ELSEVIER

Computer Integrated Manufacturing Systems Vol. 11, No. 1–2, p. 67–75, 1998

Printed in Great Britain
0951-5240/98 $ — see front matter

PII: S0951-5240(98)00010-X

A generic integrated system from CAD to CAPP: a neutral file-cum-GT approach

Henry Lau* and Bing Jiang

Advanced Technology Education Service, Regency Institute, Days Road, Regency Park, Adelaide, Australia

This paper describes a methodology for the direct information transfer from CAD to CAPP, linking dissimilar 'islands of automation'. In this methodology, a neutral file, which evolves from a CAD model, is used as the information source for generating the GT codes based on which detailed process plans are generated. The significance of this research can be shown when applied to the many designs which have been generated on dissimilar CAD systems in the past and now can be automatically coded for multiple manufacturing purposes. The creation of a generic integration system from CAD to CAPP can then be achieved. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords: computer aided process planning, group technology, neutral file, generic integration, form features

1. Introduction

Effective communication is a cornerstone of human progress. While technology strives to make our lives easier, it also has the reverse effect due to ineffective communication. Computer systems such as Computer Aided Design (CAD), Computer Aided Process Planning (CAPP) and Computer Aided Manufacture (CAM) are classified as 'islands of automation' and it is important that they can be linked together to achieve information exchange without taking into consideration the dissimilar formats being used by the various systems. In particular, CAPP is a key manufacturing technological development which can parallel design analysis and therefore it can provide insight into manufacturing alternatives prior to committing to final decisions about the design of components or products. The integration of CAPP with other interrelated systems, such as CAD and CAM, enhances design/production flexibility so that production rates and cost estimates can be used as evaluation criteria in selecting design and processing alternatives.

*Corresponding author: Tel: +618-356-1993 Fax: +618-348-4485; e-mail: henrylau@tafe.sa.edu.au

In the past years, there are successful examples of CAPP implementations employing Group Technology (GT) which can be defined as a technique for identifying and bringing together related or similar components in a production process in order to take advantage of their similarities[1]. In 1970, the Organization for Industrial Research Inc. (formerly INO) developed a CAPP system named MIPLAN[2] which used the MICLASS GT coding scheme. In the beginning of the 1980s, Lamp Equipment Operation (LEO) of General Electric Co. developed the MIPLAN CAPP system[3] for their light bulbs manufacturing machines based on the MIPLAN system. Some other typical CAPP systems developed in 1970s and the 1980s include GENPLAN by the Lockheed Company[4], and XPS-I which was CAM-I's experimental planning system[5]. GT coding is a very convenient design/manufacturing technique to summarize the important aspects of components. Researchers have found that components which have similar geometry also have similar process plans. With these characteristics, process plans for new parts can be obtained quickly by retrieving, and perhaps, modifying plans for similar parts via the GT categories.

One main problem with current CAPP systems is the lack of a complete component representation. Ideally the component description would be extracted from a CAD database and the output directed to the process planning system. This method can realize automated process planning. In practice, this is difficult, and is, in fact, a subject of the current research. Thus, it is difficult to identify the manufacturing significance from purely geometric representations. To overcome this difficulty, a flexible length coding scheme has been developed which has been specifically designed to represent the details of form features of components. In this paper, this new coding scheme is used in the development of a generic system which provides direct integration between CAD and CAPP.

Further literature study shows that Henderson and Musti[6] developed a system of software, named CODER, which can be used to analyze the solid model of a part and generate a GT part code based on the DCLASS coding system automatically. Kishinami *et al.*[7] introduced an approach for the integration of CAD/CAPP/CAM based on a geometric model named Cell-Constructed-geometric-Model (CCM) and a neutral meta-interface called Machining Kernel Software (MKS). Examples of these two proposed systems are described in great detail in the related articles. However, it remains a major concern regarding the extraction of data from CAD databases, due to the existence of dissimilar CAD systems and formats. With the lack of a generic integration methodology, the need for software rewrites to deal with individual situations seems unavoidable. This problem has been addressed in this paper with the introduction of a generic integration system which incorporates the employment of a neutral file and a new coding scheme as the means to achieve full integration from CAD to CAPP in a cost effective way. This proposed system is generic in the sense that effective data communication can be achieved regardless of the formats of the data being used.

2. Neutral file standards

A neutral file can be defined as the file with a format which is independent of any specific system standards, and which acts as an 'agent' to connect dissimilar computer systems that cannot normally communicate with each other due to format incompatibility. It should be noted that neutral format is the 'acceptable' and 'consistent' format within an organization or a group of organizations and the format may differ from other organizations which may set their own format standards.

A review shows that companies are using neutral format files to achieve integration of manufacturing systems[8–11]. More important of these standards, as summarized by Mult *et al.*[8] and Bohn[11], are:

GKS-3-D = Graphical Kernel System
CGI = Computer Graphics Interface
CGM = Computer Graphics Metafile
IGES = Initial Graphic Exchange Specification
SET = Standard d Exchange et de Transport
VDAFS = VDA-Flachenshnitt
PDES = Product Data Exchange Specification
STEP = Standard for Exchange of Product
CAD-NT = CAD-Normteile
CLDATA = Cutter Location Data
IRDATA = Industrial Robot Data
PHIGS = Programmers Hierachica Graphic System

As shown in the above table, different neutral format standards are being used for different application purposes. In manufacturing, STEP (ISO-10303) has become a widely used standard for product data representation and exchange[11]. STEP includes a series of International Standards with the ai of defining data across the full engineering and manufacturing life cycle and the ability to share data across various applications.

Within STEP, there are a number of Application Protocols (AP) designed for different application purposes. Recently, STEP AP203 interface, also called Configuration Controlled Design, has been widely supported in the manufacturing sector. AP203 is a STEP application protocol for the exchange of 3D product data along with its configuration information. It defines the exchange of product definitions with three-dimensional shape representations and the configuration of those product definitions. EDS Unigraphics started to develop the STEP AP203 as part of the AeroSTEP project in June, 1993[12]. This project was sponsored by PDES Inc. and involved Boeing and its three engine suppliers: GE Aircraft Engines, Pratt & Whitney and Rolls Royce. The goal of the project was to develop an interface to exchange digital pre-assembly data with the suppliers' CAD/CAM/CAPP vendor products including CATIA and ComputerVision's CADDS using STEP AP203 as the neutral format. The project proves to be successful due to the joint effort of vendors to work together to solve common challenges. It is widely believed that STEP will eventually replace IGES (Initial Graphics Exchange Specification) as the neutral file format for exchange data between dissimilar CAD/CAM/CAPP systems. While IGES only covers design, STEP stores complete product life cycle information — from concept through design, analysis, manufacturing, verification and support — as it matures. AP203 is supported by CAD packages such as Pro/Engineer and companies like SDRC[13] and EDS Unigraphics have built their STEP AP203 translators for different CAD packages such as CATIA, Pro/Engineer and CADDS. STEP Tools Inc. has also developed STEP converters which support a number of CAD data formats including AutoCAD SAT file and Virtual Reality Modeling Language (VRML) formats[14].

3. Principles of generic system

The approach of this research makes use of a STEP-compliant neutral file to connect dissimilar CAD packages to CAPP through a Flexible Digit Length (FDL) coding scheme which was designed to cover all the form features of a component. The methodology for the development of this generic integrated system is illustrated in a diagram as shown in *Figure 1* with descriptions covered in the following sections.

3.1. Data conversion

In order to ensure that the CAD data can be recognized by other computer systems with ease, it is essential that the data are in the form of a format which is supported by all the other systems, i.e. a neutral format. In this research, STEP AP203 is suggested to be the standard for the neutral file due to the fact that it is a universally accepted standard for conversion programs available for proprietary CAD packages including AutoCAD, CATIA, Pro/Engineer and CADDS. Furthermore, STEP AP203 is designed for mechanical assembly components and is appropriate to be used in mechanical design components for the manufacturing industry. Some packages, such as Pro/Engineer, have their own internal conversion program to convert the CAD data directly to STEP AP203 format and if an internal conversion program for certain CAD packages is not available, a third-party conversion program, such as ParaSTEP Import/Export System[15], can be used to perform the conversion task. This neutral file conversion is the first step of the whole integration process.

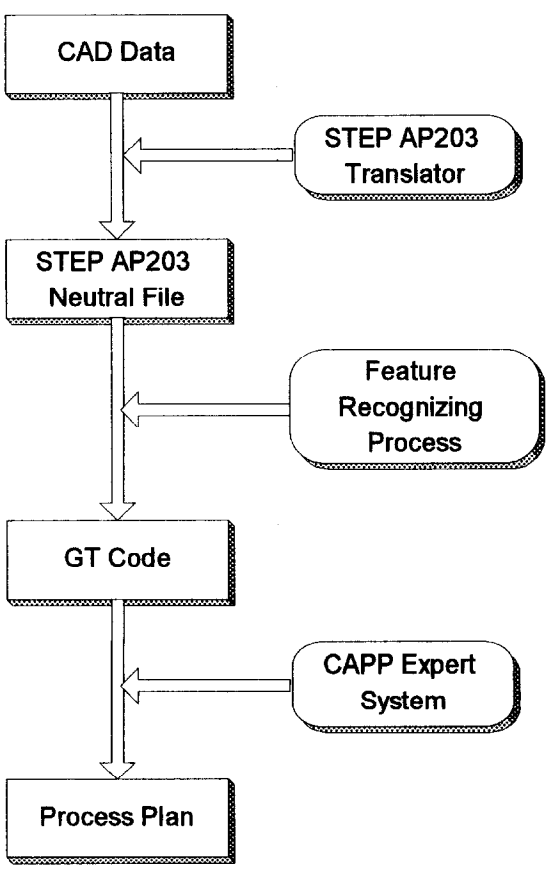

Figure 1 A generic integrated system from CAD to CAPP

3.2. Feature recognition

With the CAD data in a neutral format, the second step is to extract the form features of the design based on the data of this neutral file. In this research, form feature is defined as a geometric entity or set of entities, which either directly or indirectly determine a code-related digit. The features required by a part coding system are related to machining operations, e.g. a slot can be produced by milling operations. A form recognition program is required to scan through the data of the file and special techniques are employed to identify the component features included such as a slot, protrusion etc. Because these features are directly related to the production of GT codes of the involved component, this form recognition process needs to operate in connection with the GT coding process. In this feature recognition process, a program needs to be developed to check through the CAD file and extract the features which are used for GT coding. This program is generic since once it is written, it can be used for all other CAD files without any modification. This is possible because all the CAD data are in a neutral format which is supported by the program.

3.3. GT coding

A new coding scheme has been developed particularly for process planning purposes[16]. The main characteristic of this scheme is that unlike other coding schemes which have a rigid digit length structure, it has a flexible digit length capability and that makes it possible to include all the details of form features in the codes. The features recognition process works closely with this GT coding process. A specially written program is required to generate the component GT codes based on the information from the feature recognition process. It should be noted once this program is developed, it can be used for other components without modification.

3.4. Generation of process plans

The final step is the generation of process plans for manufacture of the involved components. This achieved through a program which is able to interpret the GT codes and generate an optimized process plan for the production of the component. An expert system may be advisable for this task due to the rule-based features of such a system.

A generic integrated system from CAD to CAPP named GISCAPP has been implemented based on the methodology as described above. There are four main units of GISCAPP including a Generic Data Converter (GDC), Generic Feature Recognizer (GFR), GT Coder (GTC) and Process Plan Gener-

ator (PPG). These four system units of GISCAPP are described in the following sections.

4. Generic data converter

The role of GDC is to convert the incoming CAD data into the STEP AP203 format. This unit is generic in the sense that it is able to perform format conversion regardless of the data format of the incoming CAD files. This is possible due to the fact that proprietary AP203 translators for various CAD systems including CATIA, CADDS, Pro/Engineer and AutoCAD are incorporated into this unit. Other CAD systems not included in this list are required to have their CAD data converted to any of the formats in compliance with the four major CAD systems. This can be achieved by converting the data to IGES or DXF formats which are considered to be the 'de facto' formats for CAD drawings and supported by all the four CAD packages. To run this unit, the user needs to specify the format of the incoming file and the appropriate translator is then selected to start the conversion process.

5. Generic feature recognizer

This unit is basically a specially written program for scanning the STEP AP203 file with the objective to identify the form features which are interpreted as STEP-compliant codes. Visual Basic is the main programming tool for developing this unit which includes a number of specially written *procedures* to perform the required tasks[17]. To understand the operations of this unit, the format structure of STEP AP203 needs to be described here. In short, STEP AP203 data file is a text (ASCII) file which contains geometric data of a component including boundary representation data (B-rep) data such as shells, faces, vertices; surface geometric data such as planes, cylinders, cones, torii; curve geometric data such as lines, circles, ellipses, b-spline curves. It is not the intention of this paper to provide a detailed description of the format structure of STEP AP203. However, it is necessary to have some basic ideas of the format features of this standard in order to understand the operations of GFR. The extract of a STEP AP203 file for a box with six plane surfaces (see *Figure 2*) is shown below.

```
.....
#21 = PLANE(',#20);
.....
#47 = FACE_BOUND(',#46,.T.);
#48 = ADVANCED_FACE(',(#47),#21,.F.);
                                Rem: The first face
.....
#53 = PLANE(',#52);
.....
#79 = FACE_BOUND(',#78,.T.);
#80 = ADVANCED_FACE(',(#79),#53,.T.);
                                Rem: The second face
```

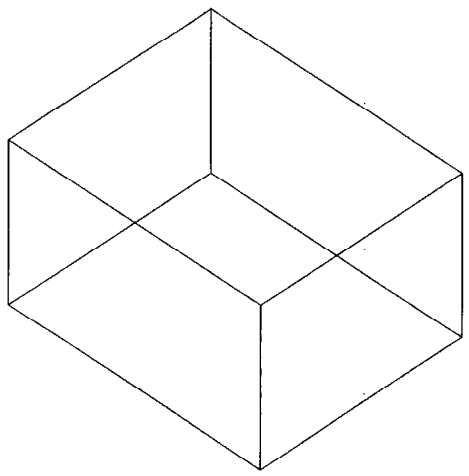

Figure 2 Block example

```
.....
#85 = PLANE(',#84);
.....
#101 = FACE_BOUND(',#100,.T.);
#102 = ADVANCED_FACE(',(#101),#85,.T.);
                                Rem: The third face
.....
#107 = PLANE(',#106);
.....
#118 = FACE_BOUND(',#117,.T.);
#119 = ADVANCED_FACE(',(#118),#107,.T.);
                                Rem: The fourth face
.....
#124 = PLANE(',#123);
.....
#135 = FACE_BOUND(',#134,.T.);
#136 = ADVANCED_FACE(',(#135),#124,.T.);
                                Rem: the fifth face
.....
#141 = PLANE(',#140);
.....
#147 = FACE_BOUND(',#146,.T.);
#148 = ADVANCED_FACE(',(#147),#141,.T.);
                                Rem: The sixth face
#149 = CLOSED_SHELL(',(#48,#80,#120,#119,
       #136,#148));
.....
```

In general, the main features of the STEP file are summarized below:

1. As shown in Line #149, CLOSED_SHELL specifies all the surfaces which form the component. In this example, the component is a box which has six flat surfaces which are referred to Line #48, #80, #102, #119, #136 and #148. The surface can be flat, curved. In this case, all are flat surfaces as indicated in lines #21, #53, #85, #107, #124, #141.
2. Every statement starts with a line number which is in ascending ordr with the increment of 1 for the following statement.

3. The geometric details of each face are specified first and then all these faces are joined together to form the final shape of the component.
4. The details of every face joined together to form the component can be traced including the edges, directions and lengths of the lines forming the face.

The type of manufacturing operations adopted is very much related to the features of the component and in this case it is obvious that milling operations are appropriate for producing this box with six flat surface.

5.1. Recognizing mid-level features

Henderson and Musti[16] define three types of form features including low-level features, mid-level features and high-level features. In short, low-level features are the primitives of a solid modeling system including face, line, point, edge and vertex; mid-level features are sets of low-level features such as 'depression' and 'protrusion' features; high-level features differ in definition from one coding scheme to another. For example, a 'hole' may be defined in classification scheme A as a rotationally symmetric depression and in scheme B as merely a depression requiring material removal. For the purpose of defining process plans, it is important to identify the mid-level feature such a slot on the surface of a block as shown in *Figure 3*.

Take the example as shown in *Figure 3*, the extract of the STEP AP203 file is as below:

```
.....
#4 = LINE(',#3,#2);
.....
#97 = CARTESIAN_POINT(',(0.E0,0.E0,0.E0));
.....
#99 = VERTEX_POINT(',#97)
.....
#130 = DIRECTION(',(0.E0,0.E0,1.E0));
#131 = DIRECTION(',(1.E0,0.E0,1.E0));
#132 = AXIS2_PLACEMENT_3D(',#129,
       #130,#131);
```

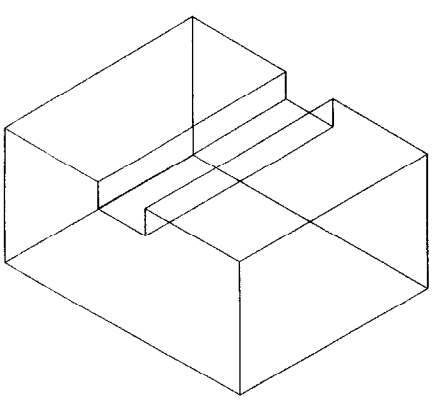

Figure 3 Block with a rectangular slot

```
#133 = PLANE(',#132);
#134 = EDGE_CURVE(',#99,#100,#4,.T.);
#135 = ORIENTED_EDGE(',*,*,#134,.T.);
.....
#150 = EDGE_LOOP(',(#135,#137,#139,#141,
       #143,#145,#147,#149));
#151 = FACE_OUTER_BOUND(',#150,.F.);
#152 = ADVANCED_FACE(',(#151),#133,.F.);
.....
#281 = CLOSED_SHELL(',(#152,#176,#190,#203,
       #216,#229,#242,#255,#268,#280));
.....
```

The technique to detect the presence of a slot is based on the 'vector directions' and lengths of the edges forming the surface. Referring to the box with a slot as shown in *Figure 3*, the slot can be detected if the directions of the edges (lines) and the lengths of the edges are known. In summary, the method to detect a slot is shown below:

- Locate the CLOSED_SHELL statement from the STEP file and retrieve the information regarding individual surfaces included. In this example, Line #281 contains the CLOSED_SHELL statement with the Line Numbers for the ten included faces, i.e. #152, #175...
- Check the vector directions and lengths of the edges which form the surfaces. For example, the edge (#134) contains the cartesian points (#97,#99) of the two ends of the edge. With this information, the vector direction of the line can be determined. Referring to *Figure 4*, the vector directions of the edges with the data horizontal right (L), vertical down (M), horizontal right (N), vertical up (O) and horizontal right (P) indicate that a slot may exist. The data about the lengths of the edges involved indicate that it is a normal rectangular slot (lengths of vertical edges are the same).
- To confirm the presence of a slot, there must be another surface which also has the edges indicating the existence of a slot. This surface must be parallel to the one detected before. In STEP AP203, it is possible to find out whether one surface is parallel to the other by checking the axis (AXIS2_PLACEMENT_3D in #132) of the two surfaces (STEP AP203 shows the axis for every surface). They must have the same direction (#130 and #131 indicate the direction for one surface).

While this technique is mainly used for detecting the presence of a slot, a similar technique can be

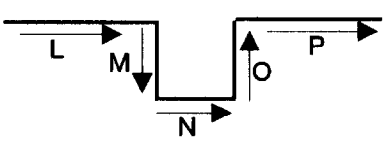

Figure 4 Detecting the presence of a rectangular slot

employed to determine the presence of other mid-level features such as protrusions, slots etc.

In this research, a program written in Visual Basic (VB) is employed to scan the file and retrieve the relevant data. A number of VB *Functions* and *Procedures* are written to perform the task. It is not the purpose of this paper to provide details about this program. However, the methodology for producing such program is described in the following context.

A number of built-in commands of VB can be used to create the necessary *Functions*. The command *instr( )* is one of them and can be used to locate a specified text string e.g. CLOSED_SHELL of a file. To illustate its use, an example is shown as below:

```
Dim filestring As String                          *Line 1
Dim start_search As Long                          *Line 2
Dim curindex As Long                              *Line 3
Dim first_surface_start As Long                   *Line 4
Dim first_surface_end As Long                     *Line 5
start_search = 0                                  *Line 6
curindex = instr(start_search+1,filestring,
  'CLOSED_SHELL'                                  *Line 7
first_surface_start = instr(curindex+1,filestring, '#')
                                                  *Line 8
first_surface_end = instr(curindex+1,filestring, ',')
                                                  *Line 9
```

In this example, *filestring* contains the text of the STEP-compliant file. Line 7 invokes the *instr( )* command which searches for the string 'CLOSED_SHELL' from the text data contained in *filestring* starting from the first character (*start_search*+1) of the text. The Function *instr( )* will return the value which contains the location of 'CLOSED_SHELL' e.g. 1234 (the 1234th character of *filestring*) and stores in the variable *curindex*. Line 8 searches for the '#' immediately after the string 'CLOSED_SHELL'. As the search starts at location *curindex+1*, this can guarantee that the location of the FIRST '#' is obtained. Line 9 locates the first ',', after 'CLOSED_SHELL'. The line number of the first surface is in between '#' and ',' of which locations are known. The line number can then be retrieved easily using a string manipulation command such as *mid$( )*. The line numbers of other surfaces and then the vector directions, lengths of related edges can also be obtained using similar programming techniques.

In this research, a number of subroutines have been developed for detecting various mid-level features including steps, rectangular and dove-tail slots, pockets and rectangular protrusions.

6. GT Coder

The next unit is the GT coder which receives the input from the GFR and output the GT codes based on a new coding scheme. The details of the new coding scheme has been presented in a previous paper[16] and is not to be covered here. However, a brief description of the scheme is necessary and is covered as below:

- The structure of the new coding scheme, which consists of three levels, is a hybrid of the monocode and the polycode structure. The digit of level 1 (overall description) is the monocode type structure in which each digit is completely independent of all other digits. The level 2 (the machined shapes) and level 3 (the machined shape's attributes) have the polycode structure with digits in the level 3 decided by the digits in level 2, i.e. how many attributes and what sort of attributes of machined shape are decided by the machined shape digits.
- The new GT coding scheme has a flexible digit length which is different from most existing coding schemes which are of the 'rigid digit length' nature. Each machined shape of the component is described in the proposed coding scheme without exception in order to ensure that a complete picture of the overall shape of the component is included. Therefore, the more machined shapes a component has, the more digits the new code will be. This flexible digit length structure ensures that the new code carries enough information for process planning.
- Level 1 of the proposed coding scheme belongs to monocode structure. The code value 1.1 (overall shape) is digit one's value. The code value 1.2 (maximum length) and the code value 1.3 (component material) are digit two's and digit three's value, respectively.
- There are three code values in level 2 for the external machined shape, the holes and the internal machined shape (other than the hole). These values represent the individual machined shapes of the component. In the new coding scheme, each machined shape's attributes are set up in level 3. Machined shape's attribute values are chosen from the level 3. Thus, level 2 and level 3 are of the polycode structures. The codes in level 3 (machined shape's attribute) are decided by the codes in level 2 (machined shapes).
- Level 3 is concerned with the machined shape's attribute code values, where the machined shape's dimensions, tolerances, surface finish and orientation are represented. The dimension attribute values are chosen from value 3.1. The value 3.1 has the code from 0 to N (i.e. *any* integer). If the dimension size is not an integer, the code number is corrected to the nearest integer.

In order to produce the correct GT codes for a component, the coder needs to work with GFR seamlessly because the codes are very much related to the various levels of form features of the component. Take the example of coding for level 1 which is concerned with overall description of the component. *Table 1* describes the code values for level 1 based on the new coding scheme. There are three digits of this level with the first digit about overall shape, second digit about maximum length and the third digit about component material. For the first digit, GFR

Table 1 Overall description of level 1

Code value 1.1 Overall shape (A = length, B = width, C = height)		Code value 1.2 maximum length (mm)		Code value 1.3 component material	
0	Do not know	0	Do not know	0	Do not know
1	Cube components, A/B < = 3, A/C < = 4	1	A < = 16	1	Mild steel
2	Flat components, A/B < = 3, A/C > 4	2	16 < A < = 50	2	Hardened steel
3	Long components, A/B > 3	3	50 < A < = 100	3	Stainless steel
		4	100 < A < = 160	4	Tool steel
		5	160 < A < = 240	5	Cast iron
		6	240 < A < = 360	6	Heat resistant alloys
		7	360 < A < = 600	7	Titanium
		8	600 < A < = 1000	8	Aluminium
		9	1000 < A < = 2000	9	Brass
		10	2000 < A		

determines whether it is a cube component, flat component or long component based on the data content of the STEP file. By checking the details of surfaces of the component, the overall shape can be determined. For example, if it is a cube component, the feature can be found from the vector directions and the lengths of the edges joining the surfaces. The maximum length of the component can be found easily from the STEP data file. The component material is included in the STEP file provided that it is mentioned in the CAD drawing. It can be seen that the three digits of level 1 for the new GT schemes can be determined with the information from GFR which is able to scan and detect any detailed geometric features of the component. Level 2 and level 3 are more related to the form features of the component such as slots, pockets, steps, etc. As illustrated in the previous sections, most of the form features can be detected by GFR using special written subroutines. Thus, the GT code of the component based on the new coding scheme can be derived.

7. Process plan generator

This is the last, but not the least, system unit of GISCAPP. It consists of six modules which include blank selection, process selection, machine selection, cutting tool selection, cutting condition (cutting speed, feed rate and cutting depth) selection, processes sequencing. A module is a part of the system with a well defined function and well defined interfaces to both the other modules of the system and users. Modules are based on the knowledge of basic machining processes, cutting tools, Numerical Control (NC) milling machines and cutting conditions such as cutting speed, feed rate and cutting depth to generate new plans. The knowledge comes from workshop experience and reference materials. The optimisation of the process sequence is initially set to minimise the number of workpiece setups and the required number of cutting tool changes.

The goal of this unit is to assist the machinist in determining an optimal process plan for the manufacturing of components. It is desirable to automate as much of the planning decision-making as possible. To meet the requirement for developing this unit, an expert system shell, ART-IM[18], has been selected as the tool for the task. In this unit, the tooling knowledge including tools and machines knowledge are encoded as *facts* and *schema*. A *fact* is a fundamental piece of knowledge sitting in the knowledge base. For example,

f-2 (cutting-tool T-MAX145 face-milling-cutter)

This is a fact number 2, which says that the cutting tool T-MAX145 (T-MAX145 is a cutting tool code used by Sandvic Company) is a face-milling cutter. Thus, a fact is just a simple statement linking various pieces of information in some significant way. A *schema* is a collection of *facts* that represents a single object (or class of objects). A *schema* is similar to a frame representation. The *facts* in a *schema* like slots in a frame. For instance, a group of face-miling cutters can be represented easily as:

```
(defschema face-milling-cutter
  (process-capability plane-surface)
  (minimum-diameter 40))
```

In this example, the *schema* face-milling-cutter includes the information that all face-milling-cutters can produce the plane surface and minimum diameter of face-milling-cutters is 40 mm.

In this unit, operational knowledge is expressed by *rules*. Suppose we have a rule tht selects a face-milling-cutter to produce a plane surface. The Process Plan Generator uses the following rule:

```
(defrule plane-surface-cutting-tool-selection
  (geometrical-shape plane-surface)
  (material hardened-steel)
  =>
  (assert(harden-steel plane-surface face-milling-
  cutter U-MAX))
  (printout t 'The cutting tool for hardened steel
  plane surface is face-milling cutter U-MAX't))
```

The symbol (=>) is used in ART-IM to separate the conditions pattern and action portions of a rule. The actions in this rule are to add new *fact*

(harden-steel plane-surface face-miling-cutter U-MAX) to knowledge and screen print the sentence 'The cutting tool for hardened steel plane surface is face-milling cutter U-MAX'.*7.1. Blank selection module*

Blank is the raw stock material which is cut by cutting tools to produce a component. Blank selection depends on the component's geometry. The blocks data are stored in knowledge base of GISCAPP as *facts* and *schemas*. For prismatic components, the blocks data are set to two classes of objects: Prismatic-Blank and its child class, Cubic-Prismatic-Blank, Flat-Prismatic-Blank and Long-Prismatic-Blank. Some of the *schemas* representing blanks data are:

```
defschema Prismatic-Blank
  (Length $?)
  (Width $?)
  (Height $?)
  (Materials $?))
(defschema Cubic-Prismatic-Blank-1
  (Length 16)
  (is-a Prismatic-Blank))
(defschema Flat-Prismatic-Blank-3
  (Length 100)
  (is-a Prismatic-Blank))
```

The '$?' stands in place of several elements of a fact by a segment operator '$' with a wildcard '?'. The 'is-a' symbol is one of system-defined inheritance relations in ART-IM. It allows child *schemas* inherit *facts* from parent *schemas*.

The selection of blanks depends on the level one digit values of new code, i.e. the overall shape, the maximum length and material. Several rules are written to choose the blank from the knowledge base. One of the rules is:

```
(defrule Component-Blank-Selection
  (digit 1.1 1)
  (digit1.2 1)
  (digit1.3 1)
  (schema Cubic-Prismatic-Blank-1
    (Length 16)
    (is-a Prismatic-Blank))
⇒
  (assert(schema component-1-blank
    (is-a Cubic-Prismatic-Blank-1)))
```

The term *assert* is a system-defined function which adds new *facts* or *schemas* to knowledge base. Thus, the blank data of the new component has been stored in the knowledge base and will be used by the rules in other modules, such as process selection module. This alteration of knowledge base also provides communication among modules.

7.2. The process selection module

The function of the process selection module is that it selects operations and cutting path(es) for each machined shape (i.e. feature). For example, a plane surface may need two operations, rough cut and finish cut. Another example is a big hole needs one drilling and one boring operation. Cutting path is also important for each operation. Thus, the process selection module of the GISCAPP decides the operations needed for the component manufacturing.

The process selection module of the GISCAPP contains the machined shape production capability knowledge from the workshop and the component geometry knowledge which comes from the component code value. The shape producing capabilities of a process is the geometrical shape a specific machining operation can produce. Some selected machined shapes are considered in the prototype GISCAPP, such as plane surface, slot, pocket, contour and straight hole.

In the process selection module, the level two digit values and the surface finish digit values of level three in the new code are used to reason the most appropriate processes and operations. The processes decision depends on the machined shape and the rough and finish operations which also rely on the surface finish of each machined shape.

7.3. The tool selection module

The process selection module selects a set of feasible processes and operations for each feature (machined shape). The most appropriate cutting tools are then selected. Cutting tools have the capabilities to produce features (geometrical shapes). They can affect the dimension, the tolerance and surface finish of features.

In the tool selection module, tool selection rules return a set of feasible tools depending on the digit values in level 2 and level 3. The tool knowledge base uses *facts* and relational *schemas*. In each knowledge record, information on tool number, tool material, tool geometry, dimension, tool life left, and operation types are stored.

The GISCAPP tries to use the same tool for several operations or machined shapes. Although one tool may not be the best for all of the operations or machined shapes, reducing the number of tools for a job can reduce the number of tool changes.

7.4. Machine selection module

In the machine selection module, the machine selected depends on the digit values of the overall shape and maximum length of the component. The selection of machines also considers the cutting tool selected and only the machines with tools selection and set up capabilities are chosen.

7.5. Cutting conditions selection module

Cutting conditions include cutting speed, feed rate and depth of cut. The knowledge of cutting conditions selection comes from workshop experience.

Depth of cut for drills, end mills should not exceed half the diameter of the cutter. For finish cuts, it is necessary to increase cutting speeds and reduce feedrates by approximately 50% and then use climb milling. Thus, the cutting conditions selection *rules* are written based on the component material digit values and the tools diameter which are stored in the knowledge base by tools selection module.

7.6. Process sequencing module

The processes sequencing module chooses the suitable processes and operations sequence for the final plan. A process is defined as the ability to produce a particular component feature or to meet a particular component requirement. Generally, there is always more than one process sequence which is feasible for producing a particular characteristic of a component.

In GISCAPP, two factors are taken into consideration. The first factor is the general knowledge of process sequencing to produce certain geometric component features and to meet certain component tolerance and/or surface finish requirements. Another factor is the availability of a cutting tool to perform the required process and minimize the number of tool changes.

The general knowledge used to determine the processes sequencing is based on the information obtained from published material and practical sources. For example, one cannot bore nor ream a hole before it has been drilled.

8. Testing of GISCAPP

GISCAPP has been tested for a limited number of components with complex features encompassing protrusions, depressions and slots. Test results indicate that the subroutines written in VB based on the technique described in the earlier section are useful to detect the existence of various mid-level features. The only concern is that because GFR requires to scan through the whole file one time in order to detect one specified feature, it becomes fairly time-consuming when a long STEP file with a significant number of features, say, 20, are to be detected. In this case, the file has to be scanned through 20 times in order to ensure that any of these 20 features will not be missed. Therefore, it is recommended that a fast computer, preferably a Pentium PC with 200 MHz is used to deal with some complex components. Further study is in progress to improve the scanning technique so that various features can be detected by scanning through the file with just one go. In general, test results are promising and that provides proof that GISCAPP can be used in an industrial environment where complex components exist in daily production.

9. Conclusion

GISCAPP is a system which links CAD to CAPP for direct information transfer in a generic way. It is generic in the sense that dissimilar CAD models can be used to generate process plans without any additional software rewrites. This can be achieved by adopting the STEP AP203 as the neutral format, which is a universally accepted format and supported by major CAD software developers. CAD models from Pro/Engineer, AutoCAD, CADDS and others are able to be translated to STEP AP203 format by using built-in facility or third-party translators. GISCAPP contains a generic converter, a generic feature recognizer and a process planner, all of which work together seamlessly. Generic methodology for the creation of these system units have been discussed in this paper. The contribution of GISCAPP is its important role in the generic integration from CAD to CAPP systems, which is considered as a 'bottleneck' in the implementation of a Computer Integration Manufacturing (CIM) system in a company.

References

1 **Gallagher, C and Knight, W** *Group technology*. London, Butterworth: 1973
2 **OIR** *Introduction to MIPLAN*. Organization for Industrial Research, Inc., 1981
3 **Schaffer, G** GT via automated process planning. *American Machinist*, (1980) May, pp 119–122
4 **Tulkoff, J** Lookheed's GENPLAN. *Proceedings of 18th Numerical Control Society Annual Meeting and Technical Conference*, Dallas, Texas, 1981, pp. 417–421
5 **Sack, C F** CAM-I's Experimental Planning System, XPS-I. *Proceedings of AUTOFACT'5*, Detroit, Michigan, Nov, 1983
6 **Henderson, M R and Musti, S** 'Automated group technology part coding from a three-dimensional CAD database', *Transaction of the ASME* Vol 110 (1988) 278–286
7 **Kishinami, T, Kanai, S and Saito, K** 'An integrated approach to CAD/CAPP/CAM based on cell-constructed-geometric-model (CCM)', *Robotics & Computer-Integrated Manufacturing* Vol 3 No 2 (1987) 215–220
8 **Millett, B C** 'Neutral manufacturing data: the gateway to CIM', *1990 Pacific Conference on Manufacturing Proceeding* Vol 1 (1990) 534–541
9 **Mult, H C and Hans, I** Advanced manufacturing to achieve world competitiveness. *Paper for Workshops Manufacturing Australia*, July, 1991
10 **Luca, A D** 'A step-by-step effective approach to computer integrated enterprise', *Australian Conference on Manufacturing Engineering* (1993) 109–113
11 **Bohn, J H** *Introduction to rapid prototyping (ISO-10303 STEP)*. New York: Randolph Hall, 1996
12 **Lietz, C** *EDS Unigraphics* URL: www.edsug.com/new/step.html, 1996
13 **SDRC Working Ideas**. *Data Translators* URL: www.sdrc.com/pub/working-ideas, 1997
14 **Step Translation Service**. URL: www.steptools.com/translate, 1997
15 **ParaSTEP**. *Grumman data systems* URL: axon.scra.org/products_and_services/products, 1997
16 **Jiang, B, Baines, K and Zockel, M** 'A new coding scheme for the optimisation of milling operations for utilisation by a generative expert CAPP system', *Journal of Materials processing Technology* Vol 63 (1997) 163–168
17 **Microsoft Corporation** *Programmer's guide*. Microsoft Visual Basic, 1995
18 **Clayton, D** *ART-IM programming tutorial*. CA, USA, Interence Corporation, 1990