features from non-manifold solids, although at the cost of computational effectiveness.

3.2. Design by feature

Design by feature is the other major school of thought that can be found in form feature related research. This approach consists of designing the part using a set of features from a database of features. The main advantage of this process is that since the features are essential for use in other downstream applications such as process planning and assembly planning, it makes sense to specify them at the design stage.

Finger and Safier [10] have aimed to enable designers to compose mechanical designs from high-

level features. The system is also capable of providing manufacturability, assemblability, functionality and cost feed back during the designing process. Avasarala and Stern [1] have implemented a form-feature-based modeler for the design of turned parts. The environment provides the designer with high level features for conveying the design intent without manipulating the low level graphics. The authors have pre-defined several classes of axisymmetrical and non-symmetrical features. However, the system is restricted only to turned parts.

The design by feature method seems to overcome all the problems of feature recognition but suffers from its own disadvantage of being limited by the number of features in the pre-defined library and the existence of large quantities of legacy data which cannot be used by this method.

3.3. STEP-based feature extraction

The ongoing effort to develop an IS for product data exchange has lead to the development of the STEP standard. Gilman [13] has demonstrated the use of product data exchange using STEP (PDES) [2] form features in a feature-based designing environment. The work focuses on the implementation of the PDES form feature information model (FFIM) as a conceptual schema for an object-oriented database.

Some of the other work in this area includes that of Ssemakula and Satsangi [32] for describing interfaces between process planning systems and CAD systems using the PDES standard, Shah and Mathew [30] which includes developing a translator to and from FFIM to ASU testbed developed at Arizona State University and Wu et al. [39] for computer-aided engineering (CAE) and CAM applications for mechanical systems.

3.4. Objective

There is a need for converting the design information from geometry and topology into manufacturing relevant form-features. The literature review presented earlier in this section can be summarized as: (1) form features capture more information and practical data than just low-level topological entities; (2) CAD packages, other than those based on feature-based modeling techniques, store part information in the form of low-level entities which cannot be used directly for downstream manufacturing activities. On the other hand, feature-based designing suffers from the distinct limitation of the number of features that can be created; (3) The existing feature recognizing systems are restricted to the use of certain form of input (from a certain CAD) system and produce native output either as textual information or as feature information into a file. This does not lend itself easily to communication and data sharing between different partner companies as warranted by the Agile Manufacturing paradigm.

The objective of this work is to develop a standards-oriented form-feature extraction system which converts design data into relevant manufacturing data. The system uses the STEP definition of the part as input to the extraction process. The effort is to keep the feature extraction system as general as possible so that the data input can be from any STEP schema. The extracted features will be saved in the STEP AP224 format so that this can be further interfaced with CAPP and CAM systems. The AP224 STEP file not only contains feature information but also consists of all the geometry information already existing in the input file and can thus be visualized directly using a STEP supporting CAD system.

4. Characteristics and properties of features relevant for the extraction process

The STEP-based feature extraction algorithm draws significantly from the procedures developed by Brun [5] and Qamhiyah [24]. The algorithm is currently developed for prismatic solids (where the part envelope consists of plane surfaces), produced by milling operations, and that contain elementary shapes such as plane surfaces, cylindrical surfaces, etc. However, representations of surfaces in the form of non-uniform rational B-splines (NURBS) is accommodated in the algorithm. Approximately 40 to 50% of the parts produced in the industry using milling can be represented using the above type of representation [13].

[2] PDES was the pre-cursor to STEP in the US.

4.1. Basic shape unit for purpose of feature extraction

The *advanced brep shape representation* (ABSR) is used as the functional unit for the feature extraction process. For more information on the various other forms of shape representations, the reader is directed to Ref. [4]. The ABSR consists of a *set of representation items*. This set contains an entity of the type *manifold solid brep*. The *manifold solid brep* entity or its subtype namely *brep with voids* contains the complete definition of the geometry and topology of the solid in the B-Rep format. The *manifold solid brep* is a finite, arcwise connected volume, bounded by one or more surfaces, each of which is a connected, oriented, finite, closed manifold. There are no restrictions on the number of holes or voids within the volume. It contains an outer *closed shell* which defines the outer extent of the solid. If the solid contains any voids, they are defined using *brep with voids*, with the void formed by a *set of oriented closed shells*. Void shells are contained completely within the outer shell. The *closed shell* which is a type of *connected face set* is a set of arcwise connected surfaces known as faces. The working unit in an ABSR is a subtype of *face* called *advanced face*.

The face is the topological entity of dimensionality 2, corresponding to a piece of surface bounded by *loops*. Each face is represented by its bounding loops which are defined as *face bounds*. Each face must have at least one bound, and if there is more than one bound, they must be distinct. One of the bounds can be classified as the *face outer bound* and this defines the outer loop of the face. A geometric surface can be associated with each of the faces forming the solid. This can be done by using the *advanced face* entity or the *face surface* entity. A Boolean flag is used to signify whether the loop direction is oriented in accordance or opposed to the surface normal. Each *advanced face* of the solid shares an edge with exactly one other face to form a closed set of faces. The *loops* bounding the surface can be constructed by a single vertex, or by stringing together connected edges, or by linear segments beginning and ending at the same vertex. The loop formed by a single vertex is called *vertex loop*, this typically represents an apex of a cone where the entire surface is degenerated down to a single point and is bound by a single vertex. A loop which bounds the face by a collection of edges is called *edge loop*. It is a path in which the start and end vertices are the same to close the loop around the face. The loop formed by a group of straight line segments is called *polyloop*, in which a planar region is bounded by straight lines. This type of loop is primarily found in faceted B-Rep models for solids.

The *edge loop* consists of a *list of oriented edges*, which are constructed from another edge and contain a Boolean flag to orient them so as to form a closed bound around the face. An *edge* is a topological entity bounded by two vertices not including them. The edge is oriented from the start to the end vertex, this orientation can be reversed by the Boolean flag for the oriented edge. The edge can have an underlying geometry associated with it. This can be modeled using the *edge curve* subtype which has the geometry fully defined for an edge. A Boolean flag is attached to this entity indicating whether the direction of the edge and the underlying curve are the same or opposed to each other.

The vertex is the topological construct corresponding to a point. A single vertex can be shared by many edges and subsequently by many faces. A vertex is represented as a *vertex point* which can be represented as a *cartesian point* in three-dimensional space.

The B-Rep solid, thus, formed from faces, edges and vertices, satisfies Euler–Poincaré formula [28] for closed solids. If v, e, f, s, r and h are the number of vertices, edges, faces, shells (surfaces), interior loops in the faces (rings) and through holes in the solid respectively, then according to the formula:

$$v - e + f = 2(s - h) + r \quad (1)$$

4.2. Classification of edges

The edges are classified as either convex or concave depending on the angle between the faces sharing this edge. This classification of edges is done using the underlying surface geometry of the faces sharing this edge. In a B-Rep representation of a solid, the edges are represented in such a way that if the edges are traversed in the direction of the loop

bounding the face, the material of the solid always lies to the left. Due to this the outer loops bounding the face are in the counterclockwise direction and the inner loops are in the clockwise direction [5]. This is shown in Fig. 4.

Let e_k be the edge shared between two faces F_i and F_j. Then, convexity or concavity of the edge is found as follows.

(i) First the cross (vector) product (c) of the face normals of the faces is calculated as in Eq. (2), where f_i and f_j are the normals to the surfaces F_i and F_j, respectively. Note that the order of the surface normals is from right to left of the edge view perspective.

$$c = f_i \times f_j \tag{2}$$

The direction of c will be parallel to the direction of e_k because of the face vector geometry and the right hand rule [19]. If the direction of c and e_k is the same, the edge formed is a concave edge, otherwise it is a convex edge. (ii) The direction of c is determined by calculating the dot (scalar) product (d) of the resulting cross product c with the directional vector of the common edge e_k as in Eq. (3), where $\|c\|$ is the magnitude of c, $\|e_k\|$ is the magnitude of e_k, and θ is the angle between c and e_k, which in this case would either be 0° or 180°.

$$d = c \cdot e_k = \|c\|\|e_k\|\cos\theta \tag{3}$$

If the value of d is positive, it implies that θ is 0 and the edge is concave with respect to F_1 and F_2, otherwise the edge is convex. Fig. 5 illustrates this procedure for establishing the convexity or concavity of the edge. The edge e_1 is directed from vertex 1 to vertex 2. The right face is F_1 and the left face is F_2. Using the right hand rule the vector c is determined

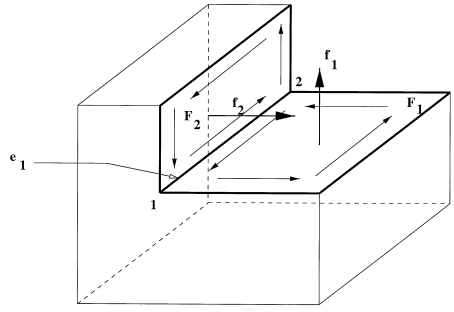


Fig. 5. Part showing the edge directions and the surface normals.

which is orthogonal to the right and left face directions. The value of the dot product of c and e_1 is positive, indicating they are in the same direction. Therefore, e_1 is a concave edge.

4.3. Classification of form-features

This section gives an overview of the classification of the form-features. The classification is based on the effect of the features in changing the basic shape of the solid [5,24]. The form features are classified into four basic types:

1. void features,
2. internal form features,
3. external form features, and
4. connectivity modifying form features.

Form features belonging to the type *void features* form voids within the solid. Voids are areas which are totally enclosed within the outer shell of the solid. Voids are represented as *void shells* within the solid and the solid is represented as *brep with voids*. Fig. 6 shows an example of this type of feature.

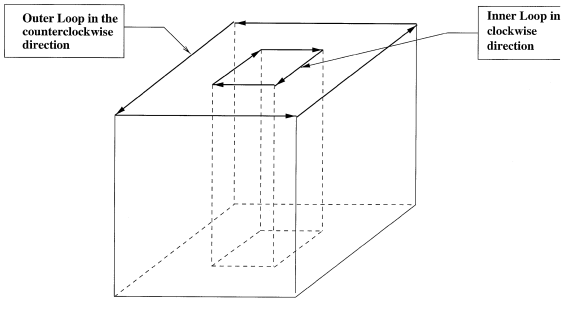


Fig. 4. Edge directions in the outer and inner loop.

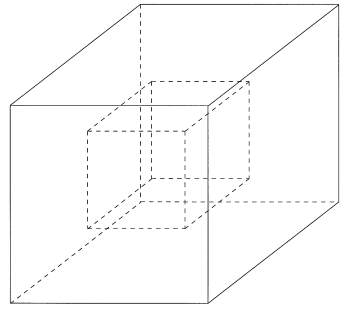


Fig. 6. Example of feature of type void.

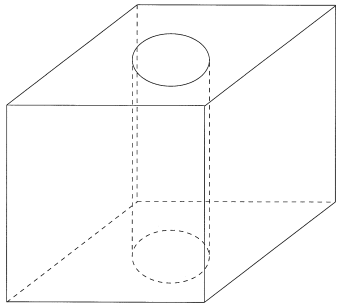


Fig. 7. Example of feature of type through.

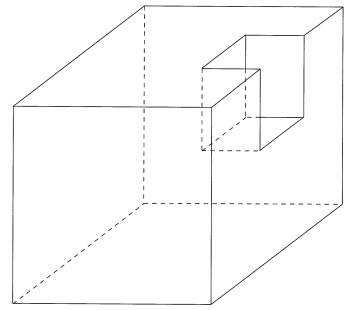


Fig. 9. Example of feature of type external edge modifying.

The form-features of the type *internal* are the ones which modify the internal shape of one or more surfaces. Internal features are classified into two sub-types: (i) *through features* which go right through the solid and create a path from one outer face of the solid to another outer face, e.g., through-holes and through-pockets, Fig. 7 shows a solid with this type of feature, and (ii) *blind features* which alter the internal shape of a surface but do not penetrate through the solid; these types of features begin at one of the outer surfaces of the solid and end inside the solid, e.g., blind-holes and blind-pockets. Fig. 8 shows a solid containing this type of feature.

Form features belonging to the type *external* modify the external shape of the solid, e.g., steps, slots, external pockets, etc. External form features are further classified in the three sub-types: (i) *Edge modifying*: these features modify an edge of a face or between a pair of faces to create a feature. An example of this type of feature is an external pocket formed between two faces. Fig. 9 shows this type of feature. (ii) *Face modifying*: these features are probably the most widely found features and are typical in engineering applications. These features split up a single face to create a feature on that face. However, these features may not be restricted to a single surface only. Modification of one or more of the adjoining faces may result depending on the type of feature. Examples of this type of feature are steps, slots, etc. Figs. 10 and 11 show examples of face modifying features. (iii) *Vertex modifying*: these types of feature modify the vertex between adjacent faces, e.g., a cutout would fall under this category. Fig. 12 shows a solid with a vertex cutout.

Connectivity modifying form features are those which alter the ''joint'' between the various faces of the part, e.g., chamfers and rounded edges. These form-features are usually created as last or finishing operations during the manufacturing process. Fig. 13 shows a part with this type of form feature.

The order in which the above form features have been classified depends on the general manufacturing steps that are followed during the manufacturing process. Generally, features of the type ''void'' would be produced first. These features are generally

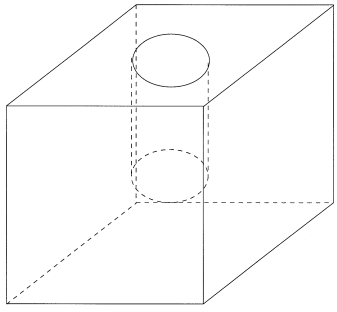


Fig. 8. Example of feature of type blind.

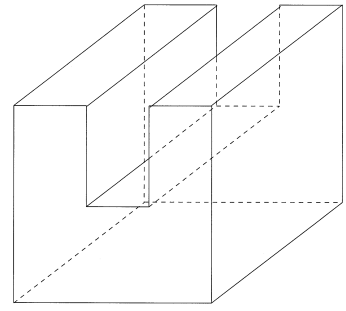


Fig. 10. Example of feature of type face modifying.

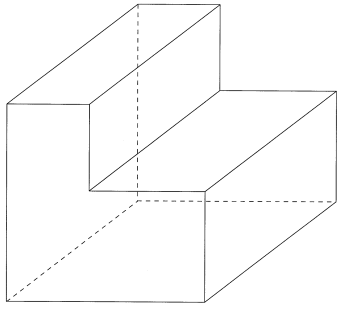


Fig. 11. Example of feature of type face modifying.

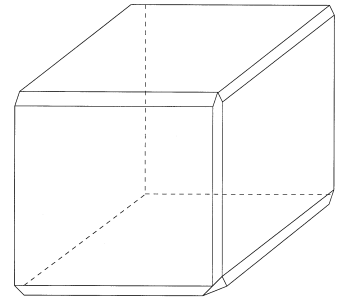


Fig. 13. Example of feature of type connectivity modifying.

created by the primary manufacturing processes like casting and molding. The internal and external features are then created by the secondary manufacturing processes like milling and turning. Finally, depending on the tolerances and other design requirements, connectivity modifying features are created at the last stage of manufacturing using tertiary processes like grinding and deburring. These processes do not involve considerable amount of material removal but require only the alteration of the joining edge between two faces.

4.4. Properties of the feature types

The properties of some of the features defined in Section 4.3 are detailed in this section. The properties of the feature of type void have not been identified as the sufficient condition for the features of type void is the presence of a *brep with void* type entity in the STEP file. A void shell or the inner shell forms this feature.The description of the properties of the remaining features are based on geometric reasoning and the understanding of the manufacturing processes by the authors.

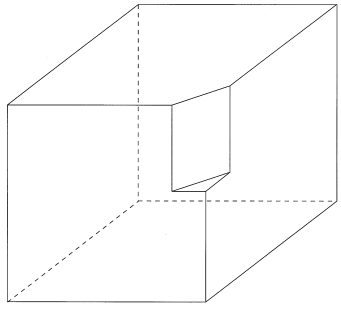


Fig. 12. Example of feature of type vertex modifying.

4.4.1. Type internal

As explained in Section 4.3, these features form internal elements in the solid. The internal elements could pass right through the solid in which case the feature would be identified as a through feature or they penetrate only to some distance into the solid in which case a blind feature, would be formed.

The presence of these features is identified by the existence of inner loops on the faces. As explained previously, a face of the solid is bounded by an outer loop of edges (or a vertex in some special cases) and by one or more inner loops if there are any internal features on the face. The edges of the inner loops are shared between the faces forming the outer cover of the inner faces. The type of underlying curve geometry for the edges forming the inner loop and the surface geometry for the faces forming the inner elements provides a clue as to the type of feature that is formed.

For example, the existence of an inner loop with two edge elements bounding the loop with cylindrical surfaces or NURBS surfaces of cylindrical nature that form the inner elements asserts that the internal feature is a *hole*. If on the other hand the inner edge were bounded by four edges and the edge geometry was defined by straight edges, the feature that is formed is an *internal-pocket*. The presence of curved edges would indicate that the feature being formed is a rounded or circular pocket. These types of feature are common in machined parts and are produced by milling operations, using an end mill to cut out the required profile from the work-piece.

The next important thing is to determine whether the internal feature is blind or through. Here, the property of the STEP file to associate the geometry to the faces proves to be of great importance. If the

geometry of the face forming the inner element is shared by more than one outer faces, the feature would be of the type *through feature* (Fig. 7). On the other hand, if this were not the case and the surface is shared by only one outer element, the feature would be of the type *blind* (Fig. 8). Blind features also contain a ''base face'' which has concave loops. The concept of loop convexity and concavity is explained in Section 4.2. If this ''base face'' has an internal loop, it would indicate a stepped type of internal feature. Typical examples of these would be a counter-sunk or counter-bored hole or pocket. These properties cover most of the cases of milled features that are likely to be seen in a typical manufacturing setup.

4.4.2. Type external

As explained in Section 4.3, these features alter the external shape of the solid, and are further classified into three sub-types: (i) those that modify an edge between two faces, called *edge modifying*; (ii) those that modify a single face of the solid, called *face modifying*; and (iii) those that modify a vertex between adjacent faces, called *vertex modifying*. Each of these have a different set of properties which can be used to uniquely identify these from the B-Rep structure of the solid.

- *Edge modifying*: These features are characterized by the presence of a pair of faces sharing more than one edge. Fig. 14 shows in bold the edges forming the two faces F_1 and F_2. The two faces share the edges e_1 and e_2. These faces share more than one edge and this indicates the presence of an edge modifying feature between the two faces F_1 and F_2. The above is only the necessary condition for the presence of an edge modifying feature. The sufficient condition would be the determination of the convexity–concavity of the edge of faces F_3 and F_4. These two faces must have exactly three concave edges and only one convex edge which is the edge shared between these faces and the outer faces of the solid namely F_1 and F_2. If the above necessary and sufficient conditions hold true then there exists a feature of the type *edge modifying* in the solid. Typical examples of this type of feature are blind pockets and blind slots (which are notably produced using the milling operation).

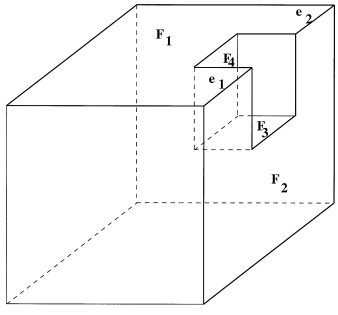


Fig. 14. Feature of the type edge modifying.

- *Face modifying*: These features partition at least one face of the solid to generate new faces, or eliminate at least a single edge. If the face is changed to the extent of being partitioned into several new surfaces, its identity in the B-Rep is lost completely. Due to this reason, these features cannot be identified by tracing their effect on the surface underlying the face, edges or vertices. These features can be isolated by determining the convexity or concavity of the edges forming the loops bounding the face and also that of the neighboring faces. To extract a feature of this type the edges forming the loops are analyzed, and depending on the number of concave edges, the features are classified. Given a face containing four edges (of a prismatic solid) out of which two edges are concave and the remaining two edges are convex, there is a very high probability that this surface is a part of a slot feature. The neighborhood of this face has to also be looked at before any such decision can be made. If the faces sharing the concave edge have only one concave edge, it gives more credibility to the fact that there exists a slot in the solid. If on the other hand the given face has only one concave edge, and also the neighboring face which shares the concave edge has only one concave edge, a step exists in the solid. Figs. 10 and 11 show examples of face modifying features.

More complex features need further probing. Not only are the immediate neighboring faces examined but the neighbors of these neighboring faces and so on are recursively analyzed till the outer face of the solid is reached. If there are more than one faces which must be traversed to reach the outer face of the solid, there is a complex feature present in the solid. A T-Slot would be a typical example of such a complex feature. At this time, the algorithm is only

capable of classifying this presence as a complex features. The exact identification of the type of face modifying form feature is not done at this time. This is because the interpretation of the complex features would be dependent on the manufacturing setup that exists within the company, or it would be dependent on the designer and manufacturer of the product. This should however not be considered as a deficiency of the feature extraction algorithm. The algorithm can later be expanded with setup information to accommodate more complex features which may exist at the manufacturing site.

- *Vertex modifying*: These features modify or eliminate a single vertex between adjacent faces. In most of the prismatic solids, a triplet of faces share a common vertex. This vertex is the intersection point of the three edges bounding the three faces. The elimination of the vertex leads to the creation of a *vertex modifying* feature. Another condition which can be regarded as the sufficient condition for establishing the vertex modifying form feature is that the faces formed as a result of eliminating the vertex must have at least one concave edge. Existence of this type of feature can be found by first checking for adjacent faces. From this adjacency information, the B-Rep vertex data is checked to determine whether these faces intersect at a vertex. Once this has been established, the convexity and concavity of the edges can be found to satisfy the necessary and sufficient conditions for this type of form feature. Cutouts generally fall under this category of form-features (see Fig. 12).

4.4.3. Connectivity modifying form features

So far, all the above form features require the existence of concave edges. The form features of the type *connectivity modifying* do not require this condition. These form features are generally created in the final stages of manufacturing and can be called finishing features. These features are very important during the assembly stage of the part to ensure the correct fit of the various parts into one another. Chamfers and rounded-edges are some of the examples of this type of form feature. Features of the connectivity can be identified by looking at the edges which form these features. The angle made by the surface normal to the face is compared to the angles that typically exist for these types of features or the underlying geometry of the face is examined at the joint to determine what type of joint exists between the faces. If a cylindrical surface joins two surfaces, the faces are joined by a rounded-edge. The presence of a plane surface at the intersection suggests that a chamfer is used for smoothing the edges and removing the burrs which may be produced during manufacturing.

5. Implementation of the feature extraction process

The feature extraction system is implemented using object-oriented modeling principles and C + + programming language. Fig. 15 presents a flow chart describing the feature extraction process. The feature extraction module takes as input the STEP file defining the geometry and topology of the part. The input module is currently developed to accept AP202 file

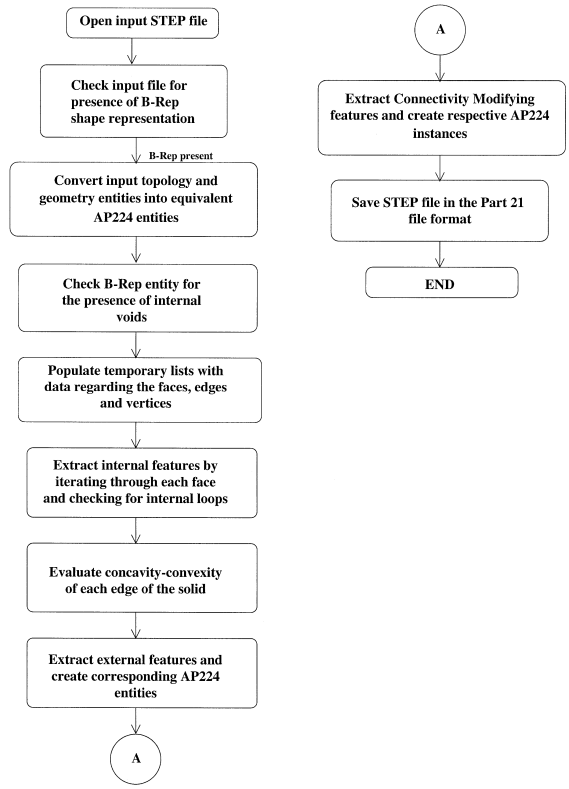


Fig. 15. Flow chart describing the feature extraction system.

and can be interfaced with the IGES to AP202 translation module [3,4]. However, the feature extraction module can be modified to accept AP203 files as input; this would require rebuilding the input module and linking with the AP203 schema definition. The feature extraction module uses an object-oriented database called ROSE [15]. The EXPRESS schema definitions of AP202 and AP224 are converted into C++ classes using the ''express2c++'' utility provided by ST-Developer [34]. The C++ classes are compiled to create AP202 and AP224 libraries which are linked to the main module.

After the module ''reads-in'' the STEP file, it goes through the file to check whether the file has the required type of shape representation, namely the ABSR. If the shape representation does not exist in the file, it implies that the file does not have topology information which is essential for extraction of features. At this point the feature extraction process stops, informing the user that the file cannot be processed further. On the other hand, if there exists an ABSR entity in the file, the features can be extracted from the given topology information within the file and the extraction module continues further.

The first step after obtaining the correct type of file input involves the conversion of the AP202 topology and geometry into the respective AP224 entities. As has been previously described, an AP224 file contains geometry, topology and form feature data. The geometry and topology entities are derived from the integrated resources and are the same in both these schema. The conversion routine does a one-to-one mapping between the input and the output entities. This conversion is necessary because currently a STEP file can have entities which are specific to only a particular schema. Fig. 16 shows the default header of a STEP file generated by ST-Developer [34]. The header contains information regarding the description of the file, the name of the file, the date and time stamp, author name, organization name generating the file and other information. The schema to which the file belongs is also defined. In this particular case, the file schema is ''Associative Draughting'' meaning that the file is an AP202 file. The conversion process is written as a separate module linked to the main module through a single function call. The advantage of this is that, if a later date the issue of the schema inter-operability gets

```
ISO-10303-21;
HEADER;
/* Exchange file generated using ST-DEVELOPER v1.5 */

FILE_DESCRIPTION(
/* description */ (''),
/* implementation_level */ '2;1');

FILE_NAME(
/* name */ 'part',
/* time_stamp */ '1997-04-04T12:46:46-08:00',
/* author */ (''),
/* organization */ (''),
/* preprocessor_version */ 'ST-DEVELOPER v1.5',
/* originating_system */ '',
/* authorisation */ '');

FILE_SCHEMA (('ASSOCIATIVE_DRAUGHTING'));
ENDSEC;
```

Fig. 16. STEP header.

resolved and the STEP file could have entities belonging to several different schemata, the feature extraction module can be updated easily by deleting the conversion module from the main program.

After the entities have been converted, the module goes on to extract the feature information from the B-Rep information of the part in the AP202 file. To facilitate the extraction process, two lists have been created to maintain temporary storage of the faces which share a common edges and vertices. The above information linking faces, edges and vertices is present in the B-Rep tree. However, it is very cumbersome and memory intensive to parse the tree multiple number of times to extract out adjacency information. Thus, even though there is some data duplication, it results in saving in computation time. The lists are created using the ability provided by ST-Developer that allows creation of non-persistent aggregate types from C++ classes. The first list, called *edge face list*, consists of a list of a pair of faces and the edge(s) that they share. To populate this list the STEP file is scanned for edges and faces and all the edges are added to the *edge face list*. The edges that bound each face are traversed one after another and the respective face is added to the correct position in the *edge face list*.

The second list, called *vertex face list*, consists of a vertex and the faces sharing this vertex. Since feature extraction is done from prismatic solids, in most of the cases three faces come together at a single vertex. The general method for the population

of this list is similar to that of the previous list, whereas it is more in-depth since it involves traversing the loops bounding the face right down to the vertex level. To populate this list, all the vertices and faces in the STEP file are collected and each vertex is added to the list. The loops of each of the faces are traversed to get the edge information, the edges are traversed to get the vertex information and the faces are associated to the correct vertex in the *vertex face list*.

Once the lists have been populated the extraction of features begins. The first features that are extracted are the *void features*. Presence of these features can be found by the presence of the *brep with voids* entity in the STEP file. The *brep with voids* contains a set of *oriented closed shells*, each of which contains a set of *connected face sets*. The surface underlying each of these faces is checked to establish the type of void. If all the faces are plane surfaces instantiated as *plane* entities or *rational b_spline surface* entities, the *void feature* forms a rectangular void. The *rational b_spline surface* is analyzed by using either the information attribute called *b_spline surface form* which specifies what type of surface is being defined or by using the degree of the knots in the *u* and the *v* direction and the list of control points of the *b_spline surface*. The explanation on the analysis of *b_spline surfaces* can be found in Ref. [4]. After the presence of a rectangular void is established an instance of a *rectangular closed profile* is created. As the name suggests this entity is used to define the presence of a rectangular profile in the solid. The required attributes of this entity are populated. If the faces forming the void are cylindrical surfaces or spherical surfaces, a feature instance called *circular closed profile* is created and its required attributes are populated.

The next step in the feature extraction process involves the recognition of internal features. As described in Section 4.4.1, these features are extracted by the presence of internal loops for a particular face. The faces forming the outer shell of the solid are parsed one after another. The extraction of the internal features is done in two passes. In the first pass, if any face has an internal loop, an instance of a class called *internal* is created. The edges forming the internal loop of the face are traversed and from the *edge face list* the adjacent face is found. The surface underlying this second face is added to a temporary list to trace the surfaces that have already been traversed. If a surface which is already in this temporary list is encountered a second time, the internal *through* feature is present. At this time (i.e., when the faces are encountered a second time), the faces are deleted from the temporary list and the instance of a through feature is created. The internal surfaces are analyzed to find out what type of internal feature is formed. If the surface is a plane surface or a NURB representation of a plane surface, an instance of a *pocket* is created. The *name* attribute of the pocket is set to ''rectangular''. An instance of the *pocket bottom* entity is also created with its name attribute set to ''through''. If the surfaces forming the internal feature are a combination of plane and cylindrical surfaces, the pocket formed is named ''non-rectangular'', with the bottom condition remaining the same as the previous case. If the surfaces forming the internal element are all cylindrical surfaces, an instance of a *round hole* is created with a ''through'' bottom condition.

After all the faces have been parsed for the internal features the temporary list is checked to see if there are any remaining faces. The presence of faces in the temporary list indicates the presence of a ''non-through'' feature. Each of the remaining faces in the temporary list is examined and depending on the underlying surfaces the feature instances are created. The reasoning behind the type of feature chosen is the same as described earlier, only the bottom condition varies with its name attribute describing the type of bottom that exists for the feature.

The next phase involves the extraction of external features. Before the feature extraction algorithm proceeds, the edges are classified into convex or concave using the edge classification algorithm described in Section 4.2. Each element of the *edge face list* contains an enumeration type attribute which is used to describe the edge type. The classification algorithm first requires the determination of the surface normal information. If the surface is an instance of a *plane* surface, the surface normal is determined by using the *axis2_placement_3d* entity which determines the position of the surface. The *axis2_placement_3d* entity consists of a points through which the surface normal passes and two direction entities which define the local *X*-axis and

Z-axis for the plane. In STEP, the *plane* is defined in the local *X* and *Y* axes, hence, the surface normal is the local *Z*-axis direction.

In a more complex case where the surface is a NURB surface, the surface normal is calculated using the DT-NURBS Generalized Spline Geometry Library [8]. DT-NURBS is a FORTRAN library developed by Boeing for the Navy and is available as freeware. The library takes the cross product of the two first-order partial derivatives of the spline surface to find the normal to the surface. The output of the DT-NURBS routine is normalized. To use this surface normal for the concavity–convexity calculation, the normal must be directed out of the solid. This can be checked using the Boolean flag of the *advanced face* entity which is ''true'' if the parameterization of the surface is in the same direction as that of the face and ''false'' otherwise. Depending on the Boolean flag the direction of the normal is directly used for further calculation or reversed before it is used.

After determining the edge convexity–concavity, the *edge face list* is scanned to check whether any of the faces share more than one edge. If there exists such a pair of faces sharing more than one edge, there is a possibility of a pocket existing between the two faces. Using the *edge face list*, the faces that form the pocket are determined and checked for the necessary and sufficient conditions for the faces to form a pocket. If all the conditions are satisfied an instance of a *pocket* is created and an instance of *pocket bottom* is created to define the pocket completely.

The B-Rep data is next evaluated for the presence of *face modifying* features. These features are determined using the information of edge concavity–convexity determined previously and the criterion for the determination of these types of features. If the presence of a ''slot'' feature emerges after the procedure is run, an instance of a *slot* is created and the required attributes are populated. There is no direct defined feature for ''step'' in AP224; if a ''step'' is present in the solid, an instance of a *flat face* is created. The removal direction for the flat face is defined parallel to the flat face of the ''step''.

Next, the *vertex modifying* features are extracted according to the algorithm described earlier. The *edge face list* is used to find adjacent faces and the *vertex face list* is used to check whether these adjacent faces have a common vertex or meeting point. The absence of a vertex is tested further with the concavity constraint. A vertex feature does not have a one-to-one mapping in AP224, and instead it has been mapped to a *removal volume* entity. *Removal volume* is a sub-type of a *machining feature* entity and is used for representing generalized features. An instance of a *removal volume* is created and the attributes are populated.

Finally, the *connectivity feature* set is extracted. The *connectivity features* are mapped into AP224 *transition features* which include *chamfer*, *edge round* and *fillet*. The surface normal information derived earlier for calculating the edge concavity–convexity is used to determine the angle between the intersecting faces. Depending on the type of *connectivity feature* between the faces, determined by the algorithm in Section 4.4.3, an instance of a *chamfer* or *edge round* is created and its attributes are populated.

The feature extraction process is now complete and all the features that are present in the part (and are within the scope of this work) are extracted at this point. The AP224 STEP file is saved in the Part 21 file format. This file is computer interpretable and can be exchanged between various CAD systems or can be input to an automated process planning system which creates a manufacturing process plan from the feature data in the file. The working of the above feature extraction process is demonstrated with the help of two examples in Section 6.

6. Demonstration of the feature extraction process with examples

Fig. 17 shows a test part created using Pro/Engineer for the testing of the feature extraction system. This file is used as input to the feature extraction system. The feature extraction system first extracts out the blind hole and creates an instance of the same in the AP224 STEP file. Next, the external ''steps'' are extracted by the system. All the entities are saved in an AP224 STEP file. The extraction process takes less than 10 s of CPU time on a Silicon Graphic O_2 running IRIX version 6.3 to complete execution.

Fig. 18 shows a bull pin block which is a more complicated solid than the previous example and is used to demonstrate the effectiveness of the feature extraction system. The STEP AP202 file containing the B-Rep of the bull pin block serves as input to the feature extraction system. The part contains 36 faces, 96 edges and 62 vertices. It can be easily seen that the bull pin block satisfies the Euler–Poincaré formula (Eq. (1)) for a valid solid. The feature extraction system first checks whether there are any internal voids in the solid; in this particular case, there are none and hence no void features are extracted. Next, the system checks for internal features. The bull pin block contains two round holes and one through pocket. The through pocket is non-rectangular in nature identified by the presence of cylindrical and plane surfaces forming the walls of the pocket. The round holes are bound on the outside by two faces, both of which are cylindrical surfaces. Next, the system establishes the concavity–convexity of all the edges by calculating the surface normals and their subsequent cross-products. The surfaces of the bull pin block are instantiated as NURB surfaces and, hence, the DT-NURBS FORTRAN library is used to calculate the surface normals. The feature extraction system next extracts the external features. The system extracts out two end cutouts that are present in the solid and one blind pocket which is formed between the triangular faces of solid. The entire extraction process take less than 15 seconds to complete.

From the limited examples attempted, we observe that the run time is acceptable for simple parts (i.e., parts with a reasonable number of faces, edges, vertices of the B-Rep). We hope to acquire larger STEP files which can then be tested and do a comparative study of the run times and perform optimizations as necessary. However, that activity is outside the scope of the current paper.

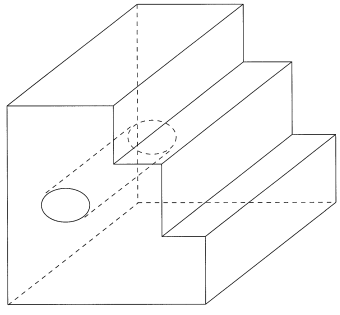


Fig. 17. Test part for the feature extraction process.

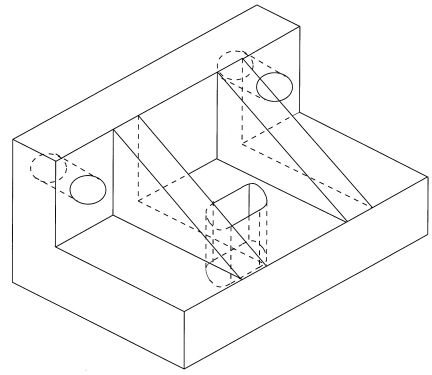


Fig. 18. Bull pin block.

7. Conclusions and recommendations

This work is aimed at addressing the data exchange and sharing needs of Agile Manufacturing. The contribution of the research is the development of a feature extraction system which takes a STEP file as input and produces a form-feature STEP file. This STEP file can be exchanged between various companies and can serve as input to further downstream activities such as process planning, scheduling and material requirement planning (MRP).

The primary objective of this work is to develop a feature extraction system that can store the feature data in a computer interpretable format and which can be transmitted between various locations. The features are extracted from the design data for the particular product and this would force the designer to seek input from the manufacturing engineering and the manufacturing engineering to provide input to the design engineer, so that the product developed can be directly manufactured and the number of design changes made as a result of manufacturing constraints is restricted to a minimum. The feature extraction system developed is aimed at overcoming the shortcomings of the design by feature approach

which is limited by the number of features in the pre-defined library of features. The extracted form features are saved in the STEP Part 21 file format which can be read by some of the currently available CAD systems and exchanged between various systems over the Internet or by electronic mail. The output file generated by the feature extraction system contains all the geometry and topology information required to completely define the solid and also form feature information. The form feature information gives higher level meaning to the group the faces forming the solid and can be used for CAPP and CAM.

The feature extraction mechanism has currently been developed for solids which can be manufactured by milling process. AP224, as mentioned previously, covers products that can be manufactured by milling and turning. Thus, the current work should be expanded to extract features from turned parts. Also, currently, the APs supported are AP202 and AP224. The feature extraction system could be expanded to support AP214 which is an AP pushed forward by the automotive companies and is used for representing geometry, topology and assembly information. The AP224 STEP form features can be used for generative process planning. Another possible extension to this research could be the use of the form-feature information along with the geometry and topology information for Group Technology (GT) coding of the solid. The GT code can be used for identifying part families of similar parts. This can be further used for generating alternative process plans and cellular manufacturing types of algorithms.

Another possible future research direction involves the creation of an integrated manufacturing environment in which a part would be designed using a standard available CAD system. The design would be saved in the STEP format. The feature extraction system would be used to extract out form features. Thus, the generated AP224 file could be used as input to an automated process planning system which would generate the manufacturing process plan and an automatic NC coding system which would generate the NC program for manufacturing the product. The NC program can be used for actual manufacturing of the product or for a Rapid Prototyping type of operation in a research-based environment.

Acknowledgements

Rakesh Nagi acknowledges the support of the National Science Foundation career grant DMI-9624309. We acknowledge STEP Tools, Rensselaer Technology Park, Troy, NY, for a software grant of ST-Developer. The suggestions of the editor and two referees were very helpful in improving the paper.

References

[1] P.V. Avasarala, E.L. Stern, Form feature-based design of turned parts, ASME Proceedings of the 1994 ASME Design Technical Conferences, Sept. 11–14, 1994, Minneapolis, DE-Vol. 73, 1994, pp. 135–143.

[2] K.S. Barber, A feature-based CAD representation enabling case-based planning across multiple manufacturing applications, IEEE International Conference on Systems, Man and Cybernetics, 1994, pp. 142–147.

[3] M.P. Bhandarkar, B. Downie, M. Hardwick, R. Nagi, Migration from IGES to STEP: one-to-one translation of IGES drawing to STEP drafting data, Computers in Industry, 2000, in press.

[4] M.P. Bhandarkar, Satisfying information needs in Agile Manufacturing through translation and feature extraction into STEP product data models, MS Thesis, State University of New York at Buffalo, 1997.

[5] J. Brun, From characteristic shapes to form features, Proceedings of IFIP International Conference on Feature Modeling and Recognition In Advanced CAD/CAM Systems, Valenciennes, France, Vol. 1, 1994, pp. 315–326.

[6] J.K. Coles, R.H. Crawford, K.L. Wood, Form feature recognition using base volume decomposition, ASME, International Computers in Engineering Conference and Exhibition, Minneapolis, Sept. 11–14, 1994, pp. 281–297.

[7] J. Dong, H. Parsaei, Manufacturing feature extraction and recognition, Computers and Industrial Engineering 25 (1993) 141–144.

[8] DT_NURBS Generalized Spline Geometry Library, URL, http://dtnet33-199.dt.navy.mil/.

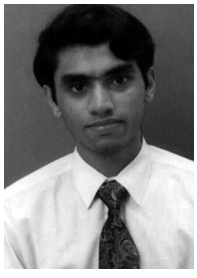
[9] J.C.E. Ferreira, S. Hinduja, Convex hull based feature recognition method for 2.5D components, Computer Aided Design 22 (1) (1992) 41–49.

[10] S. Finger, S.A. Safier, Representing and recognizing features in mechanical designs, Design Theory and Methodology — DTM '90, Presented at the 1990 ASME Design Technical Conferences — 2nd International Conference on Design Theory and Methodology, Chicago, 1990, pp. 19–26.

[11] P. Gavankar, M.R. Henderson, Graph-based extraction of protrusions and depressions from boundary representations, Computer Aided Design 22 (7) (1989) 442–450.

[12] R. Ghad, F. Prinz, Recognition of geometric forms using differential depth filter, Computer Aided Design 24 (11) (1992) 538–598.

[13] C. Gilman, The use of the PDES form feature information model as the primary representation for a form feature-based design environment, Master's Thesis, Rensselaer Polytechnic Institute, Troy, NY, 1990.
[14] Z. Gu, Y.F. Zhang, A.Y.C. Nee, Generic form feature recognition and operation selection using connectionist modeling, Journal of Intelligent Manufacturing 6 (1995) 263–273.
[15] M. Hardwick and the RPI DICE team, ROSE 3.0 User Manual, Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, NY, 1991.
[16] S. Joshi, T.C. Chang, Graph-based heuristics for recognition of machined features, Computer Aided Design 20 (1988) 58–66.
[17] M.Y. Jung, K.H. Lee, A CAD/CAPP interface for complex rotationally symmetric parts, International Journal of Production Research 34 (1) (1996) 227–251.
[18] Y.S. Kim, Recognition of form features using convex decomposition, Computer Aided Design 24 (9) (1992) .
[19] E. Keryszig, Advanced Engineering Mathematics, 5th edn., Wiley, India, 1995.
[20] S.C. Liu, M. Gonzalez, J.-G. Chen, Development of an automatic part feature extraction and classification system taking CAD data as input, Computers in Industry 29 (1996) 137–150.
[21] National Institute of Science and Technology, Index of STEP Parts, URL, http://www.nist.gov/step/parts.
[22] F. Pariente, Y.S. Kim, Incremental and localized update of convex decomposition used for form feature recognition, International Journal of Production Research 28 (8) (1996) 589–602.
[23] Part 224, Mechanical product definition for process planning using form features, Document ISO TC184/WG3 N264(T7) National Institute of Standards and Technology, USA, 1996.
[24] A.Z. Qamhiyah, Form Feature Extraction and Coding for Design by Features, PhD Dissertation, University of Toronto, 1996.
[25] A.Z. Qamhiyah, R.D. Venter, B. Benhabib, Geometric reasoning for extraction of form features, Computer Aided Design 28 (11) (1996) 887–903.
[26] H. Sakurai, C.W. Chin, Definition and recognition of volume features for process planning, in: J.J. Shah, M. Mäntylä, D.S. Nau (Eds.), Advances in Feature-Based Manufacturing, Chap. 4, 1994, pp. 65–79.
[27] D.A. Schenck, P. Wilson, Information Modeling: The EXPRESS Way, Oxford Univ. Press, USA, 1994.
[28] J. Shah, M. Mäntylä, Parametric and Feature-based CAD/CAM, Wiley, USA, 1995.
[29] J. Shah, M. Mäntylä, D. Nau, Manufacturing Research and Technology: 20. Advances in Feature-Based Manufacturing, Elsevier, The Netherlands, 1994.
[30] J.J. Shah, A. Mathew, Experimental investigation of the STEP form-feature information model, Computer Aided Design 23 (4) (1991) 282–296.
[31] J. Shah, M. Rogers, Functional requirements and conceptual design of the feature-based modeling system, Computer Aided Engineering Journal 5 (1), 9–15.
[32] M.E. Ssemakula, A. Satsangi, Application of PDES to CAD/CAPP integration, Computers and Industrial Engineering 18 (4) (1990) 435–444.
[33] STEP on a page, URL, http://www.mel.nist.gov/sc5/soap/.
[34] STEP Tools, URL, http://www.steptools.com.
[35] C.J. Su, T.L. Sun, C.N. Wu, R.J. Mayer, An integrated form-feature-based design system for manufacturing, Journal of Intelligent Manufacturing 6 (1995) 277–290.
[36] K. Al-Timimi, J. MacKrell, STEP: Towards Open Systems, CIMdata, ISBN 1-889760-00-5, 1996.
[37] J.H. Vandenbrande, A.A.G. Requicha, Spatial reasoning for automatic recognition of interacting form features, ASME International Computers in Engineering Conference and Exposition, Boston, Aug. 5–9, 1990, pp. 251–256.
[38] T.C. Woo, Feature extraction by volume decomposition, Proc. Conf. CAD/CAM Technology in Mechanical Engineering, Cambridge, MA, USA, 1982.
[39] J.K. Wu, T.H. Liu, G.W. Fisher, PDES/STEP-based information model for CAE and CAM integration, International Journal of Systems Automation: Research and Applications 2 (1992) 375–394.
[40] Y. Yue, J.L. Murray, Validation, workpiece selection and clamping of complex 2.5D components, in: J.J. Shah, M. Mäntylä, D.S. Nau (Eds.), Advances in Feature-Based Manufacturing, Chap. 9, 1994, pp. 185–213.



Rakesh Nagi is an Associate Professor of Industrial Engineering at the State University of New York, Buffalo. He received his Ph.D. (1991) and M.S. (1989) degrees in Mechanical Engineering from the University of Maryland at College Park, while he worked at the Institute for Systems Research and INRIA, France. His B.E. (1987) degree in Mechanical Engineering is from University of Roorkee, India. Dr. Nagi's major research thrust is in the area of production systems. His recent research interests are in Agile Enterprises, Information-Based Manufacturing and Logistics and Supply Chain Management. He was awarded the National Science Foundation's faculty early career award in 1996 and SME's Milton C. Shaw Outstanding Young Manufacturing Engineer Award of 1999. He is a member of IIE, SME and ASEE.

Mangesh P. Bhandarkar is a software engineer with Inso Corporation, PDM Division. He represents Inso corporation at the Object Management Group and is a co-author of the PDM-Enabler specification. He is currently a member of the finalization task force for the specification. He has an M.S. in Industrial Engineering from SUNY Buffalo (1997) and B.S. in Production Engineering from University of Bombay, India (1995).