METH-GEN

An AI-based Computer-Aided Process Planning System

for

Advanced Printed Wiring Board Assembly

16 February 1989

Brian Goff

EATON/AIL Systems
New York, NY

We would like to thank the following people for their contributions to making this vision a reality:

Dan Agresti
Cono Carrano
Matthew DiGiorgi
Brian Ford
Chris Hines
Stephen Howe
Steven Lecarie
Kathy Pomerenke
Ed Pines
Jeff Salvage
Gary Wladyka

We would also like to thank the management of AIL for their guidance and support throughout the project's sometimes tenuous but always exciting times.

INTRODUCTION:

AIL Systems, Inc. is a wholly owned subsidiary of Eaton Corporation. The company specializes in the design and assembly of electronic countermeasures and avionics exclusively for the Department of Defense. These systems incorporate the latest achievements in radar and microwave design and manufacturing and are typically modular in design.

As in most organizations, Engineering and Manufacturing are separately organized hierarchies. Engineering includes all functions associated with conception, formulation and analysis of the product. Manufacturing includes all aspects of the process of converting raw materials into the designs generated by Engineering. The Methods Engineering Department is tasked with maintaining manufacturing processes and procedures. This includes the generation of all paperwork necessary for the assembly of the product at each phase of the production process. The Advanced Manufacturing Technologies Department is responsible for blending new technologies with existing procedures to improve capabilities.

It is widely recognized that the benefits realized by improving capabilities are more leveraged earlier in the design/production cycle. An organization will attempt to reduce errors and waste during production by improving their design and manufacturing planning functions. The process planning function occurs strategically in the manufacturing cycle to offer significant savings in each subsequent stage of production. Process planning is a subjective function where the analyst's interpretation of design requirements, personal preference, and extent of shop knowledge can cause inconsistencies in the work instruction Packages. The potential for errors on the shop floor is unacceptable. Furthermore, the traditional labor-intensive nature of the execution of this function makes it fertile for improvement.

Meth-Gen is a comprehensive Computer Aided Process Planning project which integrates artificial intelligence and machine learning (using IntelliCorp's KEE platform), several relational databases (using ORACLE), and CAD/CAM (using Anvil). The goal is to deliver an integrated system that would reduce the overall time to produce work instruction packages for electrical and mechanical assemblies, while improving their consistency and legibility. Methods personnel are beginning to realize benefits in generating Production guidelines, standards and procedures by incorporating an integrated computer-based solution in a semi-automatic fashion.

PROBLEM DISCUSSION:

A variety of approaches to the process planning problem have been documented during the past twenty years or so. These approaches include the traditional approach, the workbench approach, the variant approach and the generative approach. Briefly, the traditional and workbench approaches use well trained planners to examine the engineering information, identify similar parts, manually retrieve previously drafted process plans or sequences of operations and adapt the old plan or sequence to meet the discrepant requirements of the new part. The disadvantage of inconsistency in these methods was the principal reason for pursuing computer-aided solutions at AIL.

The variant system is based upon the automatic identification and retrieval of a standardized manufacturing plan resulting from an established decision rule. The standard plan is a permanently established, ordered sequence of steps for a particular category. Classification codes are usually applied to identify parts with similar features. Continuously enumerating these features and codes will result in many homogeneous groups. Refinement and/or subdivision of these groups is then necessary to reflect the capabilities of the-particular manufacturing facility.

The generative model does not depend on any predefined sequence of operations. Instead, it can construct an optimal fabrication or assembly sequence through a series of sophisticated algorithms which, generally, operate with a greater level of detail than those of a variant system. Some of the tools that can be employed by generative process planners can include decision trees, decision tables, rule-based decision trees, constraint-based planning and (recently) expert systems. Generative process planning allows rapid and consistent generation of revised plans when new processes, equipment, methods and tooling are introduced in the production process.

Developing practical and/or complete solutions to the Process Planning problem has always been difficult. The principal obstacle has been the transformation of data between two very different databases; the product model database and the production database. The product model database is the collection of information produced by the Engineering processes. The production database, similarly, is produced for support of production operations. A subset of the production database is the set of process plans. These process plans, in effect, regulate subsequent production functions. It is the manufacturing engineering- department's responsibility to logically and systematically derive these process plans from the information available in the product model database.

METH-GEN AIL SYSTEMS, INC

The transformation of data from one database to the other is typically a two-stage process:

1. Extract a GENERAL process plan from GENERAL part features
2. Generate DETAILED plans from product model DETAILS

The process planning problem at AIL includes the fabrication and assembly of electrical and mechanical parts. Closer examination reveals that these tasks differ in non-trivial ways. For example, process planning for assembly of PWB's (Printed Wiring Boards) is different from process planning for final assembly configuration. For one, different tools (semi-automatic insertion machines versus screwdriver and pliers) are used. Furthermore, PWB design simplifies a two-dimensional layout problem as opposed to a three-dimensional configuration problem. The implication is that varying skill levels are required for the different tasks.

Conversely, similarities exist between the different process planning problems. These similarities can be summarized as follows:

- General rules reflect a progressive building process.
- General rules are common within a particular commodity. (i.e. Wave Solder must follow Component Prep for all PWB's, Cable A must be routed through a hole before the connector for Cable A is placed)
- Detailed rules are based on product features (i.e. Nickel-plated heatsinks must be abraded, Sub-assembly A must be installed before Sub-assembly B)

In addition to extracting general and detailed process specifications for a given assembly, the complete process planning system must deal with constantly changing product designs and production technologies. A typical program may have many Engineering Change Notices (ECN's) during its lifetime. This phenomenon is common throughout the aerospace industry as the systems themselves become more complex. This complexity usually leads to concurrent design and build efforts. Taxing the Methods Engineering department with perpetual ECN activity is also reflected by errors on the shop floor.

The ability to be flexible and to handle exceptions in a changing environment becomes a requirement for the organization operating in this mode. Systems which are built to enhance manufacturing capabilities must address this requirement. Therefore, Meth-Gen must provide facilities for accessing and updating process plans for it to be responsive.

The addition of new manufacturing capabilities always represents a change to existing 'ways of life.' The uneasiness, insecurities and other difficulties that accompany change must be addressed by the definition of the problem. Successful implementations will

require acceptance by the users of the system. The facilities that the system provides must be attractive enough to the user to overcome the fear of change.

In addition to the maintenance of the output of the system, the maintainability of the knowledge and functionality contained within the system by the users is an important concern. Here again, the rules for acceptability apply. The user must be isolated from the computer and the underlying programs as much as possible.

HISTORY:

Original proof-of-concept models of Meth-Gen were developed on a Symbolics computer by the Advanced Computing Technologies Group at AIL using the ART system from Inference Corporation. These models were useful in demonstrating the capability that the system can replicate some of the thought processes involved in PWB methodization. They were also useful in demonstrating pitfalls that we wanted to avoid in expert system development. For example, the learning curve associated with the non-intuitive ART code indicated that development costs and time would be higher than acceptable. This meant that maintenance costs for the system would be great, if indeed the system could be maintained by the users. Lastly, an excessive amount of code was needed to generate the graphics that are incorporated in the Work Instructions. This Lisp-based graphics code was very slow and inefficient.

During this period of education, we sought a low cost, entry-level expert system development tool as a basis for further conceptual definition of Meth-Gen. We ran TX's Personal Consultant Plus on an IBM PC AT and found it to be a good learning environment. We began to learn more about the interaction of frames and rules, rule chaining strategies and interfaces with both databases and the user. However, we quickly realized the limitations of small-scale tools. Specifically, the lag in system performance became noticeable even after only a few rules, frames and database queries were linked together. The ability to edit these elements and to interpret their many, increasingly complex relationships also was becoming a problem because of the limited capabilities of the tools.

It became clear that explicit control that was necessary to make the application perform as required would force us to develop more Lisp code than we originally wished. Lisp is a commonly used programming language for expert systems because of its ability to manipulate symbols well. We found Lisp to be relatively easy to learn (with computing fundamentals a prerequisite) and very powerful for quickly writing complex functions. Lisp, however, has some negative traits. It tends to run more slowly than other programming languages and has substantial CPU requirements. The nature of Lisp is to use as much computer memory as it can find before it stops itself, executes a 'garbage collection"

process, and then continues processing (this type of 'stop and copy" garbage collection forces the user to wait). Usually, this is more of a nuisance than an obstacle. The memory and CPU requirements became readily apparent as the body of Lisp code grew.

The choices for hardware and software were closely related because many of the software products under consideration were just emerging in the marketplace. The principal expert system software packages that were reviewed were Inference's ART, IntelliCorp's KEE, TI's Personal Consultant and Carnegie Group's KnowledgeCraft. The selection was limited to these packages because, at the time, these vendors had well established consulting services. We anticipated using these services to scale up the learning curve more rapidly. Later, we found that these services were unneeded, mostly because of our rapid understanding of the tools and the availability of internal resources.

We used this knowledge, along with the extensive consultations with the various vendors, to settle on KEE as the expert system development tool for Meth-Gen. The features of KEE that were pivotal in our choice were KEE's graphic representation of knowledge bases, full support of both frame and rule knowledge representation structures, the effective use of 'Active images' and the "Tell and Ask" natural idiomatic syntax of rule coding.

Active Images are prebuilt, fully functional input and output objects such as pushbuttons, gauges, meters and display windows. Simply put, they allow the user to input information and to monitor the state of the system by just using a mouse (instead of the keyboard). By attaching Active Images to a slot in a frame, a method function (different from Methods Engineering) can be triggered when the button is pushed. A method function is a Lisp function that can contain any common programming construct (i.e. loops, conditionals, assignments, etc.) as well as embedded KEE functions, calls to other Lisp functions, calls to the operating system and calls to external functions. Since KEE functions provide access and control over all objects in the KEE environment, and, since all Active Images are KEE objects, control over these images is also exercised through method functions. These features were combined with IntelliCorp's position in the marketplace to give the company its position in our CIM plans.

Similarly, the importance that strategic value played in the decision of a database product narrowed the choices to ORACLE and DEC's RDB. The idea was to invest in a database product that would satisfy the requirements of future applications beyond Meth-Gen. The most important functional factors in this decision were the ability to access data from a programming language (C or Lisp) and SQL capability. Ultimately, the capability, portability and support available with the ORACLE product caused it to be the product of choice.

ARCHITECTURE:

We can now begin to formulate an architecture of the process planning system. Three functional areas were immediately apparent:

1. Expert system
2. Database
3. Graphics

To design Meth-Gen for extendibility to other commodities, we sought to modularize as many of the working components as possible. This modular approach would allow us to "remove' and "install" the appropriate chunks of methodization knowledge upon demand. in this fashion, we could build a library of functions that would help to standardize the user, application, file and system interfaces.

Object-oriented programming, by nature, supports a modular programming style. In addition, KEE offers a variety of tools that facilitate knowledge aggregation in the form of knowledge bases. As a matter of fact, the KEE package is built in the same fashion; a set of 'System KB's" that logically support and group the software's various capabilities. These functions allow the developer to create new knowledge bases, define frames and slots (also called objects and attributes), define methods for retrieving attribute values and link Active Images to monitor attribute values. Extensive facilities to define and edit rule-structures, forward and backward chaining strategies and complex reasoning are also provided.

Using these tools, we separated the methods engineering expertise into the following groups:

1. Parts
2. Manufacturing Procedures (Instructions)
3. Agendas (Process Flows)
4. Rules

Each knowledge base contains a taxonomy, or classification, of logically related entities. For example, displaying the Parts Knowledge Base will graphically illustrate the breakdown of parts as such:

[Figure of binomial tree structure(s)]

Each entity in the knowledge base (in this case, parts) 'inherits" the attributes, or features, of the class of parts it belongs to. The other knowledge bases are defined similarly. In this manner, the methods analyst can alter a part's tooling or fixture number (in the Parts Knowledge Base), or the content of a manufacturing instruction (in the Procedures Knowledge Base) by merely changing the value of the attribute of the entity that represents that part or instruction.

The need for the database became apparent when we uncovered the many part features that are considered in the PWB methodization process. Both assembly-specific and part-specific data elements are used in the decision-making processes. Assembly-specific data is run-time specific and would require only temporary storage in the database. Part-specific data refers to the features of the parts that indicate alternative production methods during methodization. This feature data rarely changes and would be stored permanently in the Meth-Gen database.

Obviously, we had a requirement for data elements to be transferred between the expert system and the database. Ideally, we would extend Lisp calls to make direct database queries. Unfortunately, few database systems support Lisp access capability. However, many databases (including ORACLE) allow queries from functions embedded in programs written in the C language. Unfortunately, the only data structures that can be easily passed from C to Lisp are simple pointers or numbers (Intellicorp markets tools to effectively embed SQL database queries in KEE code, however, we felt that the cost of this capability was too prohibitive). For the sake of expediency, we chose to write C programs that would create flat textual files that could be read by the Lisp programs. These C programs contain SQL statements that are pre-compiled by the Pro-C compiler from ORACLE. The C executable program files are then called on demand via Lisp calls to the operating system.

The development of these C programs was difficult because of the requirement to make them independent of the data that was being handled. Specifically, we intended to expand data tables in the data base as part of normal Meth-Gen maintenance. During processing, some of these programs will generate SQL statements "on-the-fly" depending upon the current context of the query and the current contents of the database. Since production use of the system was a design requirement, these C programs had to be written so that the user was isolated from the activity performed on the database.

The graphics module operates relatively independently from the expert system and the database. The output of the expert system is fed to the graphics module for processing. This is the first point at which the analyst will see the work Instruction Package. He will now edit the package using a customized graphics editing environment. A set of functions are provided through a common CAD-type input tablet. The editing phase consists of construction and adding visual aids and notes to the Operation Sheets. Upon approval, the analyst will print the final copy of the Work Instruction Package for distribution to the shop floor.

USING Meth-Gen:

Meth-Gen functions as an automated assistant. The analyst is first presented with a main panel from which the different phases of PWB methodization are accessible. These phases include:

- Initiating the methodization process
- Retrieving as much data and information as possible
- Considering the manufacturing processes step by step from a predefined list of operations
- Identifying the necessary operations and associating the relevant manufacturing instructions
- Creating the output
- Reviewing and editing the output
- Printing the final documentation

Meth-Gen is required to act as an assistant for several reasons. First, thousands of parts can be used to produce printed wiring boards. While group technology concepts can be applied to create generalizations, exceptions will always occur. The system must be able to incorporate Exception information from the analyst during the session and respond to it correctly. Secondly, not all the information needed to perform complete methodization is available in the Meth-Gen database. In these cases, the analyst must refer to engineering drawings and visually interpret many things. For example, Meth-Gen needs to know if ink appears on any metallic surfaces on the assembly. The response to this question will influence the manufacturing process for that assembly. Finally, the expert system cannot and should not replicate every aspect of the analyst's capabilities. New insight and ideas about what should be and what should not be included in the system, as well as how the system itself is utilized, should be encouraged and considered.

The analyst begins using the system by requesting a Parts List for the assembly to be methodized. The assembly-related data such as the quantity to build and the current engineering revision level is first entered into Meth-Gen. Meth-Gen prepares the request which is transmitted to the Engineering DEC20. The Parts List is retrieved from the remote database and returned to the Meth-Gen system. Requests can be made for more than one assembly at a time. The Parts List is, in effect, an Engineering Bill of Materials.

Upon receipt of the Parts List from the DEC20, Meth-Gen will process the file to remove extraneous spaces, parentheses and irrelevant characters as it loads the data in the Meth-Gen database. As its next step, Meth-Gen will begin to match each part on the Parts List with the feature data that is permanently stored in the Meth-Gen database. Feature data will include items such as the reference designation of the part, number of pins on a DIP, the center-to-center distance of the leads on a discrete component, tooling and fixture numbers and the respective class of parts in the Parts Knowledge Base.

If any required feature data item is missing but Meth-Gen can recognize the part family, it will ask the user to supply more information about the part. It will only ask the user for feature data germane to that category of part. It does this by presenting the user with a context sensitive ORACLE form. If a given part is unrecognized, Meth-Gen will ask the user if it should remove the part from further consideration by retaining it in the category of 'Unrecognized Parts", and thereby continue with the methodization process. If the user does not opt for this option, Meth-Gen aborts the methodization process until the part is filed in the permanent database.

Once all the parts have been reviewed for completeness of data, Meth-Gen will call a C program that will construct a large, comprehensive data file. This data file is then read by the expert system and each part, and its feature data is installed in the Parts Knowledge Base. Meth-Gen will now look for the presence of another Engineering data file. This assembly-specific data file, if present, will contain coordinate information for each component on the printed wiring board assembly. The information contained in these files is extracted directly from the Engineering Computer Aided Design system. We call this file the "LNL Data File" because it contains the Location Net List that will be later used by the methods analyst to incrementally construct visual aids that will appear on certain Operation Sheets. If found, the records from this file are read into the expert system and matched to the reference designation of the parts previously installed. If this file is not found, the methodization process will continue without the benefits of semi-automatic visual aid generation.

During the next phase of the methodization, Meth-Gen will begin to ask questions to ascertain the necessary information not provided by the feature data. The questions asked are guided by the data thus far provided to Meth-Gen. For example, Meth-Gen will need to identify and group certain types of parts. These groups form what we call Part Sets. Parts that make up a Part Set will be removed from the standard process flow and applied at a different manufacturing operation. A good example would be a capacitor that requires sleeving over the leads. Meth-Gen would recognize the presence of both the capacitors and the sleeving from the Parts List and would ask the user to define a Part Set. These parts would then be removed from the standard Semi-Automatic insertion operation and placed in a manual Assembly operation.

When Meth-Gen has accumulated all the information it needs, it will begin to consider the manufacturing process step by step. A set of rules associated with each manufacturing operation will determine if that operation is required. If so, the rule will further conclude which manufacturing instruction is retrieved from the Procedures Knowledge Base. Meth-Gen continues to identify the parts and instructions relevant to each operation in the manufacturing process until all operations have been considered. At this point, the analyst can continue to the next phase of methodizing this assembly, or he can begin to methodize another assembly. By continuing to methodize the same assembly, the analyst will now enter the graphics module of Meth-Gen.

The graphics module is utilized in several ways. The first step is to process the output of the expert system. The expert system's output during the methodization process is a collection of GRAPL (GRAphics Programming Language; MCS's proprietary programming language used to manipulate entities in the ANVIL 5000 environment) program files. Each program file contains the GRAPL code that will construct an Operation Sheet. In addition, Meth-Gen will create a GRAPL "control" file and a keystroke file for each assembly it methodizes. These files control the loading and compilation of the individual page-specific GRAPL files.

Once the GRAPL files have been compiled, Meth-Gen will contain the basis of the Work instruction Package in one file. The user will now edit this file to add visual aids and assembly-specific notes. The user has a customized editing environment that is driven by. GRAPL programs are accessible through a tailored tablet pad. These functions support a variety of functions such as scrolling through pages of the Work instruction Package, editing textual notes on an Operation Sheet and developing visual aids which consist of images of the PWB in various stages of assembly. Visual aids are constructed by executing other files that were created by the expert system. These files were built using the information obtained from the Engineering "LNL Data File." They will retrieve predrafted part images for the parts called out by Meth-Gen for a given operation and will locate them correctly with respect to the circuit board. Presently, this library contains 385 images of electrical and mechanical parts.

DEVELOPMENT EFFORTS:

Meth-Gen currently contains more than 3500 lines of Lisp, C and GRAPL code. The expert system contains more than 440 "static" frames. These are frames that are permanent members of the knowledge bases. At run-time, anywhere from 2 to 200 frames can be "instantiated" (instances of a class are created). Each of these frames represents a part on the Parts List (as described earlier). We have 35 rules that represent each manufacturing process used in the production of PWB’s.

Development of the code for each of the Meth-Gen modules began in September 1987 following a two-week training course in using KEE. The training course was attended by the full-time knowledge engineer (Brian Goff) and the full-time domain expert (the part-time knowledge engineer was already familiar with the product). The first substantial prototype of the expert system was demonstrated six months later in February 1988. (Actually, a working expert system was ready several weeks before this. The demonstration was delayed bringing the database functionality up to par.)

In July 1988 we decided to move the system into the production environment for beta-test. We felt that this would be useful to temper some of the anxious anticipation of the system as well as to begin bringing the users up to speed. During the next six months we encountered a great deal of development-related issues. These were requests by the users to incorporate additional functionality that was previously unforeseen. For example, we found that significant savings would be leveraged by coding tasks that were expected to be handled in the graphics editing phase. Consequently, a great deal of functionality was added to the system during this period. Meth-Gen reached the point where it had demonstrated savings and was committed to production use as of January 1, 1989. Currently, there are two workstations and two methods analysts trained in the use of the system.

The most prominent obstacle encountered during the implementation phase of Meth-Gen was in labeling it a 'production system.' The maintenance of the system has consistently required frequent attention by a group of highly skilled engineers (members of the development team). This, in effect, has created a layer of intervention between the expert knowledge and the users of that knowledge. To adequately satisfy the design goal of delivering a system to the Methods Engineering department, the system's implementation must progress from "controlled" use to "production" use by people with a very different skill set.

The high level of attention required by Meth-Gen is directly due to the high level of complexity that is hidden from the user. This level of complexity is not unusual in expert systems of this magnitude. The most influential factor that determines system complexity is the problem or application itself. It is often quoted that the decision to use an expert system to solve a problem requires the selection of an appropriate problem. The process planning problem, as mentioned earlier, is not a trivial problem.

To understand where this complexity is derived from, we can consider the development of an expert system more closely. To build the expert system, the knowledge engineer must first study the domain and the expert's knowledge on a level that is independent of the details associated with the expert system's implementation. The knowledge engineer must choose a mechanism appropriate for the representation of the kind of knowledge. The choice of the knowledge representation mechanism is driven by how easily it allows knowledge capture. In addition, the more direct that the expert's knowledge is mapped into the representational structure, the greater the validity of the implementation will be.

In Meth-Gen, we have used frame representation extensively to model certain kinds of knowledge about the PWB methodization process. The efficiencies experienced are reflected by the fact that the thousands of parts are reduced to several groups (simple group technology) and that each group will implicitly inherit a set of features. Consequently, Meth-Gen knows immediately what features to expect for a given part in each class of parts. Furthermore, the great deal of knowledge included in these structures means that only 35 rules were required in the system. The interrelationship of the rule structures is unimportant to Meth-Gen. If the rule is written with the proper conventions, Meth-Gen will consider it during any methodization session.

The complexity of Meth-Gen stems from the great deal of Lisp code that ties together the frame and rule representation structures. The Lisp code is used to represent the procedural aspects of the methodization process. For example, many of the repetitive tasks that Meth-Gen automates include retrieving the Parts List, recognizing parts and part features, generating the format of the Operation Sheet and entering information in various sections of the Operation Sheet.

SUMMARY:

There are several issues to review in summary: modularity and extendibility, robustness and benefits. In each of these cases we can point to what we feel are successes as well as areas where we would have hoped for greater success.

In the area of modularity and extendibility, we have successfully captured and modularized a great deal of the heuristic knowledge used to methodize Printed Wiring Boards. For example, the knowledge base of manufacturing instructions is independent of the parts used in production. This means that the Procedures Knowledge Base can be removed, modified, expanded or applied to a different parts taxonomy. Similarly, the Agendas (Process Flow) Knowledge Base and its underlying mechanisms and methods can be extracted and used with completely different Parts and Procedures Knowledge Bases.

The Meth-Gen database is constantly expanding to include new parts. These parts are not restricted to electrical parts. In the course of time, this database of part features will grow to be an asset to AIL. The customized graphics tools that the methods analyst has available to him are also not restricted to the development of PWB Work Instruction Packages. These tools will be extended to other applications in the Methods Engineering Department.

Another area where we experience the benefits of modularity is the relative independence of each phase of processing during PWB methodization. An analyst has the flexibility to retrieve several Parts Lists from Engineering at once if, for example, he is aware that the network will be down for maintenance. He can continue to methodize these assemblies without interruption. If the expert system or the database is unavailable because of updates to the software or new development activity, the analyst can continue to edit Work Instruction Packages that have reached the stage of graphic editing.

An area where the extendibility of the Meth-Gen modules has been disappointing is in its extendibility to three dimensional complex assemblies. We are currently trying to understand how the methodization process for these complex assemblies can be modeled after the processes used by PWB methodization. This will be done by reviewing the similarities of methods described earlier. Also, the great deal of Lisp code will have to be reviewed to understand the magnitude of modifications.

The related topic of robustness addresses the completeness of the solution. We have successfully captured enough knowledge to methodize any given Printed Wiring Board built at AIL to approximately 90 percent completion in a matter of a few hours. The remaining 10 percent of effort is in the final editing of the Work Instruction Package using the graphics tools. The fact that this small fraction of incompleteness takes almost as much time to process as the initial phases of methodization indicates the relative inefficiency of the task. At this point, we have reached the limit of the level of functionality that makes sense to incorporate in the expert system. This forces us to rely on the analyst's CAD skills while we seek improvements in the Work Instruction Package editing phase. Note that this is not necessarily without value because these skills will be employed when the graphic tools are used for other applications.

The benefits in using Meth-Gen are beginning to become apparent as the work instruction Packages are being distributed to the shop floor. Preliminary evaluation indicates improved throughput in the methods Engineering department because of a reduction in cycle time. Each Work Instruction Package is being released with more consistent content, regardless of the analyst who created it. We have seen that small changes can be made to an Operation Sheet relatively quickly. Substantial changes are also responded to quickly by completely re-methodizing the assembly (it is sometimes advantageous to utilize Meth-Gen's efficiency by completely re-methodizing the assembly). Most importantly, the quality of the Work Instruction Package that was modified to incorporate an ECN is now the same quality as the original Work Instruction Package. The result, which is the desired result, is that we will see fewer interpretation errors on the shop floor.

Meth-Gen has a variety of opportunities which can now be pursued. It can be used as a front end to a paperless shop environment since its output is a document in electronic form. It can be linked to the Computerized work measurement System to automatically retrieve setup and run times for the production processes it calls out. We can incorporate these standards in a sophisticated simulation of the production environment based upon the fact that Meth-Gen knows each assembly's process flow. This can also be expanded to allow Meth-Gen to generate optimal assembly routes based upon real time data from the shop floor. And, of course, we expect to expand the use of Meth-Gen to other programs and commodities.