Integration of Design, Planning, and Manufacturing Subsystems
In Sheet Metal Processing

T. M. Hancock
School of Business
University of Louisville
Louisville, KY 40292

ABSTRACT

This paper describes the elements and integration of a prototype manufacturing system which achieves a fully automated link between all CIM functions from design to machine output. A generative computer-aided process planning module operates directly on CAD exchange files, supplying part process information to a master production compiler. The compiler, in turn, combines small parts into standard sheet metal sizes and then issues efficient NC process instructions to a punch and shear machine series.

Obsolete methods of planning have brought a great inertia to many manufacturing operations. The requisite paperwork has become a bureaucracy that protracts production schedules and precludes expedient response to immediate conditions of supply and demand. The cost has been enormous. High inventory levels have become established as the only alternative to poor customer service and long lead times. Long planning horizons delay the introduction of new designs and prevent a quick response to special orders. They dictate a current production schedule (the product mix, process quantity, the selection of product features) based on outdated information—educated guesses made weeks or months in the past. Worse still, this bureaucracy can forestall the adoption of new process technologies because of the investment required to update established process plans.

Manufacturing flexibility and reaction time are not simply products of machine automation, but rather of the degree with which design, planning, and operation subsystems are both independent of each other and yet integrated together.

Independence permits the separate maintenance of each subsystem; allows product designs to be modified or created without bothering with process plans; allows process rules to be upgraded without specific regard to the manufacture of individual products; and allows machine resources to be redefined without concern for their adoption within the process rules.

Integration, then, is the facility to quickly coordinate information from the various subsystems (the rules of manufacture, the latest information on demand, resource availability and product design) to effect the proper system response.

This project set out to establish a prototype of such a fully automated manufacturing system within a sheet metal processing environment. Pertinent rules of manufacture have been synthesized into a generative computer-aided process planning (CAPP) module which can determine a part's process requirements by direct interpretation of its computer design representation. A master production compiler then collects this information for all parts on the imminent production schedule, sorts it according to material specification, determines optimal layouts for given sheet sizes, and then generates the necessary machine language codes to effect their production.

The Manufacturing Environment

The Otis Elevator Company, a subsidiary of United Technologies, manufactures both elevators and escalators at their Bloomington Indiana facility. A significant percentage of the component parts used in this manufacture are formed from sheet metal, requiring processing on the plant's punch press and shear line. The centerpiece of that line (between the feeding and shearing stations) is a 45-ton numerically controlled turret punch press (Model W-4560, Wiedemann Division of the Warner and Swasey Company).

This project chose to work with the sheet metal processing unit within the Otis Division for a number of reasons: First, because of the sophisticated engineering systems already in place (the achievements of both Otis and Wiedemann engineers). Second, because sheet metal processing represents a significant industrial segment in itself, and also one where the concepts governing systems integration are broadly generalizable. And third, because establishing the rules of sheet metal manufacture would be considerably more tractable than those, say, for 3-dimensional batch machining.
Since many characteristics of both the elevators and escalators (such as dimensions, features, and finish) are specified by the customer, almost an infinite variety of part possibilities exist. Otis engineers have synthesized many of these into a smaller number of standard part types (such as the cab ceiling design in Figure 1), where certain dimensions are parameterized for later detail.

This consolidation of parts facilitates their management considerably, but creates management problems of another sort. For each part type, a BASIC language program must be written which, when called, allows the input of specific parameter values and generates the source code required of the Wiedemann software. Changes in process technology, such as the addition of a plasma torch on the punch press, force a wholesale editing of these programs. Further, special parts which do not fit a particular type require the manual generation of source code.

Consequently, the primary modification to the existing system at Otis was the addition of a prototype CAPP module which generates the Wiedemann source code directly from the CAD files. Figure 2 describes the eventual system.

The Master Production Schedule indicates the parts (as well as respective part details and quantities) that are to be produced within the next scheduling period (presumably the next shift). The compiler calls for a drawing interchange file for each part, supplying any necessary parameter values. The CAPP module then applies the rules of process technology directly to this file, interpreting closed objects within the part boundary and determining the appropriate tool and coordinate information to effect the Wiedemann source code.

The compiler sorts the parts according to material type and thickness (and perhaps by special tool requirements as well), then optimizes their layouts on standard sheet stock. When completed, this specifies the material schedule (i.e. how many sheets of each type in what order). Next, for each of those material sheets, the compiler generates the optimum sequence of punch and shear instructions.

The following sections provide some detail of the drawing interchange files, the Wiedemann source code, the CAPP module which links the two, and the general optimization logic employed by the Wiedemann compiler.

Figure 1. Cab Ceiling Design
Figure 2. System Flow Chart
The Drawing Interchange File

Each design and engineering system formats its drawing files with a proprietary compactness that speeds processing on the type computer for which it has been developed. Since these CAD-CAE systems must communicate with each other, each has the facility to translate between its internal representations and an ASCII text file called a drawing interchange (or exchange) file.

AutoCAD by Autodesk, Inc. was the CAD system used for this project. Figure 3 is an excerpt of the drawing interchange file created by AutoCAD for the cab ceiling design shown in Figure 1.

The first line maps a line segment between points (0,0) and (83.75, 0). The second line maps a line segment between the latter point and coordinate (83.75, 54.25). Lines 3 and 5 complete the rectangle that defines the part's perimeter. Line 4 maps a circle with center at (41.825, 18.75) and a radius of 5.5.

AutoCAD also has the facility to locate specified drawing portions in particular layers, so those portions can be isolated or overlaid as needed. This project established a convention for at least three layers to simplify interpretation by the prototype CAPP module. The first layer contained the physical characteristics of a typical part within the group. The second held any fixed dimensioning information, and the third layer contained either dimensions which were parameterized (e.g. L and K) or specifications dependent on those parameter values (e.g. the slots on 3 inch centers). This ensured an obvious distinction between object boundaries and such things as text or dimension lines. It also permitted ready access to what variable details required definition.

The Wiedemann Source Code

The turret of the Wiedemann punch press can hold up to 36 different tools, typically various sizes of round, rectangular, square, round-end, and so-called "special" dies. The press's programming language is called Wiedepoint V and includes pattern and scaling capabilities that are amenable to 2-dimensional punching operations.

Figure 4 offers an excerpt of the Wiedepoint source code to generate the hole pattern required of the cab ceiling shown in Figure 1. Often a particular hole can be accomplished by a single punch stroke. The press has only to rotate the turret to the proper die location, position the sheet metal so the hole center point is directly beneath the die, and then trigger the stroking mechanism which punches the hole. To effect this operation, the source code requires only the X and Y coordinates of the hole center and the turret location of the proper tool. The first line of Figure 4 indicates a hole at point (.5, 17) to be made with the tool in location 33. That tool is a .32 x 2.00 inch punch which creates the slots along the part's perimeter. A number of those slots are indicated in Figure 4.

The Wiedepoint Source Code

In Figure 4, the source code includes pattern and scaling capabilities that are amenable to 2-dimensional punching operations. The Wiedepoint source code is used to generate the hole pattern required of the cab ceiling shown in Figure 1. Often a particular hole can be accomplished by a single punch stroke. The press has only to rotate the turret to the proper die location, position the sheet metal so the hole center point is directly beneath the die, and then trigger the stroking mechanism which punches the hole. To effect this operation, the source code requires only the X and Y coordinates of the hole center and the turret location of the proper tool. The first line of Figure 4 indicates a hole at point (.5, 17) to be made with the tool in location 33. That tool is a .32 x 2.00 inch punch which creates the slots along the part's perimeter. A number of those slots are indicated in Figure 4.
Where a hole cannot be created by a single punch stroke, the process of nibbling is used. A tool somewhat smaller than the specified hole is successively punched, in an overlapping pattern to create the proper size. The Wiedepoint language first requires the coordinates of the initial punching position, as well as the turret location of the nibbling tool. These are given with a MOV command such as

\[ \text{ate the proper size. The Wiedepoint lan-} \]

4.

the fourth line from the bottom of Figure

\[ \text{4. Tool 19 is selected and the sheet metal is positioned so that its coordinates} \]
\[ (29.875,30.75) \text{are directly beneath the die. The REC command in the next line indicates that a 24 x 17 inch hole is to be created with tool 19 (specified as a 4 inch square) punching to the right and up with a maximum overlap of 3.875 inches.} \]

The second to last line positions the turret to tool 23 and aligns it with sheet metal coordinate (41.825, 18.75). The HOL command in the last line, then, is enough to effect an 11 inch diameter hole using the specified 4.5 inch diameter tool. A complex sequence of strikes will be generated so that the hole's scallop (i.e. edge smoothness) is within the .1 inches specified.

The Prototype CAPP Module

The process planning module forms the necessary link between the CAD system and the Wiedemann compiler. Its function is to read the drawing interchange file, identify each of the objects within the design, determine their method of production, and issue instructions to that extent in the form of Wiedemann source code (Wiedepoint V).

Circles are read directly as round holes, while lines must be mapped into squares or rectangles to identify right angle holes. An imaginary border around the part permits identification of any notches or cut-outs along its edge.

Next, the turret configuration is scanned to determine if any tool offers a direct match to the nature and size of each object defined. Certain special conditions are tested (e.g. concentric circles indicate counter-sunk holes which require a special tool). Also, the tool and hole orientation must match (e.g. a .32 x 2.00 inch tool cannot be used to make a 2.00 x .32 inch hole). Where no direct match is found for a hole, the nearest smaller tool is selected for nibbling.

For right angle single-punch holes, generation of the Wiedepoint code simply requires calculation of their center point coordinates (for circles this information is provided directly). For nibbled right angle holes, the initial strike position is calculated for the MOV command, and the chosen tool dimensions (with some percentage allowable overlap) are supplied to the REC command. For nibbled circular holes, a standard scallop allowance is set for the HOL command.

The Wiedemann Compiler

The punch press and shear line at Otis can handle sheet sizes to 5 by 12 feet. Needless to say, a great many of the parts to be produced are much smaller than standard material dimensions. The compiler's first optimizing function, therefore, is to arrange parts of a particular material type and thickness on a minimum number of sheets. While the specific Wiedemann logic in this regard is proprietary, the problem is typically solved with a cutting-stock algorithm [1].

Once the part layout has been defined for a sheet, the relative coordinates of each punched object are respecified in absolute terms of the sheet. Each object or node is then interconnected with every other one, defining the punch network. Arc lengths are the time value associated with traveling between linked nodes, this time being the larger of two values: 1) the quotient of the physical distance between two objects and the traverse speed of the press, and 2) the time to reposition the turret for any necessary tool change. A simple traveling-salesman algorithm [2], then, determines the optimum punch sequence for the objects.

Once all parts have been planned, the compiler issues the machine and material schedules, as well as what process instructions are required of the punch and shear stations.

Summary and Conclusion

The prototype CAPP module was tested on two different part types, successfully producing the appropriate Wiedemann source code by direct interaction with the CAD exchange files. These and other parts awaiting manufacture (the source code for the latter having been produced from the existing parametric programs) were then compiled and sorted according to material specification. Next, the parts in each material class were configured onto stock sheet sizes in layouts that would provide minimum scrap. Thereafter, the system determined an efficient punch and shear sequence for each material sheet based on its overall hole pattern and the operating characteristics of the machines. Lastly, the system issued all necessary machine language and operator instructions to guide, execute and monitor the full process.

The main contribution of this project is the method with which the manufacturing subsystems were integrated. The Otis prototype represents the ideal, where a complete separation of product design, process technology, and resource elements permits their independent specification and maintenance.

New or modified parts can be included in the next production run on immediate completion of their design.
Process changes, like the planned inclusion of the plasma arc torch (to burn rather than nibble large holes), can be incorporated by editing the selection rules within the CAPP module, rather than by reviewing and rewriting the vast quantity of part programs integral to the original system. And changes in turret configuration (new punches, the loss of existing punches, or a particular combination of punches) can be implemented immediately by simply modifying the resource data file: the list and description of available punches.

The promise of integrated manufacturing is not in the sophistication of modern hardware, but in the quality of the logic that directs it and makes it responsive to the dynamics of demand, resource availability, strategic planning and innovation.

The author gratefully acknowledges the contribution of Walt Sterneman of the Otis Elevator Division of United Technologies, Carl Watson of the Wiedemann Division of Warner and Swasey, and Kieran Mathieson of Indiana University.

REFERENCES
