The HICLASS Software System: 
A manufacturing expert system shell

David Liu, Ph.D.
Hughes Aircraft Company
U.S.A.

ABSTRACT

Several expert system applications are being used at two of the Product Operations Divisions of Hughes Aircraft Company's Electro-Optical and Data Systems Group to aid in the engineering-to-manufacturing cycle of circuit board design and production. This paper focuses on the evolution of the Hughes Integrated Classification (HICLASS) Software System and its applications.

What is HICLASS?
The Hughes Integrated Classification (HICLASS) (“Softex”) Software System is a general purpose expert system shell tailored for automating "white collar" (i.e. support) activities in manufacturing. HICLASS is a convergence of artificial intelligence (AI), man-machine interface, graphics and database technologies.

The white collar activities — that HICLASS was specifically designed to automate — involve decision making. Decision making activities in manufacturing include a broad spectrum of manufacturing support activities such as planning, methods, processes, standards, cost estimating, scheduling, producibility analysis, etc. The purpose for automating these activities is to strive for consistent and timely judgment in today's dynamic manufacturing environment.

HICLASS was developed at the Tactical Product Operations Division of the Electro-Optical and Data Systems Group, Hughes Aircraft Company. The system especially excels in CAD/CAM integration applications such as generative process planning. The guiding concept behind HICLASS is that integration and automation of manufacturing practices and procedures using an intelligent computing mechanism can substantially enhance the capability of manufacturing enterprises.

To create a better understanding of the scope of issues related to HICLASS, this section is structured to include a description of both the concept and the actual implementation of the system. These general baseline descriptions are intended to provide a helpful framework in which to facilitate appreciation for the subsequent research and development.

The Evolution

Historical setting

The main thrust of CAD/CAM at Hughes is aimed at shortening the design cycle and improving the manufacturing process. One way this can be achieved is by automating process planning since the traditional manual process planning method is both time-consuming and labor intensive.

The desire to overcome drawbacks of the manual process planning system motivated Hughes to begin a Computer Aided Process Planning (CAPP) effort in May of 1981. The advantages of CAPP systems over manual process planning is clear. In a recent survey on CAPP undertaken by Martin Haas and Dr. T. C. Chang at Purdue University, the results showed that 23 percent of the companies who are using CAPP systems experienced time savings between 75 to 100 percent, 38 percent experienced 50 to 75 percent, and 39 percent experienced 25 to 49 percent [Dr. Richard Liu, 1987].

Since its conception in 1981, HICLASS has evolved through several evolutionary stages — computer aided, generative and automated — which resulted in a series of progressively refined systems: HICLASS 0, I, II and III. In this evolution, it is important to note that HICLASS has evolved from an application specific shell (namely, process planning) to the current general shell.

HICLASS 0

The first version of HICLASS (0) was a decision tree-based system. The heart of HICLASS 0 was the DCLASS (Decision Classification) Information System, a FORTRAN based software package that was originally conceived by Dr. Dell K. Allen and Ronald Millett at Brigham Young University back in 1976.

DCLASS is founded on the disciplines of group technology (GT) — the doctrine of sameness — thus, its information processing techniques are streamlined for solving classification problems. DCLASS uses tree structures to classify, store and retrieve information. In 1978, the first application and installation was made at Boeing Aircraft for the purpose of generative process planning for aircraft parts.

Hughes supplemented DCLASS with internally developed software to perform support functions such as design interpretation, file access, planning calculations, route sheet formatting, and report generation. The internally developed software was written in a mixture of SPL (an Algol-like system programming language especially designed for the HP3000 supermini-computer) and FORTRAN.

The two main objectives of HICLASS 0 were to gain knowledge in CAPP and to prove the feasibility of generative process planning. The notable achievement of HICLASS 0 was the successful demonstration of generative process planning — with the main emphasis on routing as opposed to detailed work instructions — for wiring board fabrication, printed wiring board assembly, and rotational parts.

HICLASS functioned in accordance with the three fundamental phases of process planning:
- Interpretation
- Reasoning
- Presentation
Interpretation is the problem definition phase. During this phase, part features are extracted from a computer-aided engineering (CAE) data base. These part features are the essential elements of the engineering design that fuels the reasoning phase. Initially, there were attempts to automatically infer part features from their geometric representations. However, it turned out to be more efficient to have the engineering design explicitly supply the part features.

Reasoning is the problem solving phase. Manufacturing logic is used to balance the requirements of the product against the constraints of the available manufacturing resources. Essential manufacturing steps are selected and work instructions are constructed for each manufacturing step. The analysis and evaluation continue until the process plan is complete.

In HICLASS I, the manufacturing logic embodied within the decision trees was collected via a combination of interview and literature search processes. Of the three initial manufacturing applications, the printed wiring board fabrication was, by far, the most refined. This success can be attributed to the active participation of the printed wiring board fabrication experts in providing input to the prototype development team, which was mostly made up of computer scientists.

Presentation is the solution packaging phase. It is an important step, because its result is what the users see. Just because a system is able to reason effectively does not directly imply that the ability to express the results of that reasoning is equally powerful. Thus, the output presentation modules have been isolated to receive special attention. Presentation requires a special set of software dedicated to ensure continuity, fluidity, conviviality and especially comprehensibility. Man-machine interface and human engineering factors are paramount.

As HICLASS I matured, the internally developed software grew disproportionately large compared to DCLASS. In addition, the input analysis modules and the output formatting modules had already taken over a majority of the decision making functions in order to circumvent the limitations of DCLASS. Some of the limitations encountered were inherently difficult to overcome because they were intrinsically entrenched within the implementation language of DCLASS, FORTRAN itself.

The major problem that was encountered which could not be circumvented was the approach of translating manufacturing process flow into decision trees. Manufacturing process flow diagrams are usually drawn as networks. A network is an unstructured representation, whereas a decision tree is a hierarchical representation. As such, mapping a network into a decision tree is analogous to the projection of a set into its subset. It was a relatively cumbersome procedure which caused the representation to lose some integrity in the translation. This shortcoming was intrinsic to DCLASS.

At that point (February 1983), the decision was made to create a more robust software engine to overcome these shortcomings, with a special emphasis on network manipulation. The new software engine would be written in a structured high level language which provides pointers and data structures. Pointer and data structures provided the essential functionality, modularity and flexibility for "real world" data representation.

**HICLASS I**

HICLASS I was a complete departure from DCLASS. It was founded upon the disciplines of artificial intelligence as opposed to group technology. The DCLASS subroutines and the original HICLASS executive, a network-oriented, rule-based interpretive expert system shell. Unlike HICLASS 0, the entire HICLASS I Software System was developed internally at Hughes.

The major differences between HICLASS 0 and HICLASS I were in the techniques used for knowledge representation and reasoning. HICLASS I employed networks and production rules, rather than decision trees, to represent knowledge. To reason, HICLASS I utilized declarative, as well as procedural knowledge.

**HICLASS I Software System Architecture**

HICLASS was also a general purpose logic processing software system. HICLASS I was a culmination of ideas gathered from systems such as DCLASS, SLAM II, K5300 and FLEXPLAN. SLAM II (simulation language for alternative modeling) is a FORTRAN based simulation language that supports network, discrete event and continuous modeling. SLAM II is a product of Pritker and Associates, Inc. K5300 is a LISP based expert system shell, which is a refined version of EMYCIN. K5300 is the fore-runner to M1 and S1, which are all products of Teknowledge Inc. FLEXPLAN is a decision support system designed to simulate and optimize the operation of a flexible manufacturing system (FMS). FLEXPLAN was jointly developed by Charles Stark Draper Laboratory, Inc. and Hughes Aircraft Company under an United States Army Tank Automotive Command contract.

The network orientation was an adaptation of the SLAM II network modeling technique. The network orientation supports modularity which imparts orderliness and manageability to applications. A network can also depict the importance of precedent and sequent relationships. Networks can also be nested. The HICLASS networks are used to model both the manufacturing process flow and to partition the rules into clusters. A nested network of rule clusters offers the benefit of combining the best features of the classification model — which is fundamental to group technology and DCLASS in particular — and a production system.

The idea for employing the rule construct and it's interpretation process was triggered by exposure to rule based systems such as K5300. The rule construct was attractive because it seemed to offer a more natural way of expressing knowledge than decision trees. Furthermore, the non-procedural characteristic of a rule based system was well suited for tying together seemingly disjoint chunks of manufacturing logic. The non-procedural characteristic helped to provide the cohesiveness that was largely missing from the information obtained through the interviewing process.

For instance, an experienced process planner (domain expert) can easily offer expert advice on how to plan for a particular part feature (an isolated problem). However, the cumulative effects of all the part features (the total problem) are usually not as simple as sequentially applying each piece of offered advice. The non-procedural characteristic of a rule based system provides the means of automatically analyzing the interaction between each individual piece of advice.
HICLASS I did not offer interactive viewing of its working memory during its inferencing process. To augment this limitation, a function called "snap" was made available to provide snapshots of the working memory. The knowledge engineer determines when to conduct the snapshots and the results are placed into a user-defined file. The "snap" function made it possible to "warm start" a scenario from the middle of its inferencing process. "Snap" and "warm start" allow the knowledge engineers to restart a scenario from mid-stream, thus eliminating the time-consuming process of replaying the already debugged partitions of the production rules. Together, these two help shorten the iterative cycle required to refine a knowledge based application. The "warm start" feature was an idea borrowed from FLEXPLAN.

The major achievement of HICLASS I was the utilization of artificial intelligence for generative process planning. Two other notable achievements were script-driven graphics generation and on-line color graphical instruction. The script-driven graphics generator worked concurrently with the inference engine (i.e., HICLASS executive) to automatically generate full color pictorial manufacturing instructions. Internally developed graphics converters, PLOT-10 graphics standards and the Interactive Graphics Library (IGL) graphics utility from Tektronix were used to support the color pictorial features.

HICLASS I could accommodate imported graphics from CAD systems in either vector or IGES format. The vector format, an internally developed method for transporting graphics was necessitated by the absence of IGES processors in some CAD systems. IGES is the Initial Graphics Exchange Specification format and the standard in the industry. HICLASS' internal IGES converters have the capability to process lines, points, arcs, B-splines and subfigures, adhering to the IGES version 1.0.

HICLASS I was written in PASCAL and originally resided on the HP3000. However, as preparations were made to move HICLASS I into the production environment, new hardware/software configurations were selected and installed to assure that reliability, performance, and cost requirements were adequately satisfied. The new configuration included: powerful Apollo engineering workstations, the versatile UNIX™ operating system, the Apollo Domain token ring network, Tektronix Unicorn Series color terminals, customized multiplexer hardware/software, and customized man-machine interface devices.

As the users began to extend and expand their applications, the functional requirements of HICLASS grew. Additional knowledge representations and manipulation capabilities were also necessitated by the continuous ideological push toward the minimization of manual intervention and the strive for total automation. At the same time, there was a strong desire to decouple HICLASS from its process planning application in order for it to become a general purpose expert system shell.

The three main components of HICLASS I (input interpretation modules, the HICLASS executive and the output presentation modules) were cohesive but not uniform. Only the HICLASS executive could process logic in rule form. The logic of the input interpretation modules was embedded into the source code itself while the logic of the output presentation modules was encoded in scripts. There existed a strong desire to lift the logic from both of these modules onto a higher level of abstraction — like production rules in particular — so that it could be uniformly represented.

In the process of moving HICLASS I from the HP 3000 supervisory computer to the Apollo engineering workstation, it was discovered that PASCAL is not uniform across machines. For instance, the Apollo PASCAL did not offer the string manipulation functions that were available on the HP 3000. Although the problem can be rectified by augmenting the language with additional procedures, symbolic processing is greatly hampered in the absence of a native string manipulation capability.

Motivated by the need for additional functionality, the desire to become more generic and the necessity to unify the software modules, the decision was made (in May 1984) to rewrite HICLASS on a more portable platform, i.e., "C" on UNIX. The "C" language was chosen not only for its display and portability, but also for its efficiency in handling pointers and data structures.

HICLASS II

HICLASS II was the second generation of the network-oriented, rule-based interpretive version of the HICLASS Software System. The major differences between HICLASS I and HICLASS II lied in the computing platform, underlying language, software architecture, software construction and the spectrum of knowledge representation methods. Furthermore, the application specific functions were transferred to auxiliary support modules. Thus, HICLASS became a general purpose expert system shell. Like HICLASS I, HICLASS II was written internally at Hughes. HICLASS II operated in the Apollo Domain environment, rather than the HP3000 environment. It was written in "C", rather than PASCAL. HICLASS II was constructed with UNIX software tools such as lex and yacc. Many user-callable procedures, such as set and list operators, were added. HICLASS II was an open-ended system, accommodating user-defined procedures to augment the native HICLASS procedures.

A user-defined procedure is an executable program that is written by a user to perform a certain task within the HICLASS network. A procedure might display information on the display terminal, generate external data or perform an operation that the native HICLASS procedure could not. A user-defined procedure could not, however, be used to return values to the HICLASS rule.

HICLASS II was designed with a distributed architecture. It separated knowledge management functions from control functions. The knowledge management functions were contained within a module called the Experience Table (ET). The Experience Table was primarily responsible for managing the working memory (e.g., loading, updating and retrieving data). The control functions were contained within the Rule Interpreter (RI). The Rule Interpreter was primarily responsible for flow of control, context switching, selection of production rule partitions and rule interpretation.

The concept of a distributed architecture was sparked by HEARSAY II's backboard mechanism and its support for multiple knowledge sources. The distributed architecture was designed to support multiple Experience Tables and multiple Rule Interpreters, which is analogous to multiple knowledge sources in HEARSAY II. Unfortunately, the actual implementation of HICLASS II never had the opportunity to reach that stage.

The knowledge representation of HICLASS II was extended to include frames. Ideas on how to implement frames were gathered from studying other frame based representations system such as SRL, the Schema Representation Language developed at the Intelligent Systems Laboratory of the Carnegie Mellon University. Furthermore, explanations obtained from Dr. Stephen Smith and Dr. Mark Fox of Carnegie Mellon University provided additional insights.

The information stored in the working memory of HICLASS during the network execution is known in HICLASS as the meta-structure. A frame in the meta-structure is made up of a group of related attributes called slots. Each slot represents a piece of information associated with that frame. When there are values that are not associated with any frame, they are known as HICLASS variables.
although frame based systems' representation are fundamentally
different from that of traditional production systems. HICLASS'S
network orientation was flexible enough

The main objective of HICLASS
was to develop an expressive
expert system shell robust enough to allow a wide range of
manufacturing applications. The notable achievements of
HICLASS I1 were the modularity of its design which separates the
nodes of a network and

The rule construct was extended to facilitate
the manipulation of frames, frame instances and slots variables. In addition, HICLASS II
employed a more English-like syntax. Ideas on how to
implement the rule syntax extension were gathered from studying
ROSIE (Rule-Oriented System for Implementing Expertise)
developed at the Rand Corporation. Dr. Henry Sowizral's
suggestions were also extremely valuable.

With these extensive enhancements added to its inventory of
knowledge representation techniques, it was not feasible to retro-fit
the HICLASS I knowledge bases. The HICLASS I applications
were rebuilt under the auspices of HICLASS II. In most cases,
there was a 5:1 reduction in the number of rules due to the power
and expressiveness of the new representation.

The main objective of HICLASS II was to develop an expressive
expert system shell robust enough to allow a wide range of
manufacturing applications. The notable achievements of
HICLASS II were the modularity of its design which separates the
generic from the application modules, the 20:1 increase in graphic
display speed, rule-driven graphics generation, and rule-driven
route sheet generation.

With the onset of putting HICLASS into the demanding production
environment, the issue of overall system performance became
paramount. For HICLASS II, it arrived at a trade-off point
between expressiveness versus overall system performance. Each
side represented differing sensibilities and approaches towards
optimization. Nevertheless, the main performance drawback was the
lack of speed.

In addition to the fact that the system was large and complex, there
were two other major factors that contributed to its poor performance:
- Implementation of the distributed architecture
- Process of rule interpretation

The solution was obvious. The way to increase the speed of an
interpretive language is to compile it. However, since HICLASS is a
rule-based system, many of its abilities are intrinsically tied to the
interpretive nature of the system, thus, deferring many decisions
e.g. variable references and procedural binding) until run-time.
There were many hurdles to overcome in order to develop a
compiled system. Nevertheless, it was accomplished through the
exploitation of the powerful pointer capabilities available in the "C"
programming language.

HICLASS III operates in two distinct phases: the translation phase
and the activation phase. In the translation phase, rules and frames
are translated to source code of "C" statements. The translated
source code of "C" statements are then compiled and bound with the
run-time library to form a highly efficient run-time object code.
It is during the activation phase that the object code is operational,
i.e. evaluating/executing the logic of the application's rules.

The inspiration of HICLASS III was spurred on by our fascination
with the object oriented paradigm. HICLASS III was patterned
after the functionality and architecture of the Smalltalk-80 system.
The functionality encompasses the various facets of an object
oriented system and the architecture adheres to the layering scheme
of the Smalltalk-80 virtual machine. Smalltalk-80 was developed
by the Software Concept Group at the Xerox Palo Alto Research Center.
HICLASS III possesses some of the basic characteristics of an object-oriented language: data abstraction and inheritance. HICLASS III supports the concept of reusable modules (rules, nodes, networks). The internal procedures for the representation and manipulation of data types are bundled together. An inheritance mechanism as well as a procedural attachment capability were added to HICLASS III, which rounds out its frame based representation repertoire.

An inheritance mechanism provides the flexibility of inheriting values between slots of different frames and/or instances. The use of an inheritance relationship eliminates the problem of data replication and helps to enforce data consistency. Inheritance also provides the capability to create a hierarchy or a semantic net by linking together frames and instances (via their slots.)

An explanation subsystem was added to HICLASS III. The way that the meta-structure and data retrieval mechanism was implemented facilitated the construction of the explanation subsystem. The explanation subsystem allows a user to interactively suspend the execution of a network, in order to request trace and debugging information. The information available includes the history of nodes and rules executed, changes made to slot or variable values and the data currently stored in the meta-structure.

To ease the rule base development process, a new user-friendly rule editor (a part of the Logic Capture Module) was added to HICLASS III. The new HICLASS rule editor is a syntax-driven menu-based editor for HICLASS rules. It is "syntax-driven", because at each stage of writing a rule, it offers choices which will lead to a syntactically correct rule. It is "menu-based", because choices are presented to the user on menus. Typing is therefore minimized, since rules are built up by choosing words from menus. It was designed as a teaching tool to guide the knowledge engineer in building a valid rule.

Many additional user-callable procedures have been added. HICLASS extended its open-ended system philosophy even further by providing user-defined functions and a built-in interface to the underlying UNIX operating system. A user-defined function is an executable program that is written by a user to perform a certain task within a HICLASS network. The function always returns a value when it is called from a HICLASS rule. The built-in interface to UNIX allows UNIX utilities to be used in feeding external information to the rules for decision making.
Auxiliary Support Modules

Auxiliary support modules are used to augment the output capability of HICLASS. They are used mainly (but not exclusively) by the HICLASS PWA network. Three auxiliary support modules are available:

- Rule Driven Graphics Generator
- Rule Driven File Generator
- Route Sheet Generator

The first module is the Rule Driven Graphics Generator (RDGG). It may be invoked by HICLASS applications to generate integrated text and graphics instructions. RDGG produces its output in a neutral (hardware-independent) display command format. This display command format is used primarily for indicating how the integrated text and graphics instructions are to be displayed on a color terminal.

The second module is the Rule Driven File Generator (RDFG). It may also be invoked by HICLASS applications to generate formatted text files — such as parts lists used for a printed wiring assembly — from the data available in the working memory.

The final auxiliary support module which is available at the present time is the Route Sheet Generator (RSG). This module generates data — including part numbers, assembly instructions, and corresponding bar-coded information — which appear on the route sheets that accompany the parts as they travel from one work center to another. The route sheet and the part remain together until shipment. It was the Route Sheet Generator that sparked the idea to create the Rule Driven File Generator, which is more generic.

All three of these modules take advantage of UNIX software tools, especially lex and yacc for parsing instructions within the HICLASS rules. Future auxiliary support modules which are natural extensions of these three are a Rule Driven Robotics Instruction Generator (RDRIG), a Rule Driven Numerical Control Instruction Generator (RDNCIG), and other manufacturing related instruction generators that fit into the category of data-driven knowledge based applications.

Summary of Major Milestones

<table>
<thead>
<tr>
<th>HICLASS</th>
<th>Year</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HICLASS 0</td>
<td>1981</td>
<td>FORTRAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>computer aided process planning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>automated route sheet generation</td>
</tr>
<tr>
<td>SLAM II</td>
<td>1983</td>
<td>FLEXPLAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;warm start&quot;</td>
</tr>
<tr>
<td>CAD</td>
<td></td>
<td>rule set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KS300</td>
</tr>
<tr>
<td>HEARAY II</td>
<td></td>
<td>ROSE</td>
</tr>
<tr>
<td>&quot;C&quot; Compiler</td>
<td></td>
<td>SRL</td>
</tr>
<tr>
<td>&quot;C&quot;</td>
<td>1985</td>
<td>CBOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rule reduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>graphics display speed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deploy onto the factory floor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>general purpose expert system shell</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;C&quot; interpreter</td>
</tr>
<tr>
<td>HICLASS II</td>
<td>1986</td>
<td>SMALLTALK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;C&quot; translate/compile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>portable platform</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;C&quot; on UNIX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BERT execution speed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>explanation sub-system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>syntax-driven menu-based rule editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>procedure attachment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rule application</td>
</tr>
</tbody>
</table>

261