Generation of Electronic Product Documentation

Gregory Laurence Smith

The product knowledge manager (PKM) is a multiphase knowledge base system developed to aid in the life-cycle management of electronic products at Boeing Aerospace and Electronics. Numerous pieces of documentation (including source control drawings, fabrication drawings, acceptance test procedures, qualification test procedures) are required for nearly every electronic product developed by Boeing. This documentation is highly formatted and must comply with strict Boeing drafting standards as well as customer and program requirements. Phase 1 of PKM has been in production use since September 1987. It provides an intelligent user interface and a documentation facility to generate and manage drawings, documents, and their templates. The system is written in KEE and Lisp. Approximately 60 KEE knowledge bases provide information on products, programs, authorization, drafting standards, and documentation structure. Over 600 Lisp functions operate on the knowledge bases to perform user-requested operations.

Background

In 1985, magnetics was one of the top problem areas within Boeing. It was determined that the majority of the problems originated from a

lack of communications between the organizations within the product arena and deficient product documentation. As a result, products were often built to certain specifications and tested to others. Additionally, documentation generation required in-depth knowledge of the Boeing drafting standards as well as documentation requirements levied by the program or the customer. Only a few specialized engineers and technicians had the breadth and depth of knowledge to correctly generate this documentation.

In an effort to resolve the problems identified in the magnetics area, Boeing attacked the problem in two ways. First, a team approach was implemented to address the communication issues. Second, funding was provided for analysis of the overall product development process; PKM funding was authorized based on this analysis.

An initial attempt to address problems in the magnetics area dealt with magnetics device core selection. This software was written in M.1 on IBM personal computers. Although the system fulfilled the customers' initial requirements, it failed to solve what management believed to be the real problem, that of device design.

A second attempt to address device design was initiated and outgrew the capabilities of M.1. The system was moved to S.1 on a 11/780 Vax. This version performed magnetics device design using active equations. It was converted to KEE to provide a mouse-driven graphic interface. However, as with the previous attempt, this solution failed to address management's revised view of the problem, that of the entire product life cycle and its associated documentation.

Finally, the third attempt (written in KEE on a Xerox 1186) began to address product life-cycle issues. This system was moved to the Apollo platform (still written in KEE) to use machines that were available to the user.

In retrospect, many of the miscommunications between developers, managers, and users stemmed from a lack of understanding about what could be done with knowledge base systems and what the real problem was. End users and managers did not know the strengths or capabilities of knowledge base systems; hence, they could not request specific capabilities (as shown by the length of time between production availability and deployment). Through incremental development methods, users and managers gained background in knowledge base systems, and the developers gained a deeper understanding of the application.

Although not deployed until October 1989, the system has been in production use since September 1987. Over this two-year period, the users worked with development personnel to tune the system, comment on its capabilities, and refine the user interface.

Problem Definition

After nearly two years of prototypes and interim solutions, the overall problem was redefined and is summarized as follows:

First, existing documentation generation methods were not responsive to the rigorous format and structure requirements. Each program and customer imposes constraints on the documentation and the data. The data must be consistent across the documentation. No facility ensures that these requirements are followed and incorporated into the documentation. The documentation must be formatted to Boeing drafting standards for book form drawings. No other tool (besides cut and paste) provides this capability. Configuration control and revision control are required.

Second, no defined flow or control of data exists. In the past, designs were sometimes created to a particular set of requirements and tested by another. No system provides product tracking to determine how it performed and whether it should be used again. No documentation of problem processes is available or maintained. No facility to compare new requirements to past designs exists to accommodate reuse. No single database exists to hold the product knowledge attributes and allow for intelligent queries.

Third, no single system is available to perform all the functions required by the user. An integrated environment using an intelligent front end is required. Numerous application software tools are needed for design support, analysis, and simulation. These tools run on different platforms, require unique data formats, and have different user interfaces. The users are engineers, technicians, and documentation personnel, not computer scientists; they might not understand how or when to execute the tools or be familiar with the hardware each tool runs on. No process or procedural sequence exists for running design-verification programs. Tools might not be used because the users are unaware of their existence.

Initial Capability

Version 1.00 was released for full production use in October 1989. This version represented a mature documentation generation, manipulation, and status facility.

PKM views documentation production in terms of the overall product life cycle. It provides the user an integrated interface to execute analysis and design tools and captures the product attributes in the appropriate knowledge base. As a side effect of running these applications, required documentation attributes are captured and documenta-

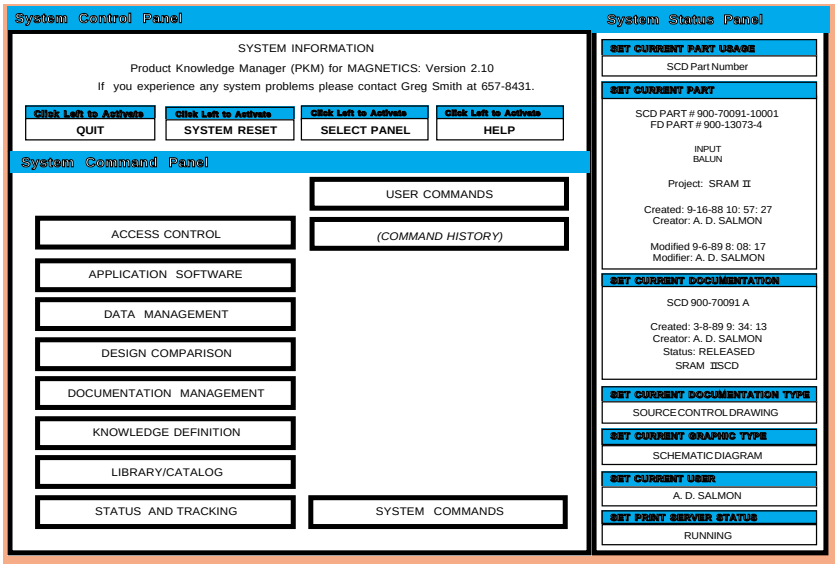


Figure 1. Product Knowledge Manager System Control Panel.

tion can semiautomatically be generated. Figure 1 shows the PKM user interface.

Documentation Generation

Central to the design of PKM is the concept of a part or product (hereafter referred to as a part), which is to be engineered under contract and must be documented. Design information about the part is captured as well as documentation requirements from the associated program. Documentation expressed as a book form drawing is generated by merging this information with Boeing drafting standards criteria. Figure 2 illustrates this documentation production process.

The process begins with the definition of a part whose attributes are defined, manipulated, and stored by the data management utilities. As previously stated, application software can be accessed to run design, analysis, and graphics programs. The organization of the documentation is then defined by creating and editing the documentation structure. Finally, during documentation production, all components are merged with criteria from the Boeing drafting standards to automatically format the text and generate a documentation definition file. The documentation definition file information is stored in a form similar to SGML (standard generalized markup language). User print requests convert this form to Postscript for output to appropriate printers. Be-

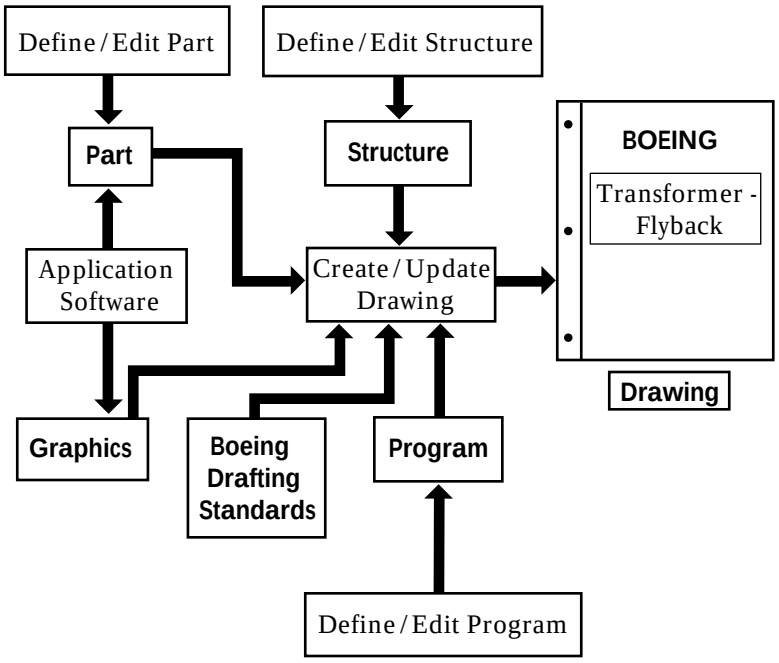


Figure 2. Product Knowledge Manager Documentation Production Process.

cause the documentation definition file is separately stored, changes to parts and structures will not migrate to previously created documentation (unless of course requested by user command).

Documentation Structure

Word processing and documentation generation systems abound within industry. What makes the documentation generation facility of PKM different is how the documentation structure is stored and manipulated. PKM separates the structure of documentation and stores it in an *ordered* knowledge base; that is, the order of text and graphic sections is fixed unless altered by the user.

Typical documentation consists of a title page, a revisions page, an active sheet record page, a table of contents page, text pages, and associated graphics pages. By linking a single parent to each of these components, a structure is formed. In this structure, the components become major sections.

Each major section in a structure has a Boeing drafting standards form associated with it. These forms contain formatting constraints dic-

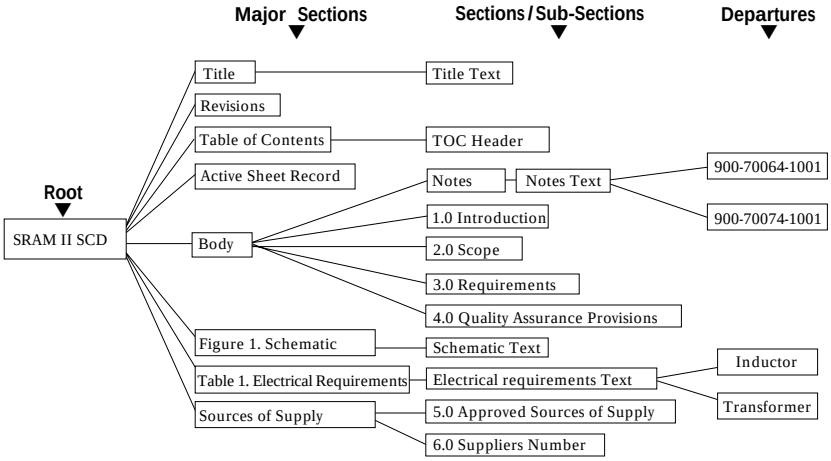


Figure 3. Documentation Structure Components.

tated by the Boeing drafting standards about where, what, and how to place text on them. A major section represents zero or more pages of text automatically placed on a specified Boeing drafting standards form. The number of pages generated by any major section is determined by the number of text lines contained in the section hierarchy and the number of text lines for each page allowed on the associated Boeing drafting standards form. Major section text is represented by a hierarchy of sections associated with the major section. Each section can contain zero or more paragraphs of text.

When producing documentation from a structure, situations arise where the specific boilerplate text supplied in a structure does not fit the requirements of the user. To handle those situations where alternative text is needed, *departure sections* are used (that is, the text can depart from the original text). Departure section text is used in lieu of section text whenever documentation for the specific part or part type is generated. Figure 3 shows the components of the documentation structure.

In addition to departure sections, a second facility is available to make boilerplate text specific for one type of documentation or part. *Part-attribute references* can be placed in section text. When documentation is generated, the attribute reference (for the current part being documented) is replaced with the value of the part attribute. This capability allows a single structure to generate unique documentation for every part.

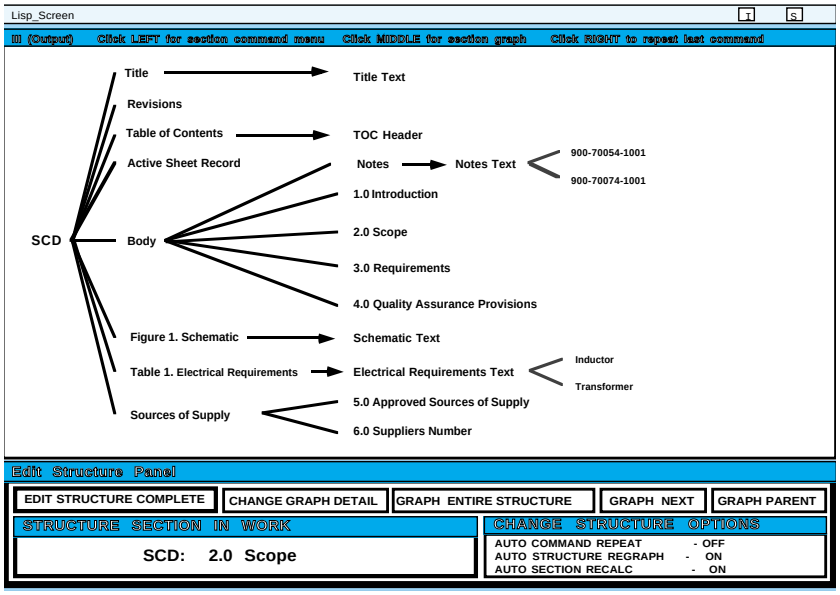


Figure 4. The Structure Editor Interface.

The Structure Editor

A complete structure editor is provided within PKM. The user interface consists of a graph of the structure and a command control panel. The panel allows the user to control the detail of the structure graph. Each structure object on the graph can be selected with the mouse, enabling a context-sensitive menu. Capability is provided to manipulate text sections, major sections, and departure sections as well as specific text within sections. Additionally, graphics from application software can be associated with major sections. Figure 4 shows the structure editor interface.

There are numerous advantages in storing the documentation structure in a knowledge base. The ability to visually display the layout of the structure allows the user to work on the documentation from a table of contents viewpoint. Because each section and subsection of text are represented as objects, and subsections are members of a section object, moving a section moves all subsections attached to it. Therefore, any amount of text can be moved anywhere in the structure simply by moving the parent object. Copying text from one section to another simply involves copying one object to another.

Industry standard documentation aids such as Interleaf, PageMaker, or Context are capable of providing most, if not all, of the capabilities in the structure editor. However, a great deal of time is spent describ-

ing to the system how the page should look (changing margins, setting fonts, positioning graphics). Typically, the end product is cut and pasted on forms, or a form is inserted in the printer before the documentation is printed.

However, the structure editor provides a controlled environment for the engineer to create special-purpose documentation. It controls where text is to be placed, what approved form the text should be printed on, what fonts should be used, how paragraph numbering should be handled as well as many other formatting and layout tasks that the user would much rather leave for someone else to do. Additionally, expert users of the industry standard documentation programs became expert users by virtue of continued use. In this engineering environment, users typically do not perform documentation activities for weeks. Whe it is time to create the documentation, users have no time to relearn command syntax of the documentation facility. Because the structure editor in PKM controls the fit, form, and function of the documentation, the user is left with the task he/she is most comfortable with, that of entering the text and creating the graphics.

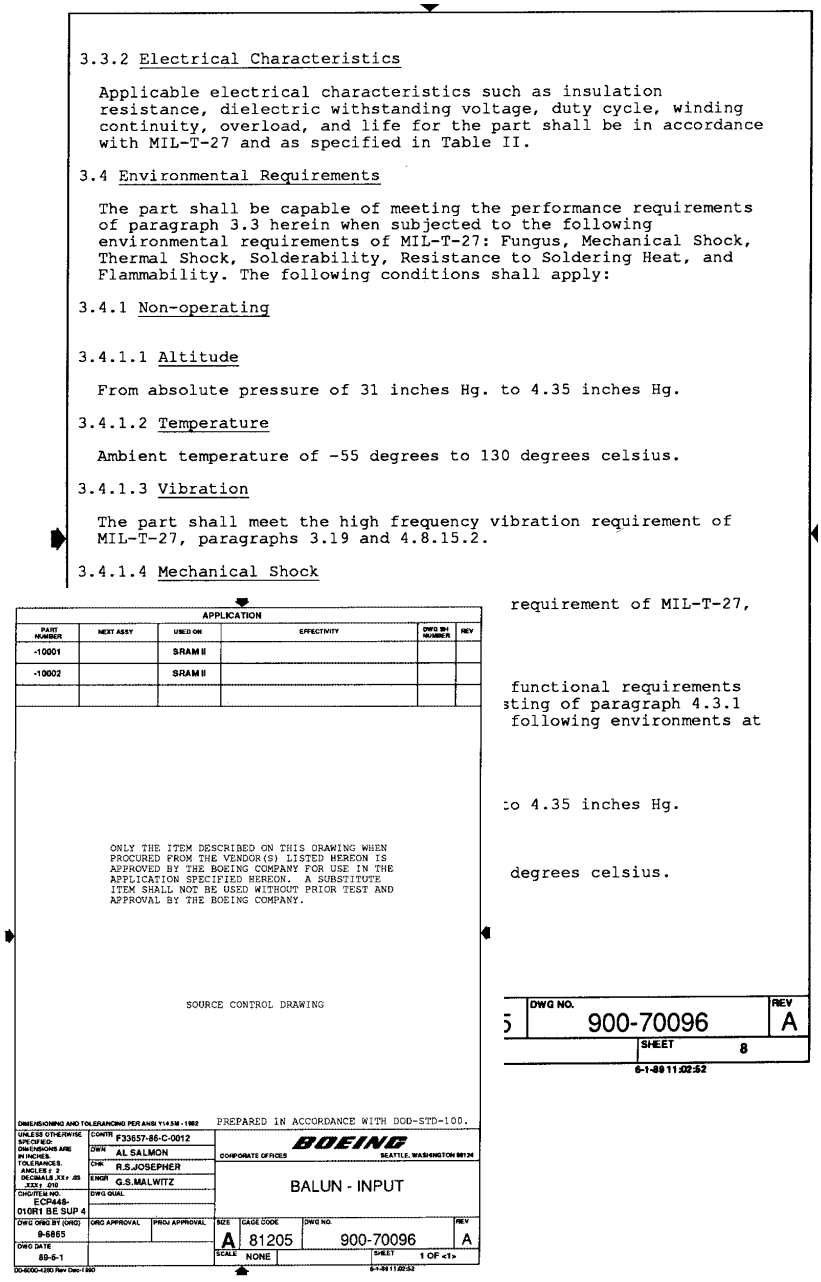
Product Knowledge Manager Knowledge Bases

PKM's documentation facility differs from other knowledge base system implementations; it uses information received from the user, as well as information contained in its knowledge bases, to control the generation of documentation. No rules are used.

In addition to the knowledge bases required for the documentation structures, numerous other knowledge bases exist within PKM. These knowledge bases store knowledge relating to command security, user authorization, application software interface requirements, part and documentation categorization, and system commands. By initiating system commands from the system control panel, users can store, collect, and retrieve knowledge from PKM knowledge bases.

Beyond Object Orientation

As previously stated, this system does not use rules. All functions are initiated by the user when a command is executed. The structure editor fully uses the object orientation of the structure knowledge base. Beyond the object orientation used by the structure editor (structure kb) and the user interface (system command kb), PKM exploits the use of inheritance and demons in various knowledge bases. Demons in the structure knowledge bases are fired whenever text in an object is al-



3.3.2 Electrical Characteristics

Applicable electrical characteristics such as insulation resistance, dielectric withstanding voltage, duty cycle, winding continuity, overload, and life for the part shall be in accordance with MIL-T-27 and as specified in Table II.

3.4 Environmental Requirements

The part shall be capable of meeting the performance requirements of paragraph 3.3 herein when subjected to the following environmental requirements of MIL-T-27: Fungus, Mechanical Shock, Thermal Shock, Solderability, Resistance to Soldering Heat, and Flammability. The following conditions shall apply:

3.4.1 Non-operating

3.4.1.1 Altitude

From absolute pressure of 31 inches Hg. to 4.35 inches Hg.

3.4.1.2 Temperature

Ambient temperature of -55 degrees to 130 degrees celsius.

3.4.1.3 Vibration

The part shall meet the high frequency vibration requirement of MIL-T-27, paragraphs 3.19 and 4.8.15.2.

3.4.1.4 Mechanical Shock

requirement of MIL-T-27,

functional requirements
sting of paragraph 4.3.1
following environments at

:o 4.35 inches Hg.

degrees celsius.

APPLICATION

PART NUMBER	NEXT ASSY	USED ON	EFFECTIVITY	DWG SH NUMBER	REV
-10001		SRAM II			
-10002		SRAM II			

ONLY THE ITEM DESCRIBED ON THIS DRAWING WHEN PROCURED FROM THE VENDOR(S) LISTED HEREON IS APPROVED BY THE BOEING COMPANY FOR USE IN THE APPLICATION SPECIFIED HEREON. A SUBSTITUTE ITEM SHALL NOT BE USED WITHOUT PRIOR TEST AND APPROVAL BY THE BOEING COMPANY.

SOURCE CONTROL DRAWING

DWG NO. 900-70096 REV A

SHEET 8

6-1-89 11:02:52

DIMENSIONING AND TOLERANCING PER ANSI Y14.5M - 1982

PREPARED IN ACCORDANCE WITH DOD-STD-100.

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES. TOLERANCES, ANGLES ± 2, DECIMALS .XX ± .05 .XXX ± .010

CONTR F33657-86-C-0012

DWN AL SALMON

CHK R.S.JOSEPHER

ENGR G.S.MALWITZ

DWG QUAL

CHG/ITEM NO. E-CP448-010R1 BE SUP 4

DWG ORIG BY (ORIG) 9-6865

ORIG APPROVAL

PROJ APPROVAL

DWG DATE 89-6-1

BOEING

CORPORATE OFFICES SEATTLE, WASHINGTON 98124

BALUN - INPUT

SIZE	CAGE CODE	DWG NO.	REV
A	81205	900-70096	A

SCALE NONE SHEET 1 OF <1>

D0-6000-A1B0 Rev Dec-1990

6-1-89 11:02:52

Figure 5. Typical Documentation Pages Generated by PKM

tered. Any text that is modified is immediately examined to determine if it contains references to part attributes, section numbers, or formatting codes; appropriate action is taken in each of these cases. Additionally, single and multiple inheritance is used in the part knowledge bases where program, form, documentation, and authorization knowledge can each provide values for part attributes.

System Maintenance

Surprisingly, PKM has required a minimum of maintenance. The majority of knowledge in each of the knowledge bases can directly be modified (depending on security requirements) by the user. Interaction with the system is controlled by the level of the user. The system maintains three levels of users: normal end users, system administrators, and system developers.

End users have the ability to manipulate knowledge in part, structure, and documentation knowledge bases (provided they have authority). *System administrators* have the added ability to manipulate forms, security, authorization, program, topology, and part-type knowledge bases; all password protection is invisible to them. *System developers* have control over the system command and user interface knowledge bases. They also have the only access to system function code and developer facilities (compiling, loading, and changing Lisp code).

As previously stated, the system consists of over 600 Lisp functions. It is estimated that 25 percent of the code is for documentation generation, 25 percent for structure manipulation, 25 percent for system and application interfaces, and 25 percent for the control of the user interface. The code for the operating system interface, the user interface, and the system maintenance could conceivably be reused with a minimum of modification.

System Savings

It is difficult to accurately estimate the savings attributable to this system. However, in the magnetics area alone (our target environment), over 200 devices are created and modified each year. Each of these devices requires over a half dozen different types of documentation. It has been estimated that using conventional word processing techniques, one document or drawing can take as many as 40 hours to generate. PKM has generated documentation in 3 hours on the average and, in many cases, has reduced this time to one-half hour. Additional savings attributable to the integrated user environment and improved

quality, completeness, and standardization of documentation is substantial but difficult to quantify.

Current Activities

Now that the initial system has been released and is in production use, marketing the system to other areas within Boeing has begun. We are currently coordinating with other part development groups and the Application-Specific Integrated Circuits Development Organization. Version 2.00 was released for production use in July 1990. This version incorporated numerous user-requested enhancements. The system is currently being used by two new programs. Each program is comparing PKM to their conventional documentation processes (Mac- and PC-based tools). The strengths and weaknesses of both systems will be identified, and the system that best fits the program requirements will be retained.

Future Directions

It is envisioned the PKM could become the user interface to the Boeing Electronics Part Database. All knowledge of Boeing parts would eventually reside in this database. Expert system gateways would provide access to, and allow for, intelligent queries of the database. PKM would facilitate the downloading of parts that fit the users' requirements. Documentation could be generated as required.

Development Costs

Since the fall of 1985, PKM development has required one-person level of effort. An additional one-person level of budget was provided to document the effort and critique system capabilities and the user interface. A third budget was provided to fund the magnetics expert's time (at a one-quarter-person level of effort).

Deployment Challenges

Another factor contributing to the long period between system availability and deployment stems from the use of KEE as a delivery platform. KEE on the Apollo provides an extremely strong development environment. However, IntelliCorp does not currently address a delivery system on this platform. To this end, numerous Lisp functions had to be generated to make KEE operations transparent to the users.

The current version of KEE requires the use of high-end Apollo workstations (model 4000 or 4500 with 32 megabytes of memory and large disk drives). Unfortunately, only a few of these computers are available within Boeing. Obviously, the number of available computers limits the number of PKM users. This requirement for high-end computer platforms will halt the expansion of PKM into other areas within Boeing until the problem is rectified. (IntelliCorp recently [August 1990] made available a beta copy of KEE 3.1 for the Apollo. This version uses the latest release of Lucid Lisp and includes a function reorganizer. This reorganizer facilitates the movement of frequently used Lisp functions to the front of a disk image and infrequently used functions to the rear. This version of KEE will reduce the memory requirements for KEE and, hence, allow it to run on lower-performance Apollos. It is expected that this version of KEE will allow PKM to efficiently run on Apollo 3000 class workstations, which are readily available within Boeing.)

Summary

PKM is currently running on Apollo workstations; it consists of over 60 KEE knowledge bases. Six hundred Lisp functions interact between the knowledge bases to perform user-requested commands. Phase 1 of PKM provides a mature documentation manipulation and generation facility as well as an integrated, intelligent user interface. Phase 2 of PKM will expand the current capabilities and provide additional interfaces to application software and part-attribute editors.

Acknowledgments

For their contributions, the author would like to thank Carl Mahnken, Dale Snell, George Malwitz, Andy Clark, Jim Hatley, Pete Baird, Dennis Hoorn, Al Salmon, Howard Smith, Pam Pincha-Wagener, and Donna Smith.