PROCESS PLANNING FOR A MILLING MACHINE FROM A FEATURE-BASED DESIGN

Dr. Thomas R. Kramer

Guest Worker, National Bureau of Standards, &
Research Associate, Catholic University

ABSTRACT

In the Vertical Workstation (VWS) of the NBS Automated Manufacturing Research Facility, metal parts are machined automatically from a feature-based design. A simple two-and-a-half dimensional part may be designed and machined within an hour, allowing half the time for design input. With a design already in hand, the VWS software (which is written in LISP and runs on a Sun computer) will automatically prepare a process plan for a milling machine for making a part of the given design. The heart of the process plan is a list of machining operations to be carried out. The operations are selected by the system from among its repertoire of 21 possible operations. The process plan also includes a header and a list of tool requirements. The process plans produced by the system are later used as input to an automatic NC-coding system which writes code for the milling machine's controller.

INTRODUCTION

The Automated Manufacturing Research Facility (AMRF) at the National Bureau of Standards (NBS) is an experimental facility for developing techniques and standards for automated manufacturing. The Navy is a cosponsor of the AMRF and provides a large share of the funding. The AMRF has been described in several papers [1, 2, 3]. The AMRF includes three machining workstations, a cleaning and deburring workstation, an inspection workstation, and a material handling system. One of the machining workstations is the Vertical Workstation (VWS).

The VWS may be operated in stand-alone mode or as a subordinate element in the AMRF control hierarchy. Running in stand-alone mode, the VWS is a computer-integrated automated machining workstation. It includes a control system, an interactive system for creating and editing designs, an automatic process planning system, and an automatic NC-code generator.

The system is controlled from a Sun microcomputer. Other machinery and computers of the VWS include a Monarch VMC-75 milling center with a GE2000 controller, a Unimate robot with its Unimate controller to tend the milling center, two roller tables which serve as the interface between the VWS and the materials handling system, a powerful vacuum cleaner for chip removal, a pneumatic vise to hold parts being machined, pneumatic NBS-made grippers for the robot, a hydraulic pallet clamping system, signal processing electronics, and an HP 9000 computer which stands between the Sun and all the other equipment as an overseer.

The VWS produces machined metal parts. It is possible to design and machine a simple part within an hour in the VWS. To accomplish this, it is necessary to have a design protocol which lends itself to automatic machining. A user-friendly Part Design Editor has been developed which prepares design documents in the necessary format. The user sits at the Sun workstation and creates a design using the editor. The Process Planner is then called to write a plan for how to machine a part of that design. Next NC-code is generated automatically from the design and the plan. Finally the user tells the control system to make the part. The control system coordinates the activities of the workstation equipment so that the part blank is loaded onto the milling machine, the NC-code is sent to the milling machine and executed (making the part), and the finished part is unloaded.

The family of parts of interest in the VWS is those that are milled, rather than parts that are turned, stamped, or made in some other forming operation. Emphasis is placed on machining in small lots, roughly one to fifty parts of a single design. Materials of interest are those which can be machined, mainly metals and some plastics. Parts have been made from aluminum, steel, and brass during test runs.

The software for the VWS has been under development for three years by the author, Mr. Charles McLean, Dr. Jau-Shi Jun of NBS, Mr. Kie Nakpalohpo and others. In this paper this software is called the VWS2 system. It is written in the computer language Franz LISP and runs on a Sun microcomputer.

Six principal modules comprise the VWS2 system: the Production Management Operating System (the control system), the State Table Editor, the Equipment Program Generator, the Part Design Editor, the Process Planner, and the Data Execution module. Two of the VWS2 modules (the Process Planning module and the Data Execution module) deal with process plans. Process plans are the output of the Process Planning module and are part of the input to the Data Execution module. In addition to the principal modules, there are several distinct subsystems (such as drawing, which works for two of the modules). This paper deals in detail with only the Process Planning module. The workstation is described in more detail

in [4].

The Part Design Editor, Process Planning and the Data Execution modules may be accessed by the user through a small friendly front-end called vws_cadm. Vws_cadm asks the user a number of questions about what the user wants to do and then activates the appropriate module according to the user's desires. Vws_cadm has several other uses.

The overall architecture of the VWS2 system is largely due to Mr. Charles McLean, Group Leader of the NBS Production Management Systems Group. Process planning concepts and issues at the AMRF have been described in "Interactive Process Planning in the AMRF" by Mr. Peter Brown and Mr. McLean [5], and in "Research Issues in Process Planning at the National Bureau of Standards" [6] by Mr. Brown and Dr. Steven Ray.

This paper discusses process plans only as they relate to the VWS milling machine. In the remainder of the paper we will describe (1) the design protocol in the VWS2 system, (2) the process plan protocol for the VWS milling machine, (3) process plan enhancement, (4) process planning generation for the VWS milling machine, and (5) limitations of the system.

DESIGN PROTOCOL IN THE VWS2 SYSTEM

The VWS2 system uses a feature-based design protocol [7]. The design of a part is expressed as a list of features on a piece of stock. The piece of stock is always a rectangular block. The stock is fixed with respect to a three-dimensional cartesian coordinate system at the bottom, front, left corner of the block. The design system currently assumes that all features are being made from one side of the block.

Although all the features and subfeatures are purely geometric, they were selected to be included in the system on the basis of being features commonly found on machined parts that could be produced in one, or at most a very few, machining operations. Each feature and subfeature is a removed volume.

The primary features in the system in September, 1987 are: chamfer_out, groove, hole, pocket, straight_groove, text, contour_groove, contour_pocket, and side_contour. There are also subfeatures which may be made on the primary features: chamfer_out, chamfer_in, countersink, and thread. A feature is specified in the system by giving its name and the values of several parameters which specify its location, shape, and size. Parameter information is kept by storing the name of each parameter followed by its value. A summary of information about the features is

given in Table 1.

TABLE 1 FEATURE PARAMETERS AND SUBFEATURES

Feature Name	Feature Appearance	Required Parameters	Optional Subfeatures	Optional Parameters
chamfer_out	45 degree bevel on edge of block	chamfer_out_depth	none	none
contour_groove	Groove using a contour outline as tool path or edge. Bottom is round or flat.	corners depth width bottom_type	none	offset reference_feature
contour_pocket	Material inside a contour outline removed to constant depth	corners depth	none	reference_feature
groove	Groove using a rectangle with rounded corners as tool path. Bottom is round or flat	upper_l_x upper_l_y lower_r_x lower_r_y depth width bottom_type corner_radius	chamfer_in chamfer_out	chamfer_in_depth chamfer_out_depth reference_feature
hole	Circular hole with flat or conical bottom. May go through part	center_x center_y diameter depth bottom_type	chamfer_in countersink thread	chamfer_in_depth countersink_depth thread_diameter thread_depth threads_per_inch reference_feature
pocket *(2 types*) pocket_corners	Material inside a rectangle with rounded corners removed to constant depth	upper_l_x upper_l_y lower_r_x lower_r_y corner_radius depth	chamfer_in	chamfer_in_depth reference_feature
pocket_center	Same as pocket_corners	center_x center_y length width corner_radius depth	chamfer_in	chamfer_in_depth reference_feature
side_contour	Material outside a contour outline removed to constant depth	corners depth	none	reference_feature
straight_groove	Groove using a straight line as tool path. Bottom is round or flat	x1 y1 x2 y2 depth width bottom_type	chamfer_in	chamfer_in_depth reference_feature

TABLE 1 FEATURE PARAMETERS AND SUBFEATURES

Feature Name	Feature Appearance	Required Parameters	Optional Subfeatures	Optional Parameters
text	Letters and digits in five fonts. Upper case only. Round bottom only	lower_l_x lower_l_y text height depth line_width	none	font reference_feature

The “text” feature type requires that there be a template for each letter of each font stored in the database. A default font called “plain” was created for the system by Alton Quist. The VWS2 system has a font_making facility which creates new fonts by transformations of the plain font. We have been using four additional fonts called “broad”, “round”, “italic”, and “angular”, which were made by this facility. A font called “cilati” was created in less than a minute using this facility for the example part shown later in this paper.

The three contour features share the idea of a contour outline. A contour outline is a continuous string of straight line segments and arcs of circles. The Part Design Editor provides an easy way of making contour outlines. A contour_groove is made when a tool follows a contour outline. A contour_pocket is a flat-bottomed feature made by removing all the material inside a contour outline to a fixed depth. A side_contour is a mesa-like feature created by removing all the material outside a contour outline to a fixed depth. A contour outline for a contour_groove or a side_contour must be closed and must not intersect itself.

The design protocol includes the use of “reference features”. If feature A is to be made at the bottom of feature B, then one of the parameters of feature A is “reference_feature”, and the value of that parameter is the feature number of feature B. Whenever B is the reference feature for A, the outline of feature A must fit within the outline of feature B, and the bottom of feature B must be flat (except in the case of concentric drill holes). The idea of a reference feature is illustrated in Figure 1, which shows a hole having a pocket as its reference feature.

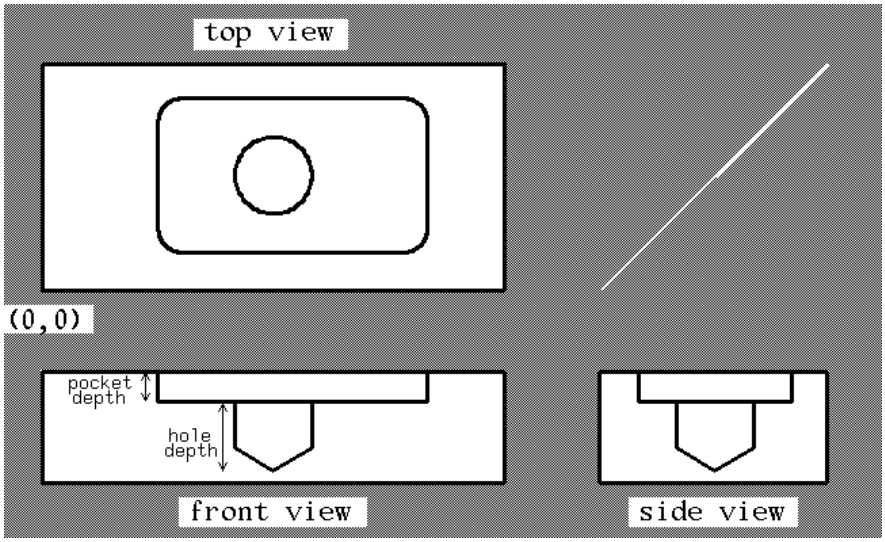

FIGURE 1 - REFERENCE FEATURE

The design data structure is a tree of attribute-value pairs. The value of some attributes is a subtree. For example, the design itself consists of two pairs: the attribute “header” followed by the header subtree (which includes the id number of the design, a very brief description, and the dimensions of the block) and the attribute “features” followed by the features subtree. The features subtree is composed of pairs in which the attributes are integers and the values are feature descriptions. This pairwise structure goes down to the lowest level of the design. The features are numbered sequentially starting with 1. The feature number is important as an identifier of the feature, but it would make no difference to the system if the features were scrambled and renumbered.

Table 2 shows an example of the design of a part. We will call it the ASME part for future reference. A picture of the ASME part prepared by the VWS Part Design Editor is shown in Figure 2. The ASME part is a demonstration part containing five feature types and three subfeature types. The design document of Table 2 and the picture of Figure 2 were prepared in 15 minutes using the interactive Design Editor. The ASME part has no mechanical function.

TABLE 2 ASME PART DESIGN

```
(setplist 'asme '
( header (header
          design_id  asme
          material  aluminum
          block_size  (block_size
            length 6  width 2.95  height 1.95)
          description "asme demo part")
 features   (features
  1 (1  feature_type         side_contour
          corners             (corners
          1 (1 x 0    y 2.5 radius 0.4)
             2 (2 x 5    y 2.5 radius 1)
             3 (3 x 4.5 y 0    radius join_back))
          depth              0.6)
  2 (2 feature_type         hole
          center_x            0.75
          center_y            0.6
          diameter            0.316
          depth              0.8
          bottom_type          conical
          thread_diameter      0.375
          threads_per_inch     16
          thread_depth         0.5
          countersink_diameter  0.45
          reference_feature     1)
  3 (3 feature_type         hole
          center_x            0.75
          center_y            0.6
          diameter            0.1719
          depth              0.3
          bottom_type          conical
          reference_feature     2)
  4 (4 feature_type         text
          text               "asme"
          font               cilati
          lower_l_x           2
          lower_l_y           1.8
          height              0.5
          depth              0.015
          line_width          0.1187434)
  5 (5 feature_type         pocket_corners
          upper_l_x           3.2
          upper_l_y           1.5
          lower_r_x           4.5
          lower_r_y           0.9
          depth              0.3
          corner_radius        0.3
          chamfer_in_depth     0.05)
  6 (6 feature_type         straight_groove
          x1                 4.25
          y1                 1.25
          x2                 3.5
          y2                 1.1
          depth               0.2
          width               0.25
          bottom_type          round
          reference_feature     5))))
```

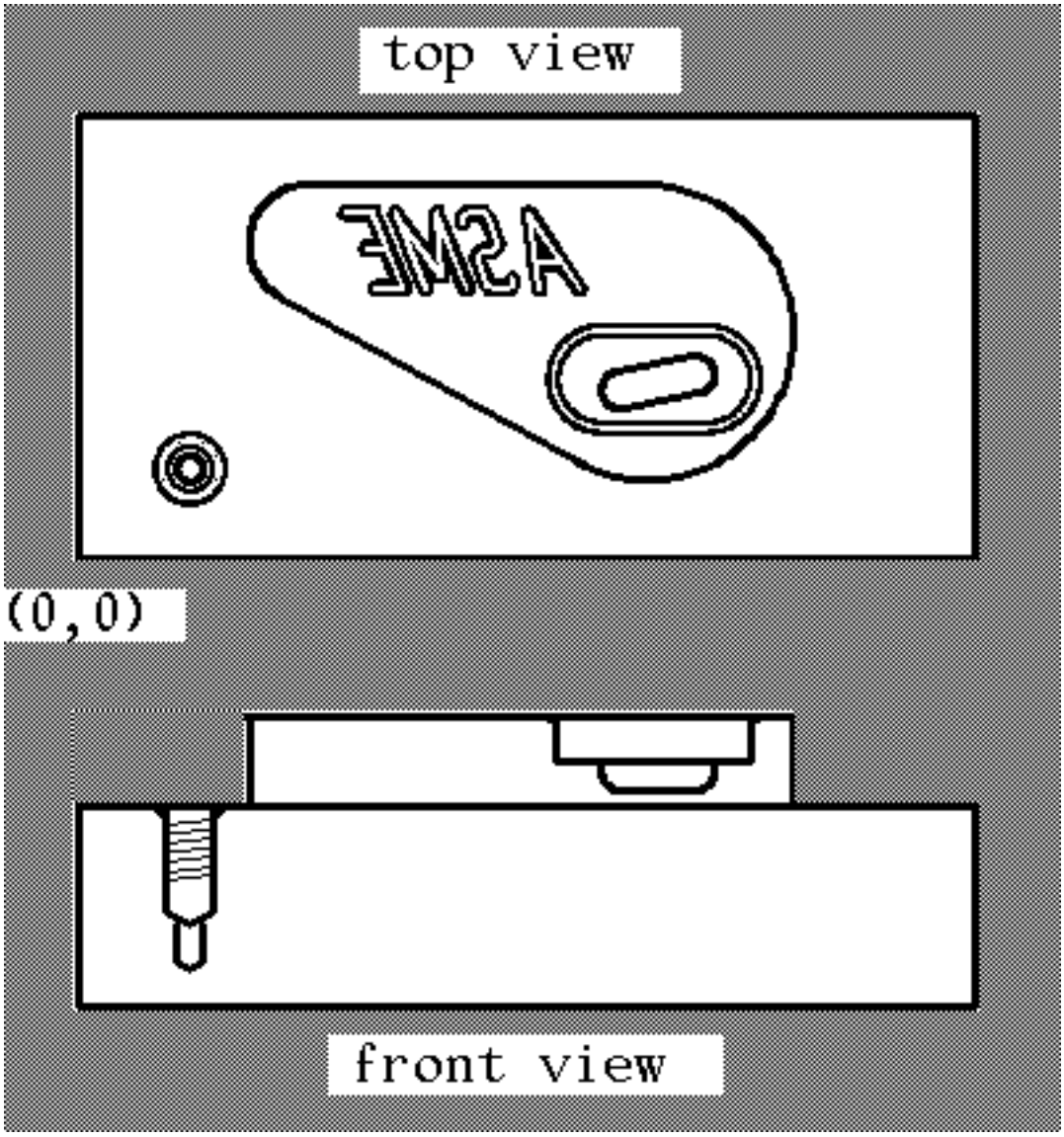

FIGURE 2 - DRAWING OF ASME PART

PROCESS PLAN PROTOCOL FOR THE VWS MILLING MACHINE

In ANSI standard Z94.10 - 1972, published by ASME, the term “process planning” is defined as “a procedure for determining the operations or actions necessary to transform material from one state to another” [8]. Chang and Wysk say, “process planning could be defined as the act of preparing detailed instructions to produce a part” [9]. In this paper we use “process plan” to mean the detailed instructions mentioned by Chang and Wysk. A method of describing process plans is called a process plan protocol.

To be usable in the VWS2 system, the format of a milling machine process plan in the LISP environment must be a LISP property list. Outside of the LISP environment there are two formats for process plans which are usable by the VWS. The first is an implementation for the VWS of the AMRF standard [10]. The second is a LISP-readable format. This paper describes only the LISP-readable format. A full description of both is given in [11].

Although the process plan protocol described here was created for the VWS milling machine, plans prepared according to the protocol could be used by any milling machine with equivalent capabilities. With the possible exception of probing, most modern NC-controlled milling machines have the capabilities needed to carry out the process plans generated in the VWS. Thus the plans themselves would not need modification (although the NC-coding system would). In this sense the VWS process plan protocol is machine independent.

A process plan for making the ASME part is shown in table 3. It was prepared by the VWS2 Process Planning module in ten seconds following 15 seconds of user input. For the author to prepare the same plan using a computer text editor would require over an hour. No test has been conducted to compare the system with the performance of an expert machinist, but it seems unlikely that human speed could come within a factor of ten of the automatic system.

Process plans in the AMRF standard format have four sections: (1) a header section, (2) a parameters section, (3) a requirements section, and (4) a procedure section. The LISP-readable format used for the VWS milling machine does not require a parameters section, but the other three sections correspond to the AMRF standard.

TABLE 3 PROCESS PLAN FOR ASME PART

```
(setplist 'asme_plan
 '(header (header
    plan_id   asme_plan
    design_id asme
    material  aluminum)
  steps  (steps
  1  (1   work_element   initialize_plan
          prog_name "asme demo part")
  2  (2   work_element    set0_corner
          tool_type_id    probe_0.25
          corner    1
          x_offset        0.0
          y_offset        0.0
          near_x          17.3
          near_y          7.45
          precedent_steps (1))
  3  (3   work_element
             mill_side_contour
          feature_id      1
          tool_type_id
             end_mill_1.0_2_ab
          precedent_steps (2))
  4  (4   work_element    mill_pocket
          feature_id      5
          tool_type_id
             end_mill_0.5625_2_ab
          precedent_steps (3))
  5  (5   work_element    mill_text
          feature_id      4
          tool_type_id
             ball_nosed_end_mill_0.25_4_bs
           precedent_steps (4))
  6  (6   work_element
             machine_chamfer_in
          feature_id      5
          tool_type_id
             chamfer_0.375_3_abs
          precedent_steps (5))
  7  (7   work_element    mill_straight_groove
          feature_id      6
          tool_type_id
             ball_nosed_end_mill_0.25_4_bs
          precedent_steps (6))
  8  (8   work_element    drill_hole
          feature_id      2
          tool_type_id    drill_0.316_2_abs
          precedent_steps (7))
  9  (9   work_element    machine_countersink
          feature_id      2
          tool_type_id    countersink_0.75_1_ab
          precedent_steps (8))
  10 (10  work_element    tap_thread
          feature_id      2
          tool_type_id    tap_0.375_4_abs
          precedent_steps (9))
  11 (11  work_element    drill_hole
          feature_id      3
          tool_type_id    drill_0.1719_2_abs
          precedent_steps (10))
  12 (12  work_element    close_plan
          precedent_steps (11)))
tool_requirements (
        probe_0.25
        end_mill_1.0_2_ab
        end_mill_0.5625_2_ab
        ball_nosed_end_mill_0.25_4_bs
        chamfer_0.375_3_abs
        drill_0.316_2_abs
        countersink_0.75_1_ab
        tap_0.375_4_abs
        drill_0.1719_2_abs)))
```

In the LISP-readable format, the header must specify a plan_id, the design_id, and the kind of material that is intended to be milled using the plan. Other information, such as the plan version and the name of the process planner may optionally be included.

The heart of a process plan is the procedure section. This is a list of steps to be carried out. Each step describes some operation. What is supposed to occur when the operation is carried out must be commonly understood by the planner and the controller that carries out the plan. The description is given by specifying the values of several parameters, one of which is always "work_element", the name of the operation. Each work element has its own set of parameters, but a given parameter type (tool_type, speed, feed_rate, pass_depth, and changer_slot, for example) may be used for many or all types of work elements. A summary of work elements is given in Table 4.

TABLE 4 SUMMARY OF WORK ELEMENTS

Work Element Name	Comments	Unenhanced Plan Parameters		Enhanced Plan Parameters Required
		required	**optional**	
center drill	Makes a small conical starter hole.	center_drill_depth, fptw	drs	center_drill_depth, cdfprstw
close_plan	Must be last step. Ends machining.	pw		pw
counterbore	Bores existing hole to given depth.	counterbore_depth, fptw	drs	counterbore_depth, cdfprstw
drill_hole	Drills a hole with a twist drill.	fptw	drs	cdfprstw
face_mill	Reduces height of workpiece.	See footnote 2, ptw	dors	See footnote 2, cdoprstw
fly_cut	Reduces height of workpiece.	See footnote 2, ptw	dors	See footnote 2, cdoprstw
initialize_plan	Must be first step. Starts machining.	prog_name, w		prog_name, w
machine_chamfer_in	Chamfers hole, pocket, groove, etc.	fptw	drs	cdfprstw
machine_chamfer_out	Chamfers block or island in a groove.	fptw	drs	cdfprstw
machine_countersink	Countersinks an existing hole.	fptw	drs	cdfprstw
mill_contour_groove	Mills a contour_groove.	fptw	drs	cdfprstw
mill_contour_pocket	Mills a contour_pocket.	fptw	dors	cdfoprstw
mill_groove	Mills a groove.	fptw	drs	cdfprstw
mill_pocket	Mills a pocket.	fptw	dors	cdfoprstw
mill_side_contour	Mills a side_contour.	fptw	dors	cdfoprstw
mill_straight_groove	Mills a straight_groove.	fptw	drs	cdfprstw
mill_text	Mills a text string.	fptw	drs	cdfprstw
set0_center	Sets x and y zero by probing a hole.	near_x, near_y, near_diam, x_offset, y_offset, ptw		near_x, near_y, near_diam, x_offset, y_offset, cptw
set0_corner	Sets x and y zero by probing a corner.	near_x, near_y, corner, x_offset, ptw		near_x, near_y, corner, x_offset, y_offset, cptw
set0_z	sets z zero by probing top of part.	x_loc, y_loc, offset, ptw		x_loc, y_loc, offset, cptw
tap_thread	Threads an existing hole.	fptw	drs	cdfprstw

1. c=changer_slot, d=pass_depth, f=feature, o=stepover, p=precedent_steps, r=feed_rate, s=speed, t=tool_type_id, w=work_element
2. Face_mill and fly_cut require: upper_l_x, upper_l_y, lower_r_x, lower_r_y, depth, and z_surf.

The parameters for a machining operation usually include the feature number of a feature from the design. This serves as a pointer to geometric information about the feature, such as its depth or its center. If a step is independent of a feature in a design (in a zero-setting operation, for example), then it is necessary to carry geometric information in the step.

Because of the relation between plan steps and design features, it is feasible to have a process plan for only partial machining of a design. A part made according to such a plan would have some but not all of the features of the part specified in the design.

The first step is always initialize_plan and the last always close_plan. All steps in between are machining operations. The VWS2 system currently supports 19 machining operations in addition to initialize and close, 16 for cutting metal and 3 for probing. Information about these operations is kept in the LISP environment in a "machine_ops" database.

Traditionally, the steps in a process plan have been sequentially ordered and carried out in that order. It is usually possible, however, for the steps to be executed in some other order, with only the requirement that before a given step is carried out, some set of other steps must already have been completed. These other steps are called the "precedent steps" for the given step. In the VWS2 system the steps of a process plan are numbered sequentially as a convenient method of identification, but execution is not necessarily sequential. Each step in a plan, except initialize_plan, which must have no precedent steps, carries a list of precedent steps with it. The assignment of precedent steps must be such that the close_plan step cannot be executed until all other steps have been executed.

In the LISP-readable format, the requirements list is a list of tool type names for the tools needed to make the part. The type name is not a unique identifier of a tool in the milling machine. There may be several tools or none in the machine with a given type name.

PROCESS PLAN ENHANCEMENT

In order to have a process plan be as broadly useful as possible, it is desirable that certain information NOT be in the plan and that the system which executes the plan have the freedom to make alterations to the plan. The plan is then altered (or enhanced) by the executing system just before it is executed, according to the user's desires and the workstation conditions prevailing at the time of execution. The

VWS2 Data Execution module enhances a process plan in several ways just before execution [11].

One type of data which is best omitted until execution is the changer slot number of a tool. That way a single plan may be used for many different machine setups, as long as all the required tools are in some changer slot. Slot numbers are inserted in the enhanced plan. If any of the required tool types is not present on the milling machine, the Data Execution module notifies the user of the problem and quits.

It is convenient if a single plan can be used for different initial workpieces. In the VWS, for example, if a workpiece is too tall it may be milled down to the correct height by a face milling operation inserted in the enhanced plan. Or, if a workpiece already has some of the features in the design in it, the steps in the plan that would make the features that already exist are deleted.

It is also convenient to use a single plan for different fixturings of the workpiece. For example, if a plan anticipates milling in the vise and establishing the zero value for z with respect to the vise, but the user wants to mill in the pallet area and establish z-zero with respect to the top of the part, the Data Execution module will change the set0_corner step of the plan and insert a set0_z step.

Some users like to have speeds, feed rates, stepovers, and pass depths included in a process plan, while other users prefer to omit those items and have them determined at the time NC-code is written. The VWS2 system gives the user a choice. If any of these four items is required but is missing, the Data Execution module inserts it in the enhanced plan. If any of the four items is present, the existing value is used. Only missing but required values are inserted.

During enhancement, if a step is inserted or deleted, the module renumbers the steps of the plan and changes the precedent steps of each step to match the new numbering.

If a step is inserted requiring a new tool, the tool is put into the list of tool requirements. If a step is deleted, and the tool used in that step is not used in any other step, the tool is removed from the list of tool requirements.

PROCESS PLAN GENERATION FOR THE VWS MILLING MACHINE

The VWS2 Process Planning module is a set of functions and data embedded in the VWS2 LISP environment. It takes the design of a part (in VWS design protocol format) as input and produces a process plan for milling a part of the given design. The module is fully automatic once the user has specified what it should do. The process plan is stored in the LISP environment. Optionally, the module will print out a copy of the plan in either or both of the formats described earlier. The user may edit a process plan produced by the Process Planning module by using a text editor, if that is desired.

The software which generates process plans is based on simple rules for selecting and ordering machining operations. The rules are built into the software and not kept in a separate database.

The reason that a plan can be created by relatively simple routines is that the VWS2 design protocol was specifically tailored to facilitate process planning. The key characteristics of the design protocol that facilitate process planning are:
1. It is a constructive solid geometry type system in which subtraction of primitive volumes is the only operation allowed.
2. Each primitive volume (or feature) in the protocol may be produced by one or a few common machining operations.
3. The notion of a reference feature makes it easy to partially order machining operations.

The header of a process plan is easily generated from information provided by the user, and the tool requirements list is simply extracted from the steps of the plan after the steps have been generated.

The first, second, and last steps are always “initialize_plan”, “set0_corner”, and “close_plan”, respectively. The system assumes that the part is to be made in the vise of the milling machine, and sets the parameters of set0_corner appropriately for the vise. The rest of the plan is created as follows.

Divide Design into Levels

First, the design is divided into levels. There may be only one level or many. There is no maximum, but the number of levels cannot be larger than the number of features. The first level contains all those features which have no reference feature. The second level contains all those features whose reference feature is in the first

level. The third level contains all those features whose reference feature is in the second level, and so on. If any closed loops of reference features have been put into the design, this error is detected during the process of dividing the design into levels. The ASME part has three levels: level 1 includes features 1, 4, and 5, level 2 includes features 2 and 6, and level 3 includes only feature 3.

Notice that the assignment of a feature to a level is not determined by the depth of the feature. Feature 1 on level 1, for example, is deeper than feature 6 on level 2.

The list of steps is put together by making a sublist of steps for each level, and then appending all the sublists together in level order. This assures that the reference feature for any given feature will always be made before the given feature. Since a feature always fits inside the outline of its reference feature, it is certain that there will be clear access from above to a feature once its reference feature has been made. There are other ways to get that assurance, but this one seems reasonable and has been implemented. Dividing the design into levels also reduces the problem to that of how to plan to machine a single level.

Generate Steps for Each Level

Within a level, operations are selected for making each feature in two stages.

1. The operation selection function for that feature type named in the features database selects operations for making the main feature.
2. The feature is examined for optional parameters indicating that it has subfeatures. If these parameters are found, then the operation selection function for each subfeature type selects operations for making the subfeature. Except in the case of a hole feature, each operation selector has only one choice of machining operation to make a feature or subfeature. Each operation selector calls the tool selector to select a tool to perform the operation. The actions of the 12 operation selectors and the tool selector are summarized in Table 5.

TABLE 5 OPERATION AND TOOL SELECTION

Feature or Subfeature	**Operation**	**Tool Type**
chamfer_in	machine_chamfer_in	chamfer
chamfer_out	machine_chamfer_out	chamfer
contour_groove	mill_contour_groove	end_mill /2 ball_nosed_end_mill
contour_pocket	mill_contour_pocket	end_mill
countersink	machine_countersink	countersink
groove	mill_groove	end_mill /2 ball_nosed_end_mill
hole	drill_hole /1 mill_pocket	drill end_mill
pocket	mill_pocket	end_mill
side_contour	mill_side_contour	end_mill
straight_groove	mill_straight_groove	end_mill /2 ball_nosed_end_mill
text	mill_text	ball_nosed_end_mill
thread	tap_thread	tap

Footnotes

1. In the case of a hole, if the hole is not a clean through hole, it is drilled if its bottom is conical and milled if the bottom is flat. A clean through hole is drilled if a drill of the right size is in the catalog. Otherwise, it is milled.

2. Grooves, contour_grooves, and straight_grooves are made with an end_mill if they are flat-bottomed and with a ball_nosed_end_mill if they are round-bottomed.

Select a Tool for Each Step

Once an operation has been selected, the tool selector first selects a type of tool according to the operation and feature. Selecting the type of tool is simple in all cases. Next it selects a diameter for the tool (and the number of threads per inch for taps). Finally, it looks in the tool catalog and returns the name of the tool that has the right type and diameter and is suitable for cutting the given material. If there is no such tool in the catalog, an error message is returned.

Selecting the tool diameter is simple in all cases except for making contour_pockets and side_contours. The tool diameter is determined by the dimensions of the feature for: groove, contour_groove, straight_groove, text, hole (if made by a drilling operation), and thread. The largest tool in the catalog smaller than a certain size is selected for: face milling, fly cutting, and mill_pocket (for which the upper limit is determined by the corner radius of the pocket). There is only one size for a chamfer tool (0.375) and a countersink (0.75).

For contour pockets and side contours, the largest tool in the catalog that will fit into the smallest corner of the feature is tentatively selected. Then the tool path required to cut the outline of the feature is generated and a check is made of whether the tool will cut away material it should not cut when it follows this path. If the check is OK, the selection is made final. Otherwise, the next smaller tool in the catalog is tried. This goes on until a size that works is found.

Order the Steps on Each Level

Next, the operations needed to make all the features on a each level are ordered. To minimize tool changes, the current version of the ordering function has a specific order in which it uses tool types: fly_cutter, face_mill, end_mill, ball_nosed_end_mill, drill, chamfer, countersink, tap. Within type, tools are used in decreasing diameter order.

Other algorithms for ordering the operations have been considered and could be implemented. An ordering algorithm based on operation type was implemented earlier but discarded in favor of the current algorithm. Clearly, any ordering algorithm must be sure it makes parent features before subfeatures. In some cases there is a preferred order for making subfeatures (countersink before tapping, for example).

Add Other Items to Each Step

Finally, if the user has asked to have speeds, feed rates, stepovers, and pass depths inserted (these four come as a package -- the option is called “extra items”), this is done for each step. Step numbers and precedent steps are added to the list. Precedent steps are assigned sequentially to force the plan to be executed sequentially.

LIMITATIONS

The VWS2 Process Planning module is limited in a number of ways.

Only parts whose design can be expressed in the VWS2 design protocol can have process plans generated for them.

Since the design protocol assumes that all features are made in the same side of a part, a part requiring machining on more than one side requires a design for each side being machined. A separate milling machine process plan must be made for each design, and a workstation level process plan is needed to control sequencing of the milling machine plans and handling of the workpiece between cuts. It is feasible but tedious to make very complicated parts in this fashion.

The automatic process planner will generate a process plan for any design for which tools of the right diameter (and the right number of threads per inch, in the case of taps) can be found in the tool catalog. The process planner assumes, however, that the tool will always be long enough to do the job. This is often not a correct assumption. The verification subsystem will catch errors of this sort when the Data Execution module is run. But there is no replanning capability in the VWS2 system which could try to find some other method of machining.

Except for features which are related through the reference feature hierarchy, the process planner has no notion of relations between features. If pocket A is contained entirely within pocket B (a dumb design, perhaps, but not precluded by the system), the planner will plan to make both pockets, when it would suffice to make only the outer one. If features intersect, the order of machining generated by the system may not be safe.

Although the VWS2 system is comfortable with making each subfeature of a feature in a separate machining operation (a countersunk hole, for example requires a hole-making operation and a countersink operation), and can handle counterboring and center drilling adequately, it cannot deal with other types of making features in

more than one machining operation. It would be nice to be able to make large features by first hogging then finishing. It would not be hard to add an operation like "hog_mill_pocket" to the list of machining operations, but modeling the execution of operations of this sort in the Data Execution module requires major changes to that module and has not yet been undertaken.

Tolerance requirements cannot currently be expressed in the VWS2 design protocol. It would be nice to have tolerances expressed and to vary the choice of machining operations needed to make a feature according to the tolerance requirements. Current work in the NBS process planning project, using a different design protocol, addresses this need [12].

The system does not deal with cutting forces and how they might deform or break the part during machining. For example, If a thin wall will result from milling two adjacent features, the system does not detect the wall and take measures (such as light cuts) to machine carefully around the wall.

The process planner does not prescribe fixturing. It assumes that the workpiece will be fixtured so that all steps of the plan can be carried out. The verification subsystem will detect interferences with fixturing, but, as noted earlier, no replanning capability is provided.

ACKNOWLEDGEMENTS

The NBS Automated Manufacturing Research Facility is partially supported by the Navy Manufacturing Technology Program.

Funding for the research performed by Dr. Thomas Kramer was provided to Catholic University under Grant No. 60NANB5D0522 and Grant No. 70NANB7H0716 from the National Bureau of Standards.

DISCLAIMER

Certain commercial equipment and software is identified in this paper in order to adequately specify the experimental facility. Such identification does not imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the equipment or software is necessarily the best available for the purpose.

REFERENCES

1. McLean, Charles R., "An Architecture for Intelligent Manufacturing Control", Proceedings of Summer 1985 ASME Conference, August 1985, Boston, Massachusetts, ASME.

2. Nanzetta, P., "Update: NBS Research Facility Addresses Problems in Setups for Small Batch Manufacturing", Industrial Engineering, June 1984, pp. 68-73.

3. Simpson, J., Hocken, R., and Albus, J., "The Automated Manufacturing Research Facility of the National Bureau of Standards", Journal of Manufacturing Systems, Volume 1, pp. 17-32.

4. Kramer, Thomas R., and Jun, J., "Software for an Automated Machining Workstation", Proceedings of the 1986 International Machine Tool Technical Conference, September 1986, Chicago, Illinois, National Machine Tool Builders Association, 1986, pp. 12-9 through 12-44.

5. Brown, Peter F., and McLean, Charles R., "Interactive Process Planning in the AMRF", Proceedings of a Conference on Knowledge-Based Expert Systems, December 1986, Anaheim, California, ASME, 1986, pp. 245-262.

6. Brown, Peter F., and Ray, Steven R., "Research Issues in Process Planning at the National Bureau of Standards", Proceedings of the 19th CIRP International Seminar on Manufacturing Systems, June 1987, Pennsylvania State University, not yet in print.

7. Kramer, Thomas R., and Jun, J., "The Design Protocol, Part Design Editor, and Geometry Library of the Vertical Workstation of the Automated Manufacturing Research Facility at the National Bureau of Standards", to appear as an NBSIR, 1987, 101 pages.

8. American National Standards Institute, "Industrial Engineering Terminology, Production Planning and Control", 1973, ANSI, p. 16.

9. Chang, Tien-Chien, and Wysk, Richard A., An Introduction to Automated Process Planning Systems, New Jersey, Prentice-Hall, 1985, p. 15.

10. Ray, Steven R. and McLean, Charles R., "Process Plan Flat File Format", not yet published, 1987.

11. Kramer, Thomas R., “Process Plan Expression, Generation, and Enhancement for the Vertical Workstation Milling Machine in the Automated Manufacturing Research Facility at the National Bureau of Standards”, NBSIR 87 - 3678, National Bureau of Standards, 1987, 56 pages.

12. Nau, Dana S., “Hierarchical Abstraction of Problem-Solving Knowledge”, January, 1987.