NASA Technical Memorandum 102156

Considerations for a Design and Operations Knowledge Support System for Space Station Freedom

Jon D. Erickson, Kenneth H. Crouse, Donald B. Wechsler, and Douglas R. Flaherty

October 1989

National Aeronautics and Space Administration

Lyndon B. Johnson Space Center
Houston, Texas

NASA Technical Memorandum 102156

Considerations For a Design and Operations Knowledge Support System for Space Station Freedom

Jon D. Erickson and Kenneth H. Crouse
Lyndon B. Johnson Space Center
Houston, Texas

Donald B. Wechsler
The MITRE Corporation
Houston, Texas

Douglas R. Flaherty
McDonnell Douglas Space Systems Company
Huntington Beach, California

CONTENTS

PAGE IV INTENTIONALLY BLANK

TABLES

FIGURES

ACRONYMS/ABBREVIATIONS

AI	Artificial Intelligence
AMRF	Advanced Manufacturing Research Facility
ANSI	American National Standards Institute
APT	Automatically Programmed Tools
ARC	Ames Research Center
ARP	Automation and Robotics Panel
ASCII	American Standard Code for Information Interchange
ASME	American Society of Mechanical Engineers
ATAC	Advanced Technology Advisory Committee
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CALS	Computer Aided Acquisition and Logistics Support
CAM	Computer Aided Manufacturing
CAM-I	Computer Aided Manufacturing - International
CASE	Computer Aided Software Engineering
CD-ROM	Compact Disk - Read Only Memory
CIL	Critical Item List
CIM	Computer Integrated Manufacturing
CL	Cutter Location
DAC	Design Augmented by Computer
DBMS	Data Base Management System
DDBMS	Distributed Data Base Management System
DMA	Digraph Matrix Analysis
DOD	Department of Defense
DOKSS	Design and Operations Knowledge Support System
DOS	Disk Operating System
EDIF	Electronics Data Interchange Format
EIA	Electronic Industries Association
EIM	Engineering Information Model
EIS	Engineering Information System
FEA	Finite - Element Analysis
FMEA	Failure Modes and Effects Analysis
GESST	Ground Expert System for the Space Telescope

IDS	Integrated Design Support
IGES	Initial Graphics Exchange Specification
ISO	International Standards Organization
JSC	Lyndon B. Johnson Space Center
KBE	Knowledge Base Environment
LAN	Local-Area Network
MB	Megabyte
MDSSC	McDonnell Douglas Space Systems Company
MIL-STD	Military Standard
MIT	Massachusetts Institute of Technology
M-O	Magneto-Optic
MSFC	Marshall Space Flight Center
MSIF	Multisystem Integration Facility
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
NSTS	National Space Transportation System
ORU	Orbital Replaceable Unit
PC	Personal Computer
PDCM	Product Data Control Model
PDES	Product Data Exchange Specification
PSC	Program Support Contractor
RAMCAD	Reliability and Maintainability in Computer Aided Design
RFP	Request for Proposal
ROM	Read-Only Memory
SASE	Specification Analysis, Synthesis, and Expression System
SCSI	Small Computer System Interface
SDP	Standard Data Processor
SDRC	Structural Dynamics Research Corporation
SES	Systems Engineering Simulator
SGML	Standard Generalized Mark-up Language
SSE	Software Support Environment
SSF	Space Station Freedom
SSIS	Space Station Information System
TMIS	Technical and Management Information System
VLSI	Very Large-Scale Integration

WORM	Write Once, Read Many
WP-2	Work Package-2

SUMMARY

Engineering and operations of modern engineered systems depend critically upon detailed design and operations knowledge that is accurate and authoritative. The purpose of a design and operations knowledge support system (DOKSS) is to provide convenient and effective access to this multifaceted information for design support and operations support. While such systems are relatively new, those that do exist have proven their cost-effectiveness, saving up to 20 percent of engineered system development cost, and have enabled improvements in quality and risk management.

A DOKSS is a modern computer-based information system providing knowledge about the creation, evolution, and operation of an engineered system. By using a distributed network of computer workstations to provide connectivity to numerous users and suppliers of design and operations knowledge, convenient access to accurate information is practical for multiple, geographically separated organizations. These information systems can be established by integrating commercial, off-the-shelf products with existing, in-use hardware and software. The ideal is that there should be a DOKSS so useful no one would think of proceeding without using it.

Conveniently accessible knowledge about the design of hardware and software, together with essential underlying rationale about "why" objects in a system were designed, built, and operated the way they were, is critical in accomplishing such major activities as systems design, engineering, and integration. The nature of these major activities for complex systems, which are usually accomplished by different personnel and organizations at different locations, depends on accurate, up-to-date information.

Design knowledge encompasses not only "what" the substance of a design is, but "how" it has evolved to its current configuration and "why" it satisfies functional requirements. This designer's knowledge is composed of object descriptions and assertions about these objects and their design. Designer's knowledge may include functional requirements, criteria or intent for selection of a design approach or solution, analyses results and conclusions, and assertions concerning expected object behavior.

A DOKSS provides access to many levels of information about the design of engineered systems and the process which the designers followed in order to reach the current design. Because a DOKSS contains so many different types of facts about both accepted and rejected designs, it makes it much easier to modify an engineered system. By making this information available through a distributed DOKSS, designers are able to explore new avenues without duplicating past failures. Two benefits of DOKSS's are that they shorten the design cycle by eliminating previously considered false trails and they help to accomplish higher quality products with less effort and decrease the overall cost by providing simple "what if" analyses which allow engineers to test the impact of their design decisions before they are put into effect. Finally, if a system should fail, failure analysis (which is very difficult without access to detailed design information) can be facilitated by tracking the design knowledge (design decisions and rationale) which was collected throughout the life cycle of the failed system.

An examination of existing technology shows that DOKSS's exist in industry which have proven cost beneficial and that the technology will support a DOKSS, including design knowledge capture, for Space Station Freedom (SSF). A DOKSS approach has been defined for the Lyndon B. Johnson Space

Center (JSC) and the Work Package-2 (WP-2) portion of the SSF program. The configuration of this DOKSS will make maximum use of existing hardware and software and will be compatible with SSF program elements such as the Technical and Management Information System (TMIS) and the Software Support Environment (SSE). Networking with DOKSS's of other SSF program elements can be accomplished through TMIS or by other data transfer mechanisms.

Currently, JSC and WP-2 McDonnell Douglas Space Systems Company (MDSSC) design knowledge support teams are capturing high-level design decision rationale. At this stage of the SSF program, the impact of this DOKSS activity on SSF engineers is minimal. The DOKSS will eventually contain knowledge about all the design related issues associated with the JSC/WP-2 component of SSF. In order to achieve this goal, copies of reports, problems, action items, and resolutions of problems which are pertinent to the "design" of all engineered elements and systems must be provided to the DOKSS personnel. Sources of these kinds of information include (but are not limited to) trade studies, design reviews, and various "Board" decisions.

As the SSF program progresses, the level of the design knowledge will become more and more detailed. During the 1990's, when the lower-level detailed design issues will be addressed, the impact of the DOKSS will become greater. In order to aid in the design knowledge capture process, the DOKSS will provide automated tools through designer's desktop computer workstations. It will be necessary for DOKSS personnel to make a coordinated effort with SSF designers in order to identify useful knowledge capture tools. The DOKSS will also provide tools to aid the design decision process. Simple simulation models which access the DOKSS will help to determine the impact of local design decisions on the overall SSF environment.

A DOKSS is advantageous to all designers, engineers, and managers of any complex engineered system like SSF. It requires a commitment to cooperation, but relatively little effort from engineering personnel. There needs to be a value structure within the organization and program that makes clear the importance of supplying "complete" design knowledge and that the consequences of even minor omissions can be serious. This value structure is required to enforce the discipline which is required to enter the knowledge because the knowledge source (who is most familiar with the knowledge and has the least use for it) tends to view entering the knowledge as unimportant. The immediate and long-term benefits are significant, particularly if it is necessary to redesign any part of the SSF's subsystems or elements. The operations benefits accrue over the longest period.

SECTION 1
INTRODUCTION

Conveniently accessible knowledge about the design of hardware and software and their interfaces, together with essential underlying rationale (MacLean et al., May 1989) as to why objects in a system were designed, built, and operated the way they were, is critical in accomplishing such major activities as system design, including design analysis and evaluation; systems engineering; manufacturing; systems integration, assembly, and checkout; improving quality; operations and operations support for maintenance and logistics, planning, training, control, fault diagnosis, repair, performance improvement, and robotics; sustaining engineering; and evolution.

The nature of these major activities for complex systems, which are usually accomplished by different personnel and organizations at different locations, depends critically on detailed knowledge that is accurate and authoritative. Certain aspects of advanced automation such as fault diagnosis by knowledge-based systems cannot be accomplished without access to design knowledge. The impact of having the knowledge easily accessible is that these essential activities can be more easily accomplished. These activities are difficult and time consuming without convenient access to such knowledge.

The retention and use of design data, and to some extent, design knowledge, is an integral part of large-scale engineered system development within NASA. However, it is largely a manual and paper-based process. For the SSF program, a TMIS will provide standards and connectivity, as well as some data bases. The concepts discussed herein build on TMIS, the SSE, and other program capabilities with an open systems approach.

The purpose of a DOKSS is to provide convenient and effective access to the required knowledge. A DOKSS is a modern computer-based information system providing knowledge about the design and operation of an engineered system for a purpose: quality through accuracy. By using a distributed network to provide connectivity to numerous users and suppliers of design knowledge, convenient access to accurate design knowledge is practical for multiple, geographically separated organizations. This convenient access increases effectiveness, efficiency, and productivity, supporting efforts focussed on improving quality and making these DOKSS's cost-effective as well. A unified system provides leverage through analysis tools and access to knowledge to greatly improve the effectiveness and productivity of personnel in these activity areas due to convenient access, reduced data handling, unified knowledge repository, unified configuration control, improved communication accuracy, and availability of key knowledge.

One definition of quality is the degree of compliance with user requirements. Another definition is that quality is meeting the customer's needs over the life cycle of the product at the best value to the customer (Bunn, 7 February 1989). This definition clearly implies that quality is the responsibility of everyone in the program, from earliest design to operational use. Knowledge of the requirements and engineered system certainly improves the ability to achieve compliance and to manage the risk of not achieving user requirements in terms of operational performance, life cycle costs, and schedule constraints. In fact, improved quality brings lower costs and improved schedules.

The notion of knowledge about the design and operation of an engineered system being practical and useful is not new, but the notion of putting it in a support system (figure 1) to be delivered in advance of, and operated with, the engineered system is a recent advance. While such systems are relatively new and NASA has not had much experience with them, the systems that do exist have proven their cost-effectiveness and enabled improvements in quality and risk management.

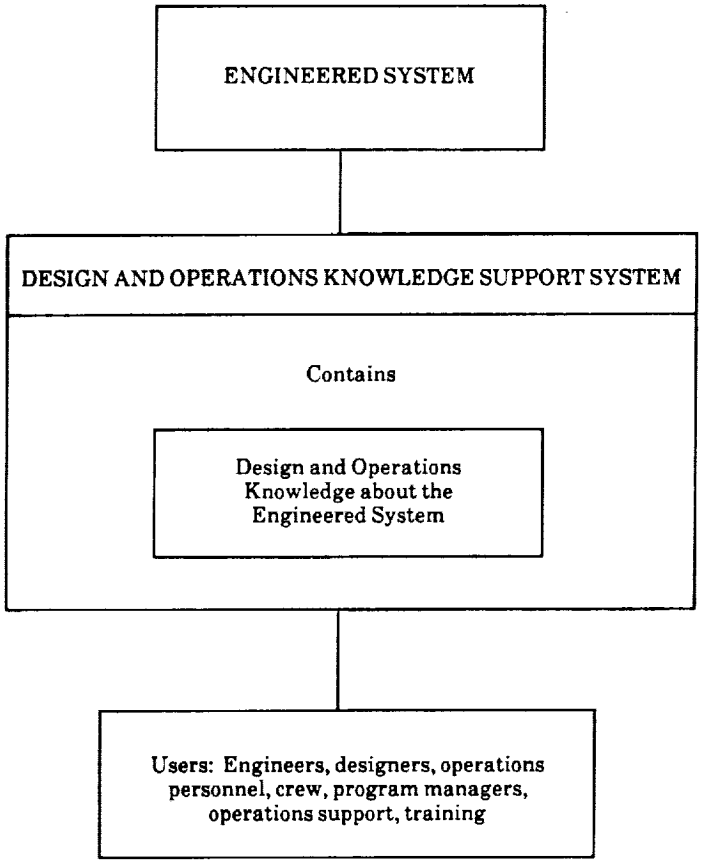

Figure 1.- Schematic showing relationship of an engineered system, its DOKSS, and the design through operations knowledge contained in the support system which is made available to the various users.

The complexity of SSF's systems, elements, interfaces, and organizations makes convenient access to design knowledge especially important, when compared to simpler systems. The life cycle length, being 30 or more years, adds a new dimension to space operations, maintenance, and evolution. SSF operations of assembly, verification, and maintenance occur on-orbit rather than on the ground, which also increases the need for easy access to design knowledge for real-time, safety critical operations.

For user convenience, it is also important to use compatible DOKSS's for the National Space Transportation System (NSTS) program as for the SSF program. In turn, these support systems need to be compatible with emerging industry and Department of Defense (DOD) standards for product descriptions and graphics exchange.

This technical memorandum provides a review and discussion of DOKSS's to be delivered and operated as a critical part of the engineered system. "Delivery" may be in-place with remote access via networks rather than total shipment to a single location. Continual updating is assumed. Examinations are made of the definition of design knowledge valuable enough to be a requirement, and DOKSS's which aid the creation and use of this design knowledge are addressed. The motivations for these DOKSS's are discussed, including different levels of automated support in a DOKSS and the benefits to advanced automation. A brief discussion is provided of some current industry and government DOKSS's which are cost-effective for conducting business, at least in large organizations or large projects involving multiple organizations.

A concept of a design and operations knowledge support system for SSF is presented. This is intended to clarify many of the features for the many users and suppliers of knowledge about SSF as the development of the DOKSS occurs. This is followed by a detailed discussion of such a system for the JSC and WP-2 portions of SSF. Four example scenarios for using a DOKSS for SSF are provided.

A major portion of this technical memorandum is a review and assessment of the technology underlying design and operations knowledge support systems. The review section supports the conclusion that the technology is adequate today to support DOKSS's for major aerospace systems. Can the technology be improved? Certainly. It is being improved every day. But these improvements should not keep anyone from proceeding with implementing such systems today, because cost-effectiveness and quality improvements to the engineered systems and their operations have already been achieved.

An important and powerful implementation concept for a DOKSS is the object-oriented paradigm for modeling. This concept supports a high degree of modularity of software for user interfaces and data bases of text and graphics and enables capabilities such as mouseable graphics and text linked to data bases of a very broad range of flexible representations. A discussion of this is provided in section 5.

In part, this technical memorandum grew out of the efforts of a subcommittee of NASA's Advanced Technology Advisory Committee (ATAC) to review the state of the art in design knowledge capture technology and systems in the fall of 1987. Mr. Donald Wechsler wrote the material for the subcommittee and it was subsequently reviewed and commented upon by ATAC. Mr. Wechsler updated this material for a MITRE working paper and this forms the basis for section 4. Other material was generated and presented to management at NASA Headquarters and JSC and forms the basis of sections 2, 3, and 5. User scenarios were developed by McDonnell Douglas and adapted for this report. Jon Gilbert, Dick Baker, and Donald Woods of McDonnell Douglas reviewed the preliminary draft and provided significant additional inputs. Dr. R. Kent Lennington of Lockheed Missiles and Space Company supplied the section on the SSE. Mr. James Dragg, Lockheed Engineering and Sciences Company contractor, provided technical writing support.

Credit is given the Space Station Automation Study participants, the Cal Space led Automation and Robotics Panel (ARP), and in particular, SRI International, with bringing attention to the requirement for and usefulness of design knowledge in 1984.

SECTION 2
DESIGN KNOWLEDGE

MOTIVATION FOR USE

A DOKSS is useful to anyone who has a need for knowledge about the design and operation of a system, whether it is an automobile, industrial plant, aircraft, or spacecraft. The technical personnel and managers come from engineering, operations, operations support, and project offices performing tasks such as design, design analysis and evaluation, systems engineering, manufacturing, assembly and integration, sustaining engineering, operations procedure development, and training.

Examples of DOKSS's in United States industry show they result in cost savings over the life cycle of systems and result in more effective activities such as systems engineering. General Motors, for example, has found that the biggest impact area for their DOKSS has been more effective and efficient systems engineering and integration for each model car they manufacture. They have also saved millions of dollars by employing such systems. Major architectural and engineering firms have also found, that by adopting such support systems, they can deliver projects under budget and in less time than normally scheduled. The support systems help to shorten the design cycle and help to accomplish higher quality work in the time allotted.

The complexity of SSF's systems, elements, interfaces, and organizations makes convenient access to design knowledge especially important. The life cycle length of 30 or more years on-orbit adds a new dimension to space operations, maintenance, and evolution. SSF operations of assembly, verification, and maintenance occur on-orbit rather than on the ground, which also increases the need for easy access to design knowledge for safety critical operations.

Of course, systems, even complex ones, can be built and operated without machine-intelligible design knowledge capture and use (just as one might operate a bank with a manual bookkeeping and paper handling system without a unified computer system), but it is more costly and slower to do so.

There are significantly different levels of automated support in DOKSS's. The paper document system of obtaining design knowledge, which has historically been used, has major deficiencies in comparison to a system in which the knowledge is retained in machine usable form. A major problem with paper document systems is an inability to keep multiple sets of documents current with constantly occurring changes. Inaccuracies begin to enter through not keeping current and the users are many, which multiplies the cost of keeping current. Also, when a use of design knowledge is attempted, the form is generally inappropriate. It is inefficient and error prone, at best, to search through the documents to find what is needed and then put it into the form needed for use, such as in an analysis program. The suppliers of the knowledge in paper document systems more often must "come with" the knowledge. That is, more of their time is used to support the reader of the documents with in-person interpretations. Many times, these and other deficiencies have caused the designer's knowledge to be retained in volatile form in the heads of humans, with operations personnel left too often to their own devices, such as looking at selected operating variables and parameters over time as a means to cope with lack of design knowledge.

The first level of automated support is provided when a machine form of the documents is available.

A second level of automated support is provided when the machine-interpretable form of the knowledge is automatically translated into the input form of various supplied value-added software applications such as computer aided design (CAD), computer aided engineering (CAE), computer aided manufacturing (CAM) systems, and data base management systems. Systems which provide a user access to any or all of the data in any or all of the distributed systems are called integrated. Knowledge representation for automation applications in the engineered system may require a different form of representation as discussed next.

A third level of automated support is provided when "knowledge engineering" approaches, utilizing artificial intelligence (AI) technology, are applied in an attempt to capture what was done after most of the design is completed. The primary knowledge representation in these knowledge-based systems or expert systems may be in specialized forms such as rule-based knowledge bases which represent the expertise of the design engineer about a particular engineered system in terms of heuristics or "if-then" rules. To achieve the knowledge acquisition and representation vocabulary and tools necessary to support this level of automation fully, several years of development will be required. Many knowledge acquisition tools are available and can be used to advantage.

A fourth level of automated support is provided when design expert systems substantially take on the function of the engineer and designer and increase their productivity substantially. Although design expert systems have been developed and are operating with tremendous advantage for those who are using them, they are not yet to the point of covering the broad range of design problems in a generic way in large, complex systems.

The second level of automation has been selected here as appropriate and sufficient for a DOKSS for SSF because the essential functionality is provided (see section 3) and can be delivered by integrating commercially available, off-the-shelf (e.g., COTS) products. The third and fourth levels can be added incrementally, but are insufficient at present without the second level.

There are viewpoints on design knowledge use stemming from automation and from robotics which are discussed in greater detail later in this section. In automation, the systems engineering and integration across interfaces for automated functions of monitoring and control, planning and scheduling, resource management, fault diagnosis, and others requires detailed knowledge of the hardware and software in distributed systems and the functional and timing details necessary to coordinate across systems. In addition, knowledge-based systems or expert systems for fault diagnosis depend on the knowledge of how a system component was designed to function and why it was designed that way in order to connect symptoms of anomalous behavior to hypothesized faults. In robotics, knowledge of the detailed geometry of the object or system being attended is necessary to accomplish assembly, repair, and servicing tasks. The broader uses of design knowledge provide the major justification, but the automation and robotics benefits of improved safety, reliability, and productivity cannot be achieved, generally, without readily accessible design knowledge. This knowledge can be obtained directly from the designer or through a DOKSS.

GENERAL REQUIREMENTS

Accurate and authoritative design knowledge is required to support efforts to reduce and control risks and achieve high quality in engineering, manufacturing, and operations. The largest single factor which determines the cost of design knowledge capture is the bounds which are established on the knowledge to be captured. Not all design or operations knowledge needs to be captured in machine-interpretable form and stored for future retrieval and use. Life cycle design and operations knowledge about an engineered system has to be a valid requirement before it is valuable enough to be put into machine-interpretable form, i.e., a specific use has been identified which defines in detail the nature of the knowledge needed. However, if the engineered system is large, complex, and distributed with operational requirements to support real-time, long-term nonstop, and mission and safety critical functions, then more of the knowledge is valuable compared to smaller and simpler applications.

The definition of the requirements for what and when to capture design knowledge in machine-intelligible form is derived from an analysis of the needs in the selected major program activities. The personnel who carry out various major program activities are the ones to be consulted in defining and recommending to management the uses that an integrated design knowledge system must support. The software models, analysis, and access tools that the DOKSS contains will, in turn, define the knowledge to capture and some bounds on the knowledge (data) requirements of these software tools. An example of a software analysis tool is a program for thermal and stress analyses of a space structure.

The need for accurate, authoritative, design, and operations knowledge (figure 2) starts with engineered system user requirements, engineering requirements, and the functional flow of these requirements from top level to subbreakdowns in the system hierarchy in such a way that visibility into authoritative requirements and traceability is easily computed. Then, changes in requirements can be traced to changes in design and operation in order that their impact can be estimated in benefit and cost terms and allow evaluation of alternative approaches to iterations of requirements and design.

The need continues through conceptual and preliminary design to final design, with the focus shifting to the design objects and their attributes in terms of their structure, function, and behavior. Function refers to selected purposes or roles performed by the object, while behavior refers to the manner of carrying out actions and responses to environmental stimuli. Thus, descriptions of these aspects (objects and attributes) of design are valuable. Designer's knowledge is also valuable for systems engineering and operations purposes. Synthesized knowledge in the form of models is needed. Such things as intentions and assumptions are added requirements to give the relevant domain "knowledge resources" of the designer. "Reasoning paths" of the designer are also a part of designer's knowledge including such things as analyses, explanations, and rationale of why the object is designed this way.

The need continues through development, testing, and evaluation. Knowledge of as-built aspects and testing results as a function of degree of integration along with the evaluations are useful. One of the largest benefits of access to accurate design knowledge is in the integration of engineered systems. The need continues through operations and maintenance phases with performance improvement, growth, and evolution supported by detailed knowledge of the current design and operation.

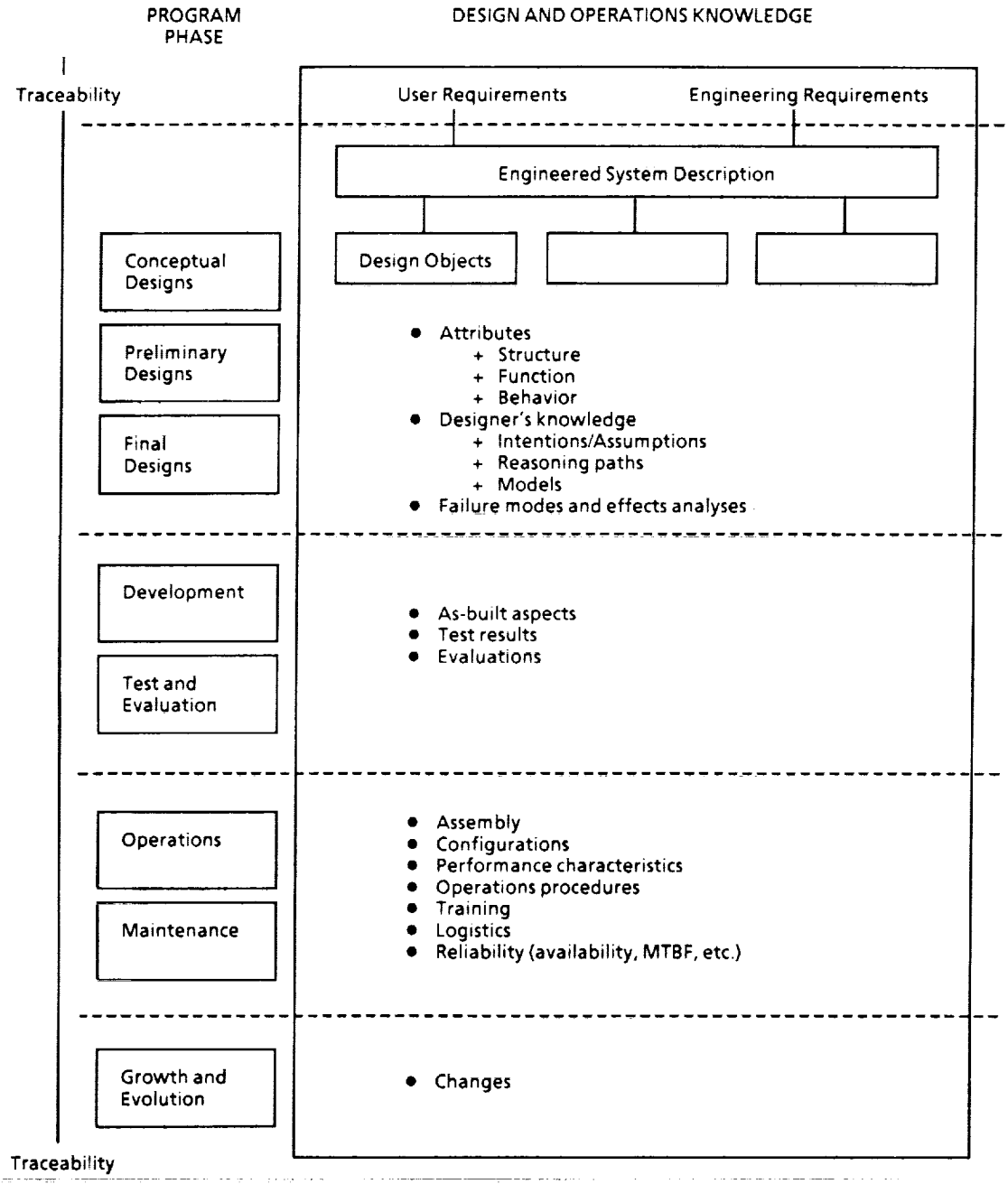

Figure 2.- Functional flow of design knowledge requirements.

The discussion herein has been following one dimension, "perspective," of three parameters of the definition of design knowledge for engineered systems. Namely, the answer to "What design knowledge is of interest?" comes from the identified uses of the knowledge. Where these uses are different, then the "perspectives" of knowledge about the design object are different. The selected uses define the detailed limit or bound on what knowledge should be put into machine-interpretable form.

But "how much" detail is needed This determines the volume of knowledge in another dimension. A certain "visibility" is required and appropriate, beyond which the value diminishes. The value in this visibility dimension can be estimated by the product of the probability of failure with the impact of failure in terms of damage, lost product, and cost of restoration. For example, if a design object has a high reliability and negligible results from failure, then the object will have a low visibility rating. An assembly of components would have a greater visibility than one subcomponent. Thus, a "visibility threshold" can be set on the level of detail using this measure of value.

The answer to the "How often" question in defining design knowledge valuable enough to require in machine form is given by: any "version" significantly different enough to be described as having quantitatively or qualitatively significant differences is to be treated as another version. An accurate accounting of changes and their rationale is required for each version. Tens of different versions are frequently used to achieve a "baseline" version.

For consistency reasons, the selection decisions on uses, level of detail, and versions should be made by management at the top-most, system-wide level, not separately, in uncoordinated ways, at lower levels of responsibility where only portions of the system are addressed.

Why design "knowledge?" Why not design information or design data? Design data to many people are just the uninterpreted or unreduced raw values for various attributes of the design objects. They are necessary to have, but are not sufficient. Design information contains distilled descriptions of the arrangements of design objects and their related structure, function, and behavior. Again, they are necessary to have, but are still not sufficient. But when we add the knowledge of the designer about the rationale of the design, the intentions and assumptions from the expertise of the designer, and synthesized models derived from the design, this truly becomes design knowledge. Design knowledge includes the necessary design data and design information, but adds an important ingredient of synthesis, designer's knowledge, to what constitutes the essence of a design.

Design knowledge encompasses not only what the substance of a design is, but how and why it satisfies functional requirements. Design knowledge is composed of design objects, their descriptions, and declarations or assertions called "designer's knowledge." Designer's knowledge may include functional requirements, criteria or intent for selection of a design approach or solution, declarations of analysis results and conclusions, and assertions concerning expected design object behavior.

Therefore, the general definition of life cycle design and operations knowledge valuable enough to require in machine-interpretable form can be summarized as follows:

a. Bounded descriptions of requirements and design objects, both hardware and software

b. The arrangement and attributes (structure, function, and behavior) of the design objects

c. The designer's knowledge consisting of knowledge resources (intentions and assumptions), synthesized knowledge (models), and reasoning paths (explanations, analyses, and rationale) of the designer and others as to why an engineered system was designed, built, and integrated the way it was

The bounded descriptions mentioned above provide partial limits on the content of what knowledge (identified uses or perspectives) is needed, how much detail to provide (risk-control-based visibility thresholded on value) and how often knowledge is needed (sufficiently different versions). Reasoning paths include analyses and rationale on design options not pursued to conclusion, including rationale for termination.

A more detailed definition of design (through operations) knowledge is given by describing in greater detail the necessary aspects of arrangement, structure, function, behavior, intentions, assumptions, synthesized models, explanations, analyses, and rationale. Examples of such detail for structure include dimensions (units), materials, and weight.

To the above design knowledge must be added: operations knowledge on assembly, configurations in use, performance measurements as a way of approaching improvement, development and support, logistics records, reliability parameters, and other historical data useful for improvement of operations.

SUPPORT FOR ADVANCED AUTOMATION

In addition to the general requirements, specific aspects and needs for design knowledge arise which are related to advanced automation.

Failure management is an important function of operations. Failure management maximizes, during operations, the end-to-end productivity and functionality of an engineered system in the presence of indications of failed portions of the system, by keeping or returning the productivity and functionality to the desired level as promptly as possible. Restoration of the desired level of redundancy is also included.

Fault diagnosis is a major portion of failure management. One of the more mature and cost-effective applications of artificial intelligence technology is fault diagnosis of an engineered system to substantially improve the system reliability and thus productivity of operation. This application involves several requirements, one of which is design knowledge useful in establishing cause and effect reasoning. The level in the system hierarchy to which isolation of the fault is traced must be specified. In spacecraft, this might be to the orbital replaceable unit (ORU) level or to the "board" or even to the "component" level. Further, the requirements must be spelled out for sensor instrumentation which provides information on the degree of performance degradation, if any, and aids in determining the location of any fault. An ideally sensor-instrumented system would only require table look-up software for diagnosis because of the one-to-one mapping of the sensors to failed parts. But this is never practical from cost, volume, and weight considerations, and reasoning by software (or humans or both) is required to make the diagnosis when there is less than the ideal complement of sensors.

Design knowledge is also required which relates to failure modes and effects analysis - critical item list (FMEA-CIL) efforts and to diagnostic (or malfunction) procedures and tests. The FMEA consists of an analysis of the symptoms (effects) of various faults and the logic of the expert designer or engineer in connecting these symptoms (effects) to system specific hardware or software failures (causes) which may involve diagnostic procedures and tests. A "rule-based" diagnostic expert system can be

constructed using this expert knowledge either to provide advice to the operations crew, or, if embedded in the engineered system, to more autonomously reconfigure, recover, or otherwise participate in the failure management process. If the knowledge of this mapping of symptoms to faults is not available, then this expert system cannot be constructed. An advantage of expert systems is that they contain an explanation facility to provide the rationale for the reasoning which led to a specific piece of advice or action. However, "rule-based" expert systems have no deeper explanation or reasoning process than is contained in the knowledge base of rules and facts (Clancy, Summer 1989).

"Model-based" diagnostic expert systems do not have this limitation. They are much more efficient and robust than a rule-based approach. In this case, the model contains the design knowledge expertise in a synthesized form which can be computer executed. Through model simulations and interpretations, a deeper knowledge and explanation is available. A comparison of the two approaches is provided in figure 3.

	RULE-BASED EXPERT SYSTEMS	MODEL-BASED REASONING
Knowledge representation	Rules	Digraphs Simulations
Knowledge acquisition	Domain expert interviews	Schematics Requirements simulation
Domain coverage	Dependent on domain expert's knowledge, mood, memory, stage of development	Dependent on accuracy of model and related simulations
Underlying reasoning	Logical inference using rules	Simulations of the actual device (i.e., its physics)
Speed	Dependent on amount of rules [inference engine (Rete, etc.)]	Dependent on simulation, qualitative or quantitative
Maturity	Well understood and widely implemented	Emerging

Figure 3.- A comparison of rule-based expert systems and model-based reasoning approaches for fault diagnosis.

MODELS AS A SYNTHESIZED FORM OF DESIGNER'S KNOWLEDGE

Design knowledge must be represented, stored, retrieved, and communicated, therefore the most condensed and efficient forms which still retain the essential knowledge should be used. Models, which provide a synthesized view, are a particularly efficient and useful form of designer's knowledge.

Models come in various types with the same general purpose, to aid in understanding some aspect of the real thing. There are physical models and mockups, scale models, and analytical models which may be either qualitative or quantitative or both. The type referred to here are the analytical models which are computer-based and represent some design knowledge about the engineered system of interest.

One use of models is to support the design process and provide feedback into modifications of the design, before it is baselined, by allowing the design engineer to see certain design details in the context of the entire object. CAD/CAE modeling capabilities support this use. So do high fidelity real-time simulation models.

An example of simulation models used in design analysis is provided by finite-element analysis (FEA). FEA is a very powerful design verification technology for analyzing mechanical structures and parts. Developed by NASA in the late 1950's, FEA software is used to import a CAD file created elsewhere or to build a detailed geometric model of a part or structure on a computer. The model is then divided into a finite number of simple building blocks called finite elements that are connected to each other at points called nodes. Mathematical equations describe how these nodes respond to simulated loads such as gravity, pressure, and heat applied to the elements in terms of deflections, stress and temperature distributions, natural frequencies, and other physical characteristics. The equations can take into account material composition and other variables. In this way, an engineer can spot flaws in a design before a costly prototype is built, thereby accelerating the design cycle and improving product quality and cost.

To determine the overall response of an entire structure, the equations for the individual elements are assembled in matrix format to provide a global description. This results in a large set of simultaneous equations to be solved by a computer.

The next step in the FEA process entails retrieving and examining the solution, either in graphical or numerical report form. This data reveal how the candidate design model performed. For example, one can identify stress concentrations where a part may fail or the output may verify that operating temperatures remain within specified limits. Analysis can also be used to compare deformation and stress results of different designs to determine which design is best.

The key concept in FEA is that any part can be analyzed since elements can be created for any geometry, no matter how complex or unique its shape. FEA is also an example of user supplied applications (analysis) software to which a DOKSS must provide input data and from which a DOKSS must obtain output for later access.

Another set of models of the type useful in design are those developed to support reliability analysis such as probabilistic risk assessment. Here, models of the digraph matrix analysis (DMA) type are built with models of connectivity and reachability, but these same models are a representation of design knowledge in a synthesized form.

Qualitative models and software tools to build them can be quite useful in the early stages of design by allowing the qualitative behavior of a system to be modeled and understood before major investments are made in any design. For example, consider a graphical modeling interface to a library of components (represented in both visible icon form and a hidden qualitative discrete event behavior form) from which a designer can assemble a schematic model of his subsystem, and in addition, be automatically provided (by the software modeling tool) qualitative discrete event descriptions of the behavior of the subsystem when various component faults are introduced. This capability can enable a designer to investigate the failure modes and effects at a very early stage and compare design alternatives on this

basis as well as other elements of behavior. The final design is represented as the final model in a form of design knowledge capture which is highly synthesized and very efficient in representing this design.

Another use of these models is to support operations directly. Since these models are executable, faults hypothesized to have just occurred can be introduced into the model to explore several analyses. One analysis is to verify that the symptoms observed could only be from this fault. Another analysis is to explore the propagation of this fault into others over time. Yet another may be to explore the implications of real-time procedure development before use.

DESIGN KNOWLEDGE CAPTURE WITHIN NASA

The level of design automation technology within NASA has reached a technological threshold for design knowledge capture implementation. Pockets of technical interest have emerged. The SSF program support contractor (PSC) is developing a capture implementation plan.

Integrated CAD/CAE

A system foundation in design automation can be an important step toward design knowledge capture. NASA Headquarters has solicited advice on computer aided design from the National Academy of Engineering - National Research Council (National Research Council, 1984). Some elements of NASA's JSC and Langley Research Center use an integrated CAD/CAE system named IDEAS-Squared. This system combines Structural Dynamics Research Corporation's (SDRC's) commercial GEOMOD solid geometry modeling system and accompanying analysis programs (SDRC's IDEAS product) with specialized NASA-developed analysis modules (a capability named IDEAS). The capabilities of commercial IDEAS include finite-element modeling, mechanisms, and structural, thermal, and dynamic analysis.

The modules of NASA's IDEAS perform the following analysis (Baker et al., May 1986):

a. Orbital lifetime

b. Spacecraft on-orbit forces and torques

c. Spacecraft low-orbit dynamic simulation and control system response

d. Forces resulting from plume impingement

e. Subsystem cost and capabilities analysis

f. Technology analysis for life support systems

The key development of IDEAS-Squared was the addition of an integration framework for the CAD and CAE facilities. The relational data base of IDEAS-Squared provides for CAD/CAE integration by storing the data from the geometric modeler for access by the analysis programs.

NASA has employed IDEAS-Squared in the SSF program for numerous purposes, including determination of SSF's torque equilibrium attitude and maximization of the microgravity volume for SSF's laboratory facility.

SSF Program Requirements

In January 1987, the SSF program established baseline process requirements for design knowledge capture. This document (SSP-30471) was updated in September 1988 (NASA Space Station Program Office, September 1988). The SSF Program Office coordinated design knowledge capture input for portions of SSF Phase C/D requests for proposal (RFP's) common to all participating NASA centers. The approach presented in SSP-30471 is sensitive to immediate SSF cost and schedule requirements. Initial approaches utilize adaptations of field-proven, commercially available technology.

Ground Expert System for the Space Telescope

Marshall Space Flight Center (MSFC) has completed a preliminary cost/benefits assessment for a Ground Expert System for the Space Telescope (GESST), resulting in an approved plan for knowledge acquisition, expert system building, and verification through simulations. MSFC will conduct the described functions for the space telescope. Ames Research Center (ARC) will also participate by addressing the core technology aspects of the MSFC test case, such as:

a. Building large-scale knowledge bases

b. Integrating design knowledge capture within a large-scale, multicenter NASA environment

c. Use of FMEA and other traditional engineering activities to supplement knowledge base development

SECTION 3
DESIGN AND OPERATIONS KNOWLEDGE SUPPORT SYSTEMS

The previous section defined and discussed the knowledge of design through operations for an engineered system, such as a spacecraft, where this knowledge is valuable enough to have accessible in machine-interpretable form. Engineering and operations of a modern engineered system depend critically on detailed design and operations knowledge that is accurate and authoritative. The purpose of a DKOSS is to meet this need as a means of achieving quality in a cost-effective way.

This section defines and discusses a DOKSS, which is required to develop and operate a modern engineered system, particularly a mission and safety critical system. A DOKSS consists of a cost-effective computer-based information system providing services and tools for manipulating (without changing) the design knowledge accessible via the support system. The DOKSS may be geographically and organizationally distributed, but connected through networks. A major objective of such a system is to serve in place of a paper document system, not in addition to a paper document system.

DEFINITION

In general, a support system is defined as "a composite of equipment, skills, and techniques capable of supporting an operational role. It includes related facilities, equipment, material, services, software, technical data, and personnel required for its operation and support to the degree that it can be considered a self-sufficient unit in its intended support environment." A design and operations knowledge support system supports engineering and operations throughout the life cycle of the engineered system with benefits to manufacturing at that stage.

From a user point of view, the functional characteristics of a DOKSS are as follows:

a. It provides convenient access to knowledge on the entire engineered system, hardware and software.

b. This knowledge is life cycle design and operations knowledge used for design support and operations support in the broadest sense.

c. This knowledge is represented in a collection of logically integrated forms of data, text, and graphics distributed over geographically separated computers, perhaps on networks. The one logical view of the knowledge comes from a single naming convention and an easy to use and "no training required" user interface with mouseable graphics and text supported by a host of services such as data integrity maintenance, location transparency, local concurrency or autonomy, and data, text, and graphics exchange standards. The "no training required" user interface hides the DOKSS implementation approach and details from the user, thus special procedures or understanding are not required for the user to accomplish what he or she wants to do.

d. It has a "bring-your-own" user applications software capability with access to the knowledge in the DOKSS.

e. It has an open system architecture enabling use of existing in-place hardware and software and integrates these elements with commercial, off-the-shelf (e.g., COTS) products. Growth is also enabled through this approach as the knowledge which the system contains evolves.

f. Accurate and authoritative knowledge is provided by allowing only one organizational element as originator/updater for each design object and its attributes and interfaces.

From an implementation point of view, the necessary elements of a DOKSS (figure 4) are as follows:

a. Naming conventions for the engineered system

b. User interfaces and (largely) user supplied applications software (CAD, models, analysis, simulation)

c. Access to all relevant design knowledge sources (possibly by connectivity to wide-area and local-area networks)

d. A distributed data base

e. Data base management systems (better if standardized, best if distributed)

f. A data model for structuring the data bases, efficient transactions, and enabling automatic data integrity and which relates to the engineered system hierarchy and work breakdown structure

g. Automatic data integrity services across the network

h. A data dictionary and directory for the entire engineered system

i. Standards for data exchange (IGES, EDIF, PDES, etc.)

j. Specification languages

k. Translation services

l. Applications input data builders

m. Multilevel security for read and write access

n. Write authority assigned to single knowledge sources; read-only authority for others

o. A policy on who has responsibility for entry/storage of the knowledge, and for how long, and on what media

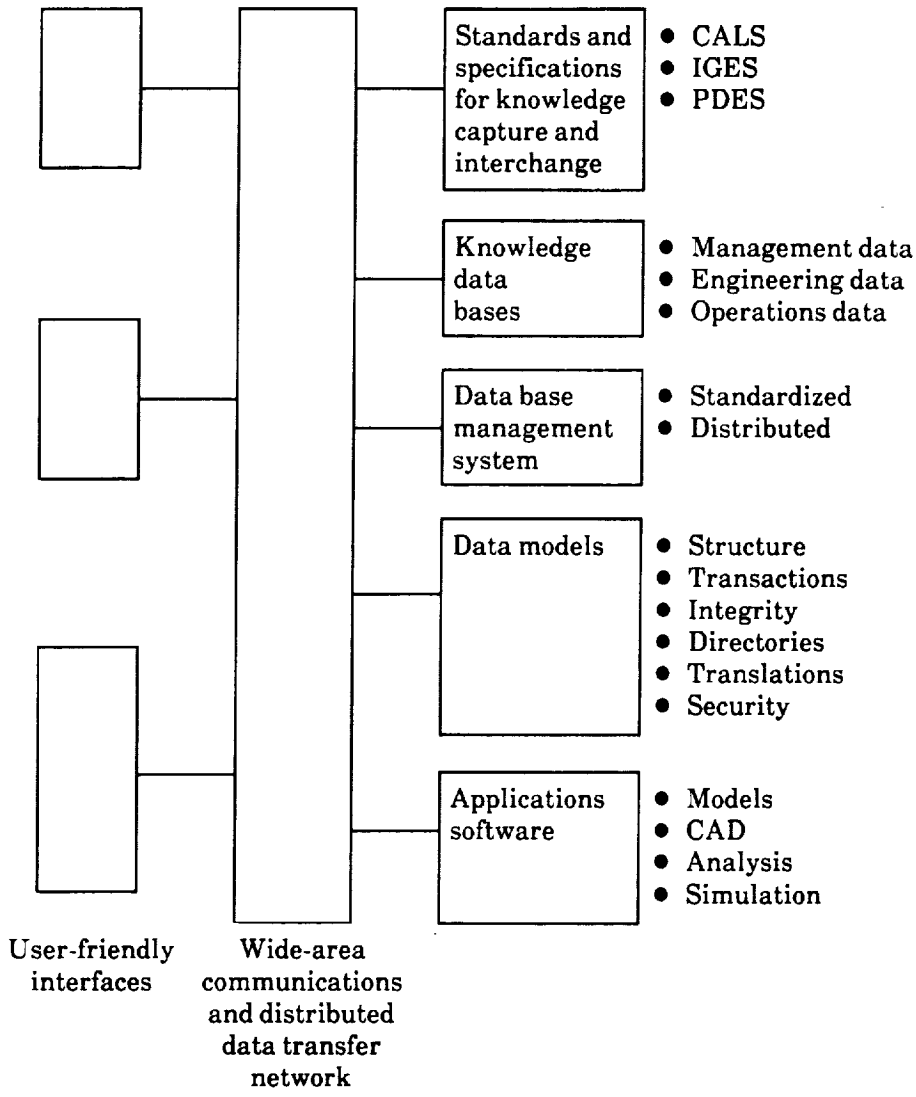

Figure 4.- DOKSS functional design.

The four technologies for sharing data between remote computers which are of interest for use in a DOKSS are

a. Transfer of data via optical media, magnetic disks, or tape

b. Use of file transfer and message routing techniques across the network

c. Use of a centralized data base server

d. Use of a distributed data base system

The distributed data base system potentially offers greater capability. However, this new technology is not well established.

A distributed data base is a collection of logically integrated data distributed over different sites of a computer network. A distributed data base management system (DDBMS) provides the capability to interact with data across a network of computers with the same functionality and ease that is commonplace today for interacting with data on a single computer.

The functionality desired from a distributed data base management system is the following:

a. Data integrity and error recovery

b. Distributed query processing

c. Distributed transaction management

d. Location transparency

e. Replication transparency

f. Fragmentation transparency

g. Global schema transparency

h. Ability to define shared versus nonshared data

i. Usage monitoring/audit

j. Local autonomy (concurrency)

k. Fault tolerance and crash recovery

l. Distributed security control/audit

m. Specific language interface (such as Ada)

n. System availability during maintenance and expansion activities

Commercial vendor claims tend to be overly optimistic about the performance and resource efficiency of DDBMS's. This assessment is the result of an evaluation of three DDBMS's which were recently completed at JSC (Williams et al., 1988). The impact of the results is that no scaled-up, real-world size application can yet be adequately supported.

Many existing DOKSS's don't make the information collection process easy enough, nor do they offer sufficient payoff to the supplier once the information is entered to provide a motivation for doing it. They are, in general, more trouble than they are worth. What is needed are DOKSS's that are less trouble than they are worth.

For design changes in particular, it is today often easier to simply try out the change and then (maybe) go back and update the design data bases. In the case of Solar Maximum, for example, perhaps no one in operations knew about the additional hardware because it had been added at the last step and never documented. Often, the documentation never gets accomplished because it simply isn't viewed as critical to the undertaking (but it is).

Organizational issues arise because design documentation is typically of least use to the original designer, at least initially, who is most familiar with the object. There should be a value structure within the organization that makes clear the importance of supplying "complete" design knowledge with the engineered system and emphasizes that, as in Solar Maximum, the consequences of even minor omissions can be serious.

The ideal is that there should be a DOKSS so useful that no one would think of proceeding without using it. Existing very large-scale integration (VLSI) design tools, for example, provide sufficiently powerful functionality that no major design is carried out without them. Even their basic functions (schematic development, capture and edit, design rule checking, and simulation models) provide sufficient payback to make them worth the trouble (Rich and Waters, 1986). An effective system is one that is useful from the earliest "sketch on the back of an envelope" stage and that aided and captured every step and decision along the way, including intended functionality, rationales for design choices, test data, operations procedures, etc.

It is generally not feasible to use, within a DOKSS, a set of disparate data bases developed for other purposes than for a DOKSS. When several mainframe computers and computer aided design workstations are used by engineers and designers in different organizations in design of a single engineered system, each computer and workstation is different with different software and input/ output formats. Coordinated naming conventions are not used, therefore the same assemblies and components have different names in the various data base systems used. The data about these assemblies and components are different, and no data model is used to ensure integrity or efficiency. A directory of all the data is not available. Access across organizations has generally not been adequately coordinated, hence communications have to be retrofitted if needed. An integrated set of user applications software is not possible, which limits the design analyses and evaluations to those available on each computer and workstation and to the assemblies and components designed on that workstation or computer.

Clearly a better approach to such a system will be to coordinate standards across organizations and do up-front design of a DOKSS.

REPRESENTATIONS AND TRANSFORMATIONS

The DOKSS must access a huge body of widely varying forms of information. The choice of a knowledge representation to support the user interface is driven by this basic need. Although they will, by

necessity, be used for many aspects of the system, traditional data base solutions (like the relational model) have the following arguments against them:

a. Data bases help organize large collections of data if the information is essentially complete (i.e., nearly all field values are known for nearly all data) and if the collection has a highly regular structure. Unfortunately, for design knowledge, neither is the case.

b. Data base techniques do not represent text or graphics very efficiently. Neither text nor graphics have a well defined bounded structure which can be molded to the idealized data structure of the traditional data models. Textual designer's knowledge such as rationale and assumptions is a major part of the knowledge to be used.

c. Functional relationships between data are not captured by traditional data base models. These must be implemented by separate tools (probably general purpose programming languages) outside of the data base environment.

d. An important use of the information in the knowledge (data) base is to support complex event and time-based models and simulations of the SSF environment. Again, data base structures are not rich enough (i.e., they lack flexibility, sufficient features, etc.) to directly facilitate this.

e. The interface with traditional data base systems is query-oriented. This means the user must know what the data base contains (and sometimes how the data base is organized) in order to use the data. Other forms of interface (such as graphic-based knowledge browsers), which are essential to the DOKSS, are not supported.

f. Data base models force the user to define the structure of the data when the DOKSS is first designed and are relatively inflexible to change. This is impossible for design and operations knowledge because the structure of the knowledge will evolve as the design and operation of SSF evolves.

Object-oriented representations allow data, their behavior, and the relationships among them, to be modeled in a flexible, but uniform way. The object-oriented paradigm is very simple: all real world entities are objects which may act independently of all other objects. Inheritance, encapsulation, message-driven activation or execution, and object identity are key technical features of this approach. An object contains structural and procedural descriptions of the entity which it represents.

Access to data stored in relational data base management systems must be provided through object-oriented user interfaces and data base management systems. Graphics and text stored in forms other than object-oriented at their source must also be provided through object-oriented user interfaces. Object-oriented system technology is further discussed in section 4.

EXAMPLES IN INDUSTRY AND GOVERNMENT

Industry and government establish the mainstream in the area of using DOKSS's for complex engineered systems. An understanding of current capability as well as future directions allows one to

obtain insights, to make contacts, to learn from the experiences of others, and to compare useful characteristics of established DOKSS's.

For example, General Motors has used DOKSS's for every car model manufactured at least since 1984. Each car division uses different data base management systems. However, designer's knowledge with emphasis on design rationale is available in machine-interpretable form.

The major impact General Motors has experienced through the use of DOKSS's is benefits to systems engineering and integration which have become more effective. Their DOKSS's are cost-effective in that the uses of design knowledge save more money than the cost of development plus operations costs, including design knowledge capture. They have the advantage of a single corporate policy to provide the designer/engineer and management discipline required to capture the essential design knowledge. Millions of dollars have been saved in the manufacturing area alone due to systems engineering and integration quality improvements.

Chrysler Corporation has had similar experiences with DOKSS's and systems engineering.

Major architectural and engineering firms have also found they can deliver typical projects under budget and in less time than normally scheduled by adopting such systems. Two reported benefits of DOKSS's are that they shorten the design cycle and they help to accomplish higher quality products in the time allotted.

A key aspect of DOKSS's comes from integrative technologies and capabilities which enable diverse organizations to derive the benefits of access to each other's design knowledge. A major contributor in this is data integration standards, in which there has been considerable progress in adoption in recent years. These standards are discussed in section 4. Three of them are mentioned here: IGES, EDIF, and PDES.

The Initial Graphics Exchange Specification (IGES) efforts can be traced from 1979 to the present. The National Institute of Standards and Technology (NIST) (formerly the National Bureau of Standards) supported this effort begun by manufacturers and software vendors. IGES is the defacto standard for graphics.

The Electronics Data Interchange Format (EDIF) covers both semantics and graphics. Efforts on EDIF stem from 1983 to the present. EDIF may provide possible integration of hardware and software representations.

The Product Data Exchange Specification (PDES) efforts were originated by an industrial team in 1984 and continue to the present. These activities involve use of object-oriented data bases (see section 4). The effort was encouraged by NIST and is now funded by DOD and driven via the Computer Aided Acquisition and Logistics Support (CALS) program.

Boeing Corporation is experienced in knowledge management for complex, long-life systems such as their commercial aircraft. Boeing has been, and continues to be, developing software tools for

designers. One focus of these tools is an integrated design environment where designers from different disciplines "talk" the same language about design. They further have developed a designer's assistant tool for electronics which they are expanding to mechanical systems.

Boeing also has a demonstration system that includes automated tools for requirements specification and knowledge capture. Boeing continues to have a long-term commitment to improving design knowledge methodologies. The simple reason is that they improve quality and are cost-effective.

The Air Force has been supporting a major program called Integrated Design Support (IDS) which is an integration technology program focussed on quality improvement in aircraft systems through capture, management, and communication of technical data from design through logistics operations. There is a 10-member contractor team, headed by Rockwell International, developing a prototype IDS system for the B-1 bomber. There is also a 20-member technical advisory group which is made up of major aerospace companies.

The objectives of IDS involve data models, data management, communications and networks, standards, and security. IDS is built around a data driven methodology. A large, dynamic, comprehensive model for technical data has been produced called the Product Data Control Model (PDCM).

In 1988, Boeing became the third contractor involved in the tri-service Reliability and Maintainability in Computer Aided Design (RAMCAD) program. Boeing is developing a prototype software system to aid designers in assessing reliability, maintainability, and supportability requirements of military digital electronic systems during design. Lockheed Electronics has developed an artificial intelligence-based system for automatically performing and documenting, in MIL-STD-1629A format, a FMEA and criticality analysis of circuit designs captured on computer aided engineering workstations under the RAMCAD program.

The DOD CALS program is an activity independent of IDS (although related) and is a major effort, being funded at several million dollars per year. The CALS program has been the driving force behind the establishment and accelerated completion of PDES, which would serve as the standard for product definition information. The goal is to have the defense contractor community make a commitment to adopt and support PDES voluntarily, but with possible future contract measures of adoption if not supported. This clearly has implications for NASA programs with these contractors. NASA has been informally participating in some PDES meetings.

CALS objectives include developing computer aided engineering tools to automate logistics functions, integrating automated reliability and maintainability engineering into contractor computer aided engineering systems, and reducing the amount of paper work, such as manuals, drawings, and volumes of reliability and maintenance data. DOD is moving forward to make CALS the standard way of doing business and has chosen five new weapons systems as test beds for the emerging CALS standard: Navy SSN-21 attack submarine, V-22 Osprey tilt-rotor and A-12 aircraft, Air Force advanced tactical fighter, and Army experimental light helicopter.

OWNERSHIP AND ADMISSIBLE BUSINESS RECORDS

The ownership of the design and design knowledge rests with the sponsor of the engineered system unless retained by the company developing the system. The individual engineer or designer generally only possesses ownership rights within universities. For government sponsored systems, not only the agency involved has ownership, but access rights exist for the Inspector General and by subpoena for congressional oversight committees. The courts also have access rights in disputes.

From a business records perspective, the Federal Business Records Act (28 U.S.C., section 1732) states that if any business or government agency has kept records "in the regular course of business" and "has caused any or all of the same to be recorded, copied, or reproduced by a photographic, photostatic, microfilm, microcard, miniature photographic or other process which accurately reproduces ... the original, the original may be destroyed in the regular course of business, unless its preservation is required by law. Such reproduction, when satisfactorily identified, is as admissible as evidence as the original itself in any judicial or administrative proceeding, whether the original is in existence or not." While not a conclusive statement about electronic forms of design knowledge, this seems to be an indication that the Federal and state judicial and administrative systems will adapt to electronic records as acceptable admissible business records. However, a legal concept called "statute of frauds" says essentially that contracts have to first be in writing. Secondly, they have to be signed. Thus, for contracts themselves, Congress must provide the legal backing; then the courts will set new precedents. But it appears design knowledge may not save this same issue. Users will need to analyze the benefits for themselves.

SECTION 4

REVIEW OF THE UNDERLYING TECHNOLOGY FOR DESIGN AND OPERATIONS KNOWLEDGE SUPPORT SYSTEMS

Design knowledge encompasses not only what the substance of a design is, but also how and why it satisfies functional requirements. Design knowledge is composed of design objects, their descriptions, and declarations or assertions called "designer's knowledge." Designer's knowledge may include functional requirements, criteria or intent for selection of a design approach or solution, declarations of analysis results and conclusions, and assertions concerning expected design object behavior.

Design knowledge can be captured in interim forms which are not optimal for knowledge-based processing, but which allow productivity gains to be initially realized using widely implemented data base technology. Such representations may later be translated for use with knowledge-based, or other advanced software, with minimum risk for information loss (Wechsler and Crouse, 1986).

Advances toward a design knowledge capture capability have been iterative, taking advantage of developments in the areas of

a. Requirements awareness

b. Computing systems

c. Software technology

Advances have been made when this triad progressed synergistically. When coordination lagged, each area was ultimately limited by the others. This section describes the progression and present state of requirements awareness, computing systems, and software technology, respectively, and addresses important events and impacts arising from convergence of these three areas.

REQUIREMENTS AWARENESS

What is presently referred to as design knowledge capture had its beginnings in the early 1960's with the emergence of two technologies: design automation and numerical control. By the 1970's, the manufacturing organization, in the role of an internal customer of the designer, pushed for improved data availability through the interface of CAD/CAM systems. This evolution in awareness led to an "enterprise perspective" in the 1980's, encouraging the advent of computer integrated manufacturing (CIM) through vendor-independent integrated data bases. By the mid-1980's, the scope of design knowledge capture extended to a life cycle requirement for product information. For example, knowledge generated during product creation was vitally needed for maintenance and operation of the product. Moreover, the need for design knowledge to move across company lines made design knowledge a "deliverables" issue.

The requirement for design knowledge has extended to capture of standards for the construction industry. The need exists to retain the critical design information of, for example, a nuclear power plant or a chemical refinery. The Center for Building Technology of NIST has developed generalized software for the management of requirements, such as those defined in building standards. A standard is defined by NIST as a structured compilation of principles and consensus judgments from representative technical domain experts, for the purpose of guiding decisions related to the domain. The Specification Analysis, Synthesis, and Expression System (SASE) was released in May 1987 (Fenves et al., 1987). SASE utilizes data items, provisions, decision tables, decision trees, functions, and networks to produce a model of a standard. NIST has characterized SASE as a knowledge-based system (Lopez et al., 1985).

Computer Aided Design

The basic concepts of design automation date back to Ivan Sutherland's work beginning about 1959, on the Sketchpad system at Massachusetts Institute of Technology's (MIT's) Lincoln Laboratory. A parallel development involved IBM's design augmented by computer (DAC-1) prototype system built to General Motors' specifications for delivery in 1963.

Through the 1960's, graphics-based systems capable of creating "electronic" drawings began to emerge. Though slow and expensive, these mainframe-based systems were used in special instances where they were justified (Prince, 1971).

Over the past 25 years, the automation of design information storage and retrieval has progressed from electronic drawing files, to CAD data files, to CAD/CAE integrated data bases.

Specifications for Data Exchange Between CAD Systems

Early CAD systems were based on custom methods for accessing individual files and records. Following a period of proliferation of these systems in the 1970's, CAD customers were faced with the sizable problem of communicating electronic designs among incompatible equipment. The solution of a few aerospace companies was a vendor-neutral, common file format. This was the conceptual basis for the IGES.

Initial Graphics Exchange Specification

The IGES is a file specification. To implement IGES, a CAD vendor must provide the translators for two-way conversion between the IGES form and the vendor system's native file format (National Bureau of Standards, 1986a).

The IGES organization is a volunteer group engaged in development of specifications with support from NIST. In its 1979 initiation, the IGES group addressed the problem of exchanging two-dimensional drawing representations between unlike CAD systems. Subsequent additions to the IGES specification have encompassed associativity, connectivity, and three-dimensional models.

The fourth release of IGES occurred in January 1988. After release, the IGES is provided to the American National Standards Institute (ANSI), via the Y14.26 Subcommittee of the American Society of Mechanical Engineers (ASME), for approval as a national standard.

Early CAD was more accurately characterized as computer aided drafting than as design. However, by the mid-1980's, general purpose data base management systems (DBMS's) began to provide the improvements demanded by increasingly complex engineering applications. As a result, CAD began to fulfill its potential as a design tool, by linking the product geometry from CAD with the principles of engineering and physics from CAE.

Product Data Exchange Specification

By mid-1984, the IGES group had recognized the need for broad-based application support for the maturing CAD/CAE data base technology. The result was the formation of a project, chartered to communicate a complete product model having sufficient information content as to be directly interpretable by advanced applications of the CAD data base. This project is named the PDES (Long Range Plan, 1986). PDES depends upon requirements input from the users of the design information.

In 1983, a group of electronics companies, frustrated by data interchange problems, formed the EDIF committee. The EDIF committee has developed and published an interchange format for electronic design data. A primary use of the initial EDIF specification was to pass schematic capture output to simulation systems. Version 2.0.0 was published in May 1987 and is now approved as ANSI/ Electronic Industries Association (EIA) Standard 548 (Electronic Industries Association, 1987). Limiting EDIF to electronic product data has led to a more streamlined description. Numerous EDIF implementations are expected by 1989.

The PDES organization contains a subcommittee for electronics. The aim of this strategy is to integrate data aspects peculiar to the electronics industry into the overall PDES model, rather than to create an industry-specific model.

Computer Aided Manufacturing

Awareness of information requirements has evolved from the geometric description of parts, to the support of stand-alone manufacturing programming, to manufacturer recognition of requirements for an integrated product data base.

Advances in the availability of design information were matched in the development of manufacturing systems. After the advent of the digitally controlled milling machine at MIT in 1958, numerically-controlled equipment was commercialized throughout the 1960's. "Part programming" languages such as automatically programmed tools (APT) came into use. By 1965, the first production part was programmed using part definition data from the CAD system (Prince,1971). By the 1970's, APT software could generate standardized cutter location data (CL file) from the part design information in the CAD system. The CL file could be interpreted into data directly usable by the machine tool (Stover, 1984).

Development in the 1980's has aimed at automating the link of CAD with part planning and manufacturing. This development centers on the concept of a "part feature." A feature is a region of interest of the part. Feature definitions can be supplied to manufacturing logic software, to automatically generate instructions for discrete part manufacturing processes such as drilling, milling, turning, bending, routing, robotic handling, or combinations of the preceding (Latombe and Dunn, 1984).

Three elements are required to implement this approach:

a. A features taxonomy

b. An automatic extraction method

c. The integrative data base technology capable of handling both the part-feature structure and manufacturing logic

In 1985, cooperative development of Computer Aided Manufacturing-International (CAM-I) produced the "Requirements for Support of Form Features in a Solid Modeling System" (Computer Aided Manufacturing-International, 1985). This specification provides a common basis for implementation of feature expression capabilities in commercial geometric modeling systems. Cognition and General Electric have developed Casper, a features-based manufacturability aid used for design of aluminum castings (Luby et al., November 1985). Casper allows the designer to create the part using features as "building blocks."

A method to extract features from three-dimensional CAD data has been described by Mark Henderson (1984). Henderson's work has provided a basis for continuing research.

The majority of PDES participants from the manufacturing industry elected to focus on data base integration for design and manufacturing. Field support was included within the PDES scope, but it initially received low priority (National Bureau of Standards, 1986b).

Integration of CAD/CAM Product Models

Integration of the solutions brought by technology will require discipline in the application environment. CAM-I has reported (Industrial Automation Standards Workshop, 1987):

The lack of adequate and effective industrial automation standards is rapidly becoming a serious situation throughout industry. The economic well-being of a nation depends largely upon productivity and competitiveness; productivity and competitiveness depend upon exploiting advances in manufacturing automation technology; and technology exploitation depends to a large degree upon adequate industrial standards.

Large organizations and communities of organizations wishing to mix and match multiple vendor solutions must standardize to succeed. The NASA community, for example, will require effective standardization of interfaces for passing text and graphical data between systems.

To identify what and when to standardize, it is necessary to forecast the break-even point where technology creativity gains are offset by losses from nonstandard approaches (Goodstein, July 1987).

Computer Aided Acquisition and Logistics Support (CALS)

DOD has embarked on a 10-year strategy to develop networks of shared systems between industry and its government customers, eventually using "intelligent" data as the exchange medium. DOD concluded that it is preferable to move common digital data between computer systems rather than move hard copy information between life cycle functions. Phase I of CALS, based on file exchange standards, is presently being implemented. Phase II will be based on an integrated product data model (CALS Core Requirements, 1987).

DOD has recognized that efforts such as PDES will be critical to CALS. In June 1987, DOD encouraged its suppliers to form a funded cooperative to accelerate the development of the PDES model (Draft Prospectus, 1987).

As an intermediate step, CALS has achieved a standard mark-up language for placement of documentation into proper locations and annotations to support later conversion to machine readable form.

The IDS Program, initiated by the United States Air Force, is developing a PDCM. The goal of IDS is to provide a design support environment to complement CALS. IDS developers are a coalition, in which Rockwell International is the prime contractor. IDS is reviewed regularly by industry representatives (Integrated Design Support Prospectus, 1987).

In its concept of design knowledge capture, NASA has extended the scope of life cycle applications by suggesting that product information could be translated to knowledge representations for active operational support, as well as for passive referrals (NASA Space Station Program Office, 1988). However, NASA has yet to develop or adopt a product model or to encourage supplier directions as done by DOD.

COMPUTING SYSTEMS

Key advances in computing systems have been made which support design knowledge capture, storage, and use. Powerful yet economical personal workstations must accommodate the capture system, while providing technical support for the individual engineer. Online storage systems must have sufficient capacity to handle the large amounts of information which will become available. Page-to-disk approaches, combining document scanning and online storage technologies, can facilitate transition from manual methods and substitute where direct capture methods cannot be applied.

The Designer's Computing Environment

During the 1970's, the interactive design environment shifted from a predominantly mainframe to a predominantly turnkey minicomputer environment. Throughout the 1980's this evolution has extended to workstations and personal computers (PC's).

By 1988, the number of PC's and engineering workstations installed for CAD applications will exceed the number of turnkey, host-based stations. PC's and engineering workstations will account for 60.2 percent of all CAD displays in use (Large Organizations, May 1987).

The ratio of workstations to technical professionals is improving, as shown in table 1 (Zengerle, 18 May 1987):

TABLE 1.- RATIOS OF WORKSTATIONS TO TECHNICAL PERSONNEL

Company Size (No. Employees)	Year 1986	1987	1990
100-1000	1:6	1:5	1:3
Over 1000	1:9	1:7	1:6

Driving this increased availability are capable technical workstations and PC's with a system (hardware, software, and enhancements) cost of less than $15,000. The figure $15,000 approximates the technical computing market's threshold of allowable capital-investment-per-worker (Gantz, 1 June 1987).

The functionality of 1970's-style CAD systems now can be provided on PC's which cost less than $5000. Solid geometric modeling and engineering analysis systems, such as in the Aries Technology Concept Station, are beginning to appear on enhanced PC's and workstations at prices approaching the allowable capital investment threshold (Concept CAD on a Desktop, May 1987).

Further, the boundaries of capabilities between workstations and PC's have begun to blur. The term "personal workstation" characterizes a new category of equipment, which encompasses both the minimally configured technical workstation and the fully configured personal computer. Table 2 describes three representative computing systems in this category (Gantz, 7 September 1987).

TABLE 2.- CANDIDATE PERSONNEL WORKSTATIONS

Characteristics	Sun 3/50M	IBM PS/2-80	Apple Mac II
Millions of instructions per second	1.5	2.0	1.2
Memory expansion (millions of bytes)	4.0	16.0	8.0
Maximum disk memory (millions of bytes)	282	230	80
System price ($ thousands)	10.0	9.0	9.0

The impact of the preceding developments will be to increase system capability and intelligence for an increasing number of designers (Weston and Stewart, February 1987).

A greater portion of personal computers and workstations will be installed in departmental and corporate computing networks. Networking will require more sophisticated administrative management and control procedures (Flynn, 1 June 1987). However, networked PC CAD is currently lagging. AutoDesk estimates that only 15 percent of its customers presently use AutoCAD in a network.

Reluctance to employ networks may reflect user uncertainty with networking technology. When PC's are connected in local-area networks (LAN's) (Derfler, 9 December 1986), PC response degrades with increased network activity (Datapro Research Corporation, June 1987; Stone, 14 September 1987). An alternative is a multiuser operating system such as Xenix. However, Xenix, a licensed subset of UNIX, is larger than PC-DOS/MS-DOS, and it demands more system resources. Its application software is more expensive and limited in availability (Foster, 1985).

Anticipated Advances

The rapid advancement of cost/performance in microprocessors and semiconductor memory devices has left other hardware areas behind. Areas offering the most potential benefit for design knowledge capture are in graphics, online storage, and input technologies.

Graphics

Graphics processing is steadily migrating from software to hardware. Essential low-level routines which had been implemented in graphics software engines are now candidates for residence in read-only memory (ROM).

Special purpose graphics co-processors are also available, such as the Intel 82786 and the Texas Instruments TMS34010. These devices can take responsibility for screen management, leaving the microprocessor free to handle transformations for manipulating complex images. Graphics co-processors, combined with the capabilities of next-generation microprocessors, can provide for advancement of the following capabilities:

a. More powerful display applications, such as solid modeling on low-cost computers

b. Larger color selection

c. Increased resolution

Graphics resolution has another limitation: the display monitor. Components of the monitor, such as the cathode ray tube, have not benefited from the advances in semiconductor technology which have forced memory costs down. As semiconductor processing capability increases to support higher resolution, the monitor will become a proportionally more expensive component of the system. Assuming current display technology, the entry-level system for professional CAD/CAE, with a 1024- by 1024-pixel monitor, will remain cost-differentiated from the system expected to satisfy most business application requirements with a 640- by 480-pixel monitor (Cummings, 28 July 1987). Advances in

graphics hardware technology will be complemented by graphics software advances. Key software development areas include graphics file compression, graphics image data base management, and pixel-based (raster) to vector-based graphics conversion.

Online Storage

The increase in the file size of graphics, as compared with text, is measured in multiple orders of magnitude. Without compression, the pictorial definition contained on a single 8-1/2- by 11-inch sheet (at 300 pixels/inch) can represent the equivalent of more than 1 megabyte (MB) of disk storage (Lockwood, June 1987). An assembly drawing in a CAD system can readily consume 2 to 4 MB. Therefore, pictorial or graphics-intensive applications may require a substantial increase in online storage capacity.

Over the next decade, mass storage technologies will produce significant improvements in cost reduction and access to large data bases. Optical storage technology currently offers the greatest technical potential to meet this demand. Optical technology uses a laser device to read and inscribe the disk media, instead of an electromagnetic device, which alters the media's arrangement of magnetic particles.

The success of compact disk technology in the entertainment industry led to interest in using this medium to store digital data. By 1985, this interest resulted in the introduction of compact disks-read only memory (CD-ROM). Early CD-ROM commercial products were hardware-dependent because they used different disk formats for data organization. In 1988, the International Standards Organization's (ISO) Standard 9660 CD-ROM format was approved.

Publishing a CD-ROM disk requires an elaborate "mastering" process which permanently imprints data on the disk. Thus, CD-ROM data are not often user-created and not user-alterable. In contrast, a user can place data on a write-once, read many (WORM) disk. WORM technology uses a multilayered metal film media. The laser marks the media by melting the metal in selected spots. However, data written on a WORM disk cannot be erased.

Developments are in progress for erasable optical media. Several manufacturers have begun pilot production for phase-change, and magneto-optic (M-O) erasable disks. The Tandy Corporation has announced the THOR development project, to produce an erasable optical disk readable with existing CD-ROM equipment. Commercially available erasable media are expected by 1990.

Table 3 shows the typical differences between magnetic media and CD-ROM storage (Zoellick, May 1986).

Limited space availability for archival data storage favors electronic media over paper. WORM technology is advantageous when the amount of data is relatively large, the data are stable, and access time is not critical. This technology is increasingly competitive with current microfilm archiving processes, while providing the additional benefits of online access and retrieval.

TABLE 3.- PARAMETERS OF DISK TYPES

Disk type	Capacity (millions of bytes)	Average seek time (milliseconds)
Average PC fixed disk	10	100
High perf. magnetic disk	456	28
CD-ROM	540	500

Since each of the preceding disk technologies possesses a different set of advantages, disadvantages, and maturity timetables, future systems can be expected to contain more than one type of online storage device. Architectural extensions such as the Small Computer System Interface (SCSI) will be required to handle integrated multidevice storage (Laub, May 1986). SCSI is documented as a national standard, ANSI X3.131-1986.

Data Input

Most design-related data will be provided directly by the human. Human interfaces with computers have progressed from the flipping of switches, to text and keyboards, to graphical interfaces. Use of a digitizing device (such as a mouse) with displayed graphics is presently the preferred interaction method (Cummings, 28 July 1987). A number of specialized digitizing devices, such as graphics tablets, are used to streamline CAD user input.

Interface alternatives such as voice input hold considerable potential, but have not progressed as rapidly. Multisensory approaches, such as the combination of audio and graphics, have not been effectively exploited in current systems.

Page-to-disk technology can provide automated data conversion from manual to electronic media and computerize storage for unstructured ancillary data. Page-to-disk approaches will transform data from paper to electronic media without intermediate manual processing. This technology combines mass storage devices (typical optical storage) with the following elements (Stanton et al., 30 September 1986):

a. Optical scanning unit. (The scanning unit senses the incoming document and translates the light reflected by lighter areas of the page into binary "gray scale" data.)

b. Image capture/recognition software. (From the scanner's digital data signals, the software produces either a bit-image representation of scanned graphics or an American Standard Code for Information Interchange (ASCII) code of scanned characters. Algorithms may exist for font recognition, conversion of half-tones to gray scale levels, and page image compression.

Scanner technology is undergoing rapid, but continuing development. Commercially available units do not regularly produce error-free results. Manual monitoring of the scanner's interpretations should be planned.

Areas of recent achievement and current development of commercial scanner capabilities include (Stanton, 13 October 1987)

a. Combined, single-pass text and graphics scanning

b. Selective scanning within a page

c. Handling of 9- by 9-inch dot-matrix character input

d. Word processor-compatible formatting of scanner output

e. Raster-to-vector conversion of scanned graphics

f. Improved data compression

SOFTWARE TECHNOLOGY

An increasing level of intelligent automation support for design synthesis, analysis, and representation can help decrease the interpretation required from the human to the capture system by permitting it to work through, and integrate with, design automation systems. Initial products and ongoing developments for intelligent design automation are described. Technology requirements for design integration and representation are identified. Object-oriented language and data base technology is presented as having considerable potential for many of these requirements.

Productivity Aids for Designers

Automation in design can improve efficiency. A higher level of automation in the design process also increases the potential of automating knowledge capture by decreasing the need for the designer's intervention. It is important that knowledge capture must be performed routinely, without requiring special effort on the part of design engineers.

Productivity also encompasses the potential for intelligent automation to improve the effectiveness of design decisions. A British study indicated that up to 85 percent of the product's cost is committed during the initial 15 percent of the design effort (Esplin, 7 September 1987). This condition suggests a need for automation support of design decision-making early in a product development.

Systems are beginning to appear which support the full range of engineering functionality. Traditional integration of drafting with engineering analysis is now supplemented with intelligent sketching tools, equation solvers, online handbooks, and report writers. Software for manufacturability analysis is also emerging. Most significantly, recently added capabilities address the

conceptual design stage of the design process, where automation has been most lacking (Front-End Mechanical, 31 May 1987).

Intelligent Design Aids

Current products increase the productivity of the designer by internalizing basic knowledge related to the design task. For example, "Mechanisms," a product of McDonnell Douglas Industry Systems Company, can create mechanical linkages, given the required motion vectors. The "Expert Cost Guide" in Cognition's Mechanical Advantage 1000 System provides relative cost estimates for various design alternatives (Steinke and Schussel, November 1985). Synergist, a product under development at Intelligent Applications, Ltd., captures electronics circuit diagnostics logic during the design stage for use in simulation, testing, and fault diagnosis (Rawsthorne, May 1987).

Designer's Assistants

The designer's assistant software environment provides for a system interaction with the designer in which the system advises on design constraints. The designer's assistant can retain knowledge as rules which state the designer's intent.

With the design knowledge captured in the ICAD's intelligent system product (Rosenfeld and Belzer, 1985), Hudson Products Corporation has utilized consensus design rules to automate the design process for an entire product line. In ICAD, design rules can be captured in classes of objects under object methods or definition of object attributes. Symbolic referencing is the mechanism by which one part of a tree hierarchy can refer to attributes, methods, and subparts defined in another part. When writing a rule about an object and its operations, the user can create a symbolic reference to attribute methods or subparts of another object.

LEAP is a prototype knowledge-based assistant for circuit design begun at Rutgers University. LEAP learns during user operation by acquiring rules from the user's manual overrides of LEAP's suggestions. LEAP then attempts to generalize these rules (Mitchell and Mostow, 14 July 1987).

The United States Air Force's Super Cockpit Program has announced plans to develop a "designer's associate," beginning about 1990 (Boff, 1987). This system will

a. Provide automated data management for designers

b. Function as a problem-solving partner

c. Maintain a corporate memory of design decisions

Integrative Technologies

Added capabilities are desirable for design knowledge support systems. Current DBMS technology is widely available and generally understood. However, certain requirements which are difficult to implement with current data base technology are more readily met with emerging technologies.

These requirements include

a. Representation of behavior shared between design objects

b. Representation of multiple relationships, such as a physical parts breakdown with a functional decomposition

c. Controls such as inheritance, which are internal to the representation system

d. Ease in handling structural change

e. Facilitation of an object-based design environment

In addition, integrative requirements for root data base technologies include

a. Data base information representations for knowledge-based applications

b. Integrated data base/inferencing systems

c. Integrated storage of engineering graphics information

Data Base Inferencing Systems

Bridges are needed between expert system knowledge bases and data base management systems. Schema translation is a promising approach for bridging. By comparing programming language commands with data base operators and query commands, researchers have developed a mapping of schema between the two. This mapping provides a basis for schema translation (Boas, January 1986).

Artificial Intelligence Technologies Inc.'s Mercury Knowledge Base Environment (KBE) product facilitates the development of medium and large-scale expert systems. This is achieved by combining a high performance rule-based inference engine with persistent (relational data base) storage. Mercury KBE permits a knowledge-based application to initiate relational data base queries and to be tightly integrated with existing software tools and technologies (Patch, August 1988; Kennedy, September 1988).

Sharing of schema can also provide for the integration of data base and inferencing systems. Meta-schema of the common data base could service a number of knowledge-based applications (Rehak et al., 1984). Portions of the inferencing procedure could also be integrated in the data base system. Then application development would consist of defining the appropriate goal statements and confirming that the supporting descriptions are in the data base.

Object-Oriented Programming

Object-oriented programming can help to reduce the semantic gap between the programming tool and the real world which its applications are supposed to model. This improvement can result in increased

programmer productivity and easy program maintenance. More important from a user viewpoint, such program organization provides rapid and accurate understanding of the content by persons not initially familiar with the program.

Below is a sampling of languages to which some object-oriented elements are attributed (Booch, 1986; Cox, 1986; Stefik and Bobrow, 1986):

a. Smalltalk-80 or /V

b. Flavors

c. Loops

d. ExperCommonLISP

e. Modula-2

f. Objective-C

g. C++

h. Ada

The preceding list includes several extended conventional languages, as well as languages initially developed as object-oriented. Capabilities possessed by each language vary considerably. Smalltalk-80 provides a fully object-oriented development environment.

An object consists of data private to that object and of a set of operations which can access the private data. A "consumer" object must request a "provider" object to perform one of its operations by sending it a message telling it what to do. The provider object responds by choosing and executing the operation and returning control to the consumer.

A fully object-oriented programming language has three major features:

a. Encapsulation

b. Inheritance

c. Message-driven computation

The item "encapsulation" refers to support for data abstraction or information hiding which is a key feature of object-oriented systems. Each object is composed of a private memory which contains the data structures and code associated with the implementation of the object and a public interface which provides (limited) access to the object. Users only need to know the names of functions (usually called "methods" in object-oriented systems) in the public interface and "not" the details of the implementation which are hidden in the private memory. Objects are arranged into classes which are arranged

into a hierarchy. This hierarchical system structure (which is a tree in the case of Smalltalk-80) is the mechanism by which subclasses "inherent" structures and "messages" (method interfaces) from their superclasses. Objects in a class are "instances" of that class and as such inherit all instance attributes (structures and messages) from their ancestor classes. This ensures that all instances of a class respond to the same set of messages. A "message-driven" model of computation has very clean and potentially highly parallel semantics. Unlike conventional programming languages, true object-oriented languages do not require a centralized control mechanism because each object is "active" within the environment. Objects respond to and generate messages independently of each other.

Hierarchy and encapsulation promote modularity, eliminate scoping problems and hide unessential details from users and programmers. Finally, Smalltalk-0, Flavors, and LOOPS are "untyped" languages, which makes them extremely flexible. Because binding can be delayed in untyped languages, the design of data structures can be easily modified during the development cycle. A dynamic or delayed binding capability could become particularly useful in applications such as sensor-based controls, where pre-written code cannot anticipate the type of data to be operated on until after the code is compiled (i.e., during run time).

The representation of a design object in an object-oriented language can contain a method to produce a graphic display of that object. Currently, CAD systems are the primary automated means of creating graphic representations during the design process. Therefore, the CAD system is the source for graphical data. The data captured by the CAD system could be translated and imported into an object-oriented environment for execution by the object's "draw" operation. This approach suggests the possibility for implementation of CAD systems in an object-oriented environment (Kaehler and Patterson, August 1986; Digitalk, Inc., 1986).

Object-Oriented Data Bases

Classic business problems are well-suited to relational, as well as hierarchical and network data base management systems. These problems possess highly repetitive records, data items with well-defined formats, and simple interactions. These types of data models do not map well with the requirements of engineering and manufacturing: many data types, complex interactions, and interactive applications demanding fast retrieval of small amounts of data. However, the absence of supporting facilities for object-oriented data storage has been a major shortcoming in the use of object-oriented languages.

In 1987, Dr. Mohammad Ketabchi defined a set of requirements for DBMS support in engineering (Ketabchi, April 1987). These requirements can be initially interpreted as the design engineering requirement for object-oriented data bases. Current development of object-oriented data models can be classified on three levels (Peterson, March 1987):

a. Structurally object-oriented. (A data model which allows data structures to represent entities of any complexity.)

b. Operationally object-oriented. [A data model which includes operators to deal with complex objects in their entirety, without requiring decomposition to simple objects. Efforts such as Postgres (Greenstein, 20 April 1987), to extend relational models, will likely produce operationally object-oriented models.]

c. Behaviorally object-oriented. [A data model which incorporates capabilities to define object types. The information private to an object instance can be used only by implementing operators which can exploit this information. Efforts such as GemStone (Servio Logic Development Corporation) (Andrews and Harris, October 1987) to extend the concepts of object-oriented programming produce behaviorally object-oriented models.]

By 1987, at least two object-oriented systems were in the latter stages of commercial development. These systems are V-BASE (Ontologic, Inc.) (Maier et al., September 1986) and GemStone.

Object-oriented data bases are clearly evolving. Issues for production use include (Peterson, March 1987)

a. Lack of a coherent data model founded in mathematical principle, such as relational calculus. This raises an issue of data base consistency.

b. Efficiency of object storage

c. Difficulty in interfacing with established languages

d. Absence of an accepted methodology for development of object-oriented systems

Hypermedia

Hypermedia is a form of electronic document in which data are stored as a network of nodes connected by links. Nodes can contain text, source code, graphics, audio, video, or other forms of data. Hypermedia documents are normally meant to be written, stored, retrieved, and read within a computing environment. Thus, they spend their entire life online rather than on paper.

Supporting both electronic and conventional paper forms of documents is a key aspect of any current system. While electronic documents may eventually replace paper ones, that day is not at hand. Even in organizations in which professionals work within a network of workstations, paper documents continue to be important. Many users prefer to edit on paper rather than on screen. Most internal documents must be printed for upper management to read them. And most documents that go outside the organization still go out through the mails rather than through a network.

There is also an interest in the cognitive processes of its users that is associated with hypertext systems. This is because the users of hypertext documents must still understand what they read (or see, or hear,...) and must construct relations between new information and old, and between one idea and another.

The underlying model for most hypertext systems is a directed graph in which content units are associated with the nodes and the sequences in which the reader may access them determined by the links. However, a network of information has properties very different from those of a hierarchy. By definition, a hierarchy addresses a single, high-level concept or purpose. Thus, it is well suited for writers who wish to make a single point or produce a specific action by their document. A network has

no such central thrust. Rather, it is an environment in which different readers may immerse themselves for different purposes and with different expected results. Thus, the emphasis is on the experience of the reader rather than any specific motivation for action. Hierarchical documents, on the other hand, provide the reader with a sense of the whole by including high-level overviews that describe what will follow. Structural information of this sort does not exist in a directed graph.

Hypermedia systems are in a state of very active growth. The memory and speed requirements for a hypermedia system are very large and only recently have there been implementations that work in a reasonable amount of time. Hypermedia systems with the hierarchical features and ease of use required for our application are not yet commercially available.

CONVERGENCE AND ITS IMPACT

There has been a progression of information requirements awareness, in terms of design knowledge-related technology developments. A trend toward convergence is described, in which newer technologies are applied, and technology developments are shared across application domains. The need for interface standardization is identified at this stage of development. A sample session for system design synthesis, analysis, and capture illustrates the benefits to be derived from convergence.

Object-Oriented Systems Design

The evolution of design knowledge requirements, as represented in the product data model, has been a major advance. Object-oriented data base technology can provide a tight integration between the data model and the data base system. Rather than creating models to be later implemented in a relational data base system, the developer may use an object-oriented data base in constructing the model.

Previously described technologies will provide the capability to implement an increasingly intelligent, object-oriented design process. Data General (McCaskey, 19 March 1987) and Hewlett-Packard (Mladejovsky, 19 February 1987) have described developments which support this type of process for design of electronic systems. Object-oriented design can revolutionize the style of computer support to engineers.

Object-oriented data base systems will provide the opportunity to produce object-oriented implementations of product data models. The Advanced Manufacturing Research Facility (AMRF) of the NIST has begun an implementation utilizing the GemStone system and preliminary information from the PDES project group of the IGES organization (Clark and Ressler, 1987; Ressler, 1987). MITRE has initially considered the development of prototype object-oriented data bases by extending product data models according to the baselined concepts of design knowledge capture (Wechsler, August 1987).

In 1987, the Engineering Information System (EIS) Program began. EIS has a decidedly electronic product orientation. The Air Force contract team headed by Honeywell will provide the services and specifications necessary to support computer aided engineering design. This effort will include

a. Framework for tool integration

b. Tool portability

c. Uniformity of the design environment

d. Exchange of design information

EIS plans to develop an Engineering Information Model (EIM) and to implement it in an object-oriented data base system (Linn and Winner, 1986).

A future object-oriented design session might proceed this way. The designer approaches the design system with concepts and specifications of functionality. While incapable of expanding the design at this stage, the designer may express his/her viewpoint through voice or written text, drawn or scanned graphics, rules, programs, or stored procedures. The system recognizes the elements created as objects and retains graphical and nongraphical information about the objects and their behavior. Instead of dealing with CAD files and CAE programs, the designer requests analysis in terms of the objects to be analyzed. The design system aids such synthesis by assisting in the evaluation of the analysis results and by suggesting alternatives. Minimal human interaction is required for routine tasks, such as generation of pre-defined manuals and reports. As a by-product of this process, the designer's expressions and analysis are retained by mass storage devices for future use.

The most difficult aspect of implementing this scenario may be in overcoming the human resistance to cultural change to achieve this level of automation.

Computer Aided Software Engineering

Computer Aided Software Engineering (CASE) is a generic acronym for a grouping of software programs which automates parts of the software development process. CASE has been characterized as "...CAD/CAM for code smiths" (Stamps, 1 July 1987). The parallelism of CASE and CAD/CAM evolution confirms the underlying commonality of product and software design processes.

CASE also has application for control software development. Hughes Aircraft Company used Cadre Technology's Teamwork CASE environment to eliminate inconsistencies from a model of factory management control (CAM-I News Alert, September 1987). Processes for developing real-time equipment controls are judged as 80 percent identical to data processing development processes (Rinaldi, August 1987). For uniquely real-time elements, special CASE aids such as state transition diagrams and matrix graphs have been provided (Index Technology Corporation, 1986).

In the early 1980's CASE consisted largely of a graphics-based diagramming capability. First experiences with CASE showed productivity gains similar to early CAD, but first-generation abstraction

tools did not lead naturally from structured analysis and design into data base design or code generation (Orr, August 1987).

CASE graphics tools are now being linked with the logic of software design. Data dictionaries in current methodologies provide storage for functional elements of the software design, as well as of data definitions (Index Technology Corporation, 1986; Ross, 1987).

Yet, integrated life cycle CASE software is still needed to support all phases of development, from planning and requirements definition through testing, integration, and maintenance (Orr, August 1987).

Automated Code Generation

CASE systems which generate machine-readable code are becoming available. In these systems, CASE provides software design rules with the data dictionary (Stamps, 1 July 1987). For example, Pansophic Systems has agreed to link its Cobol code generator to Cadre Technology's Teamwork and Index Technology's Excelerator CASE tools (CASE Accord, 14 September 1987).

CASE Data Interchange

Visual presentation is a significant feature of CASE design analysis, for depiction of software objects which do not have an inherently visual representation (Chang, January 1987). However, utilization of visual representation raises within CASE the same problems of data exchange experienced within the CAD/CAE community. Mr. Lou Mazzucchelli, president of Cadre Technology, characterizes the CASE interchange requirement (Goering, 1 September 1987):

An interchange standard isn't a luxury for CASE vendors; it's a necessity. A lot of the major systems we develop are built by prime contractors working with subcontractors. There's no way to ensure that those people will all have the same tools, yet they all have to work together on large government programs.

Cadre has developed a prototype interchange between Excelerator and Teamwork, by using a semantic extension to the electronic community's EDIF interim standard. Cadre has proposed that EDIF be extended to provide an interchange standard for CASE graphic and semantic output (Goering, 1 September 1987).

Such a standard could provide the catalyst for integration of CASE and CAE. This integration would allow design automation to be applied at a higher level of abstraction, such as for system description and partitioning. System analysis and simulation could then be conducted with joint consideration of hardware and software (Goering, 1 September 1987).

Future Developments

The next developments for CASE will include (Meyer, March 1987; Coolidge, 31 August 1987)

a. A central data repository, accessible by a variety of tools

b. Expert systems to direct software reuse via centralized libraries

c. Automatic translation of specifications to application source code

AI Technology Trends

The momentum of AI technology development during this period of convergence indicates that relevant developments may emerge from this field. Technology integration into multiple application domains will aid the adoption of these developments. The following trends are probable for development of AI techniques in design (Mitchell and Mostow, 14 July 1987):

a. Increasing domain-specific systems

b. Multimodel, bottom-up design

c. Increasing emphasis on reusability versus generation

d. Explicit reasoning about design goals

e. Interfacing AI methods with algorithmic methods

As expert system development tools, or shells, become increasingly easier to use and integrate, these facilities will be made available directly to the end-user engineering organization (Schindler, 9 July 1987). Such facilities will enable the designer to emulate and execute his/her own decision processes. For example, Hughes Aircraft used the Nexpert Object shell to embed diagnostics rules into a printed circuit board tester. The rules utilized design data and readings from the tester to produce diagnostics or suggestions for further tests (Schindler, 14 May 1987).

This scenario can be extended to a design team in which a consensus of design decision logic could be applied.

SECTION 5

A DESIGN AND OPERATIONS KNOWLEDGE SUPPORT SYSTEM FOR SPACE STATION FREEDOM

Now that design and operational knowledge valuable enough to require in computer form has been defined, the support system which contains this knowledge has been described, and the state of technology to implement such a support system has been assessed, the need and application to SSF can be addressed.

SSF is an engineered system which is large, complex, and distributed with operational requirements to support remote operation, real-time, long-term, nonstop mission and safety critical functions. This makes it an ideal candidate for using a DOKSS. In fact, such a system is under development.

A key conclusion drawn from the status of current operating design knowledge support systems and the technology assessment of section 4 is that these systems are very useful, cost-effective, and productive systems which can be built using commercial software and hardware almost entirely.

This section presents user scenarios, ground rules and guidelines, a concept for a DOKSS for SSF, and outlines possible roles for TMIS, the SSE and the Space Station Information System (SSIS), and other elements. This concept provides the context for a conceptual design for the JSC/WP-2 design knowledge support system.

After presenting this JSC/WP-2 conceptual design, the initial portion of this system, being implemented in 1989, can be described. This initial portion embodies and demonstrates some of the characteristics as a tool to obtain detailed user requirements for a complete system. Included is a brief discussion of the initial operational capture of design rationale at JSC and MDSSC in 1989 and the storing and use of this design knowledge. This section concludes with a discussion of additions of remaining portions and potential improvements to the JSC/WP-2 DOKSS.

USER SCENARIOS

The following scenarios are examples of the potential utility of a DOKSS for addressing system engineering issues for SSF. Such scenarios might occur from the time of initial design up to 15, 20, 25 and more years after the Phase I SSF is operational.

Scenario 1: A design engineer needs to recover the reasoning behind the shape of the cupola window in order to make design decisions on reconfiguration of the resource node of SSF.

Consider a designer of the resource node of the SSF sitting at a workstation which is "connected" to the DOKSS for SSF. With a graphic of SSF displayed, the designer would point with the mouse arrow to the resource node containing the desired cupola and click the mouse. The resource node will then be

enlarged and displayed. In a similar manner the designer could zoom in on the cupola. At this time the designer could ask for the DOKSS options available to him/her. These options could include

a. Display of CAD drawings of the cupola

b. Display of drawings and results from trade studies

c. Display of requirements and specifications from documents (text)

The design rationale behind the shape of the cupola window might be determined through selecting any or all of the above options.

For example, trade studies results could be linked to other text which discusses the design decision process from which the current configuration of the cupola window is reached and then its rationale. The ease and speed of obtaining this rationale makes for better quality work and greater productivity by this designer. The display presents an example of the type of information available and the structure of that information for the cupola structure object. This object shows a link to a section in the SSF program requirements document which gives the requirements for the cupola.

Scenario 2: An assessment is needed of the impacts if the weight of the standard data processor (SDP) were increased 1 kg.

In this scenario, the user might be a project office staff person who is asked to get this information to support a management decision about a proposed change in memory size and radiation protection. He/she could quickly zoom in and then select the attributes of the SDP which make up that object. The user could then change the weight entry just for the purposes of this "What if?" calculation. This weight change can then be allowed to ripple through all the parent objects which contain the SDP object. This could automatically cause a recalculation of all the weights and centers-of-gravity of the associated objects again just for the purposes of this question and not to make a version change to the design. The user could be informed that this increase in weight might now exceed a particular Shuttle manifest.

Scenario 3: During the design process for a piece of hardware, one promising option may increase the projected power consumption of that hardware. The designer needs to know what effect this increased power requirement may have on the overall system.

The "What if?" question may cause a simple simulation model program to be executed. Several results would be possible. The simulation may indicate that it would be necessary to decrease the power available to some other piece of equipment. This may mean that the power available now falls below the minimum required for that equipment. The user may be notified that the power carrying capability of a cable could be exceeded, indicating that unless the cable can be upgraded, the increase in power demand cannot be supported.

Scenario 4: An operations console operator needs to develop a modified operations procedure for a new distributed system configuration. The modified procedure is dependent on having an understanding of

the various modes of operation and hence on the design of every portion of the spacecraft that has first order involvement in any step in the operations procedure.

This scenario shows how operational and operations personnel may benefit by use of a DOKSS. Consider the operations procedure developer sitting at a workstation which is part of the DOKSS for SSF. With a graphic drawing of SSF displayed, the developer would, in turn, point with the mouse arrow and zoom into each element of SSF and click the mouse to ask for the DOKSS options available. These options could include

a. Display of the descriptions of the operational modes, conditions necessary to use each mode, and the effects achieved with each mode for the current system configuration

b. Display of the same descriptions for the planned configuration

c. FMEA's for each configuration

d. The current version of the operations procedure

The current procedure could be stepped through for each mode of the planned configuration to check if the conditions necessary to use the mode are met or violated (or uncertain) and to check the consequences of the likely failure modes. This would aid in ascertaining any modifications to the procedure required to accommodate the new system configuration. By splicing any needed modifications into a copy of the current procedure, the modified procedure is achieved and stored for future reference with links to the proper configuration. Clearly, easy and quick access to such accurate and authoritative design and operations knowledge will affect the quality of the modified procedure and the productivity of the developer.

GROUND RULES AND GUIDELINES

Definition of the ground rules and guidelines is an important step in establishing a DOKSS. A ground rule or guideline is needed for the DOKSS at the system level and for each necessary element. System level ground rules are

a. Meet the users' needs cost-effectively

b. Augment, don't invent

c. Stay in the mainstream

d. Use existing resources, facilities, tools, etc.

e. Use integrated design, not piecemeal

f. Use information asset management philosophy

g. Evolve to meet user needs better

h. Try to have compatible systems for SSF and NSTS

Meeting the users' needs is the reason for the DOKSS and its cost is minor compared to the long-term benefits. The next ground rule is "augment, don't invent." Namely, this is an opportunity to use what industry has already done and is in place. NASA is not leading in implementing this technology. Inventing involves risk and time which is not appropriate here. Certainly NASA should support research and advanced development in this technology, but not as a part of this DOKSS project. Another related ground rule is "stay in the mainstream." Namely, the aerospace community, including DOD, has a certain direction and momentum in specifying and implementing these systems which NASA should recognize and leverage. "Use existing resources, tools, working groups, etc.," is part of this ground rule. In specifying these ground rules and guidelines, NASA, being the customer, is asserting its role as decision maker. The PSC can help in this activity. Community standardization guidelines can be worked with the TMIS, SSE, and PSC organizations.

OBTAINING AND INTEGRATING USER REQUIREMENTS

Inventory of the existing and planned users and suppliers of design knowledge is best accomplished by NASA and contractor work package personnel and supporting PSC personnel. However, a demonstration of a preliminary prototype which can show in detail how this system offers benefits is the best method of communication. Once the concept has thus been shown to be a reality, user modifications and requirements will be more likely.

OBJECT-ORIENTED MODELING CAPABILITY

An important and powerful concept for a DOKSS is the object-oriented paradigm for modeling. The object-oriented paradigm is simply: all real world entities are "objects" which "may" act independently of all other objects. This supports a high degree of modularity. An object contains structural and procedural descriptions of the entity which it represents. These descriptions are not visible outside of the object. An object is a "black box" which responds to "messages" from the outside world sent by users and other objects. When an object receives a message, it performs an appropriate action by executing a procedure ("method") which may display its contents, change its internal structure, cause messages to be sent to other objects or perform any combination of the above. An object may be an "instance" object (which represents a single entity within the environment) or a "class" object (which represents a collection of similar instances). Objects are arranged in an "is-a" hierarchy such that the most general class object is at the root. As this hierarchy is traversed towards its leaves, an object's properties become more and more specialized. Instance objects appear at the leaves of the hierarchy. Attributes (methods and structure) are inherited through this hierarchy from ancestors (objects closer to the root) by their more specialized descendants. A major advantage of an inheritance hierarchy is that the common data do not have to be stored with all objects, which saves space and makes updating shared data and methods simple, efficient, and modular. New objects are often defined by copying the generic description of a class object and customizing it to satisfy the current need.

COMPATIBILITY WITH DOKSS FOR THE NATIONAL SPACE TRANSPORTATION SYSTEM

Both the Space Shuttle and SSF have life cycles longer than a decade. The interfaces between them are more complex than simply that SSF is passive cargo since the assembly and checkout on-orbit requires, for example, Space Shuttle remote manipulator modification and use. There is overlap in the NASA line organizations which support both programs. Although the NSTS program has been using various design and operations data for some time, it could still benefit from the retrofit of a DOKSS. Since part of the intent of the design of a DOKSS is to make the implementation details of the DOKSS hidden from the user, it should be possible to design the NSTS and SSF design knowledge support systems to be compatible, i.e., usable through the same user interface generally, even though the precise services may be different or the data organized differently and supported with different DDBMS's and the like.

Initial concept discussions with the Space Shuttle Orbiter prime contractor, Rockwell International, indicate that many of the implementation details could also be similar and that the same user interface may be possible for the Shuttle Orbiter and the JSC/WP-2 portion of the SSF. This is because Rockwell has independently reached and implemented many similar approaches to those in this technical memorandum due to its involvement in DOD programs such as IDS, CALS, and PDES.

A CONCEPT FOR A DOKSS FOR SPACE STATION FREEDOM

Section 3 provided a definition of the characteristics of a DOKSS for any engineered system. For SSF, there are seven major developmental organizations: WP-1, WP-2, WP-3, WP-4, and the Canadian, European, and Japanese hardware and software portions of the program comprising "the engineered system." Setting aside the non-U.S. portions, (although notable design knowledge capture efforts are underway at least in Europe), the DOKSS would encompass projects at four (at least) NASA Centers and their prime and subcontractors, plus the Kennedy Space Center and the SSF Program Office (Level II) in Reston, Virginia.

Several existing efforts are critical to incorporate into the concept and to supply sufficient resources to a DOKSS. The TMIS (SSP 30519, Rev. A), SSE, SSIS, Systems Engineering Simulator (SES), and Multisystem Integration Facility (MSIF) are program or total-station-level activities which are necessary elements of the concept (assuming they are adequately funded).

Figure 5 shows a concept diagram of a possible future SSF program DOKSS. The user portions of the diagram represent multiple user workstations on local-area networks and TMIS wide-area networks located at NASA Level II in Reston, various NASA Centers, work package prime contractors and subcontractors with access to CAD/CAE data, engineering data bases, and designer's knowledge text, graphics, and models. The SSE is a key portion of the Level II part of the concept for software design, verification, and maintenance purposes. The SSE design has many elements of a DOKSS and has recently adopted CALS standards such as the standard generalized mark-up language (SGML). In this concept, data, information, and knowledge accuracy is maintained at its single source. Access is supported through a variety of existing and planned hardware and operating system combinations enabled by open network communications implemented by many vendors. The integrated system concept is enabled by object-oriented user interfaces and data bases which, in turn, interact with the

rest of the system. Other NASA Centers and their contractors will access the JSC/WP-2 portion through a gateway to the program-wide TMIS network.

ROLES FOR TMIS, SSE, SSIS, SES, AND OTHER PROGRAM SYSTEMS

The DOKSS concept assumes the various SSF program systems such as TMIS, SSE, SSIS, SES, and MSIF essentially fulfill the functions envisioned for them separately and, without usurping their responsibilities, integrate them cooperatively into the concept of the DOKSS. Various SSF program test beds are possible important sources of design knowledge.

The SSE is a collection of software, hardware, and procedures which will allow the efficient creation, management, and integration of ground and flight software developed for SSF. The fact that this software will be developed by multiple geographically distributed contractors and that it must have an expected useful lifetime of 30 or more years implies that special capabilities must be provided to record and manage the products and artifacts of the software design effort as it progresses. This record of the software design process must be available both to provide visibility and control into the software design process to make possible the formidable task of software integration as well as to ensure that modifications and technology growth over the long lifetime of SSF can be accommodated.

Design knowledge capture is accomplished in the SSE by providing a project object base which is a central repository for all the elements of the design as they are created. This project object base will have an object-oriented structure with many possible types of objects, relationships between objects, and attributes characterizing objects. At each stage in the software design and development process, a software developer will interact with the project object base via process management software. The process management software controls all objects in the project object base and determines which applications may access a given object. For instance, an Ada compiler may access a completed Ada source code, but not an incomplete source code or, for instance, a requirements document. This capability allows the system to ensure that required steps in the life cycle model have been completed in logical order. The process manager is also careful to record which developer is working on a particular object and will not allow anyone else to access that object while it is in work. Finally, the process manager automates the recording and identification of new versions of design and development objects so that configuration control and traceability may be maintained.

The combination of the centralized project object base and the process manager provide for the automatic capture and documentation of all the major design products in the software life cycle including requirements documents, design artifacts (Buhr-Booch diagrams, data element dictionaries, etc.), preliminary design language code, source code, object code, and documentation. The concept, while extensive and essential to the SSF software development effort, does not extend to explicitly recording the reasons for particular design decisions unless they are specifically introduced into notes or other object attributes. As the SSE develops, the design knowledge capture process itself is designed so that it may evolve to provide for easier and more complete capture of all the design decisions and products which are a part of the software development process.

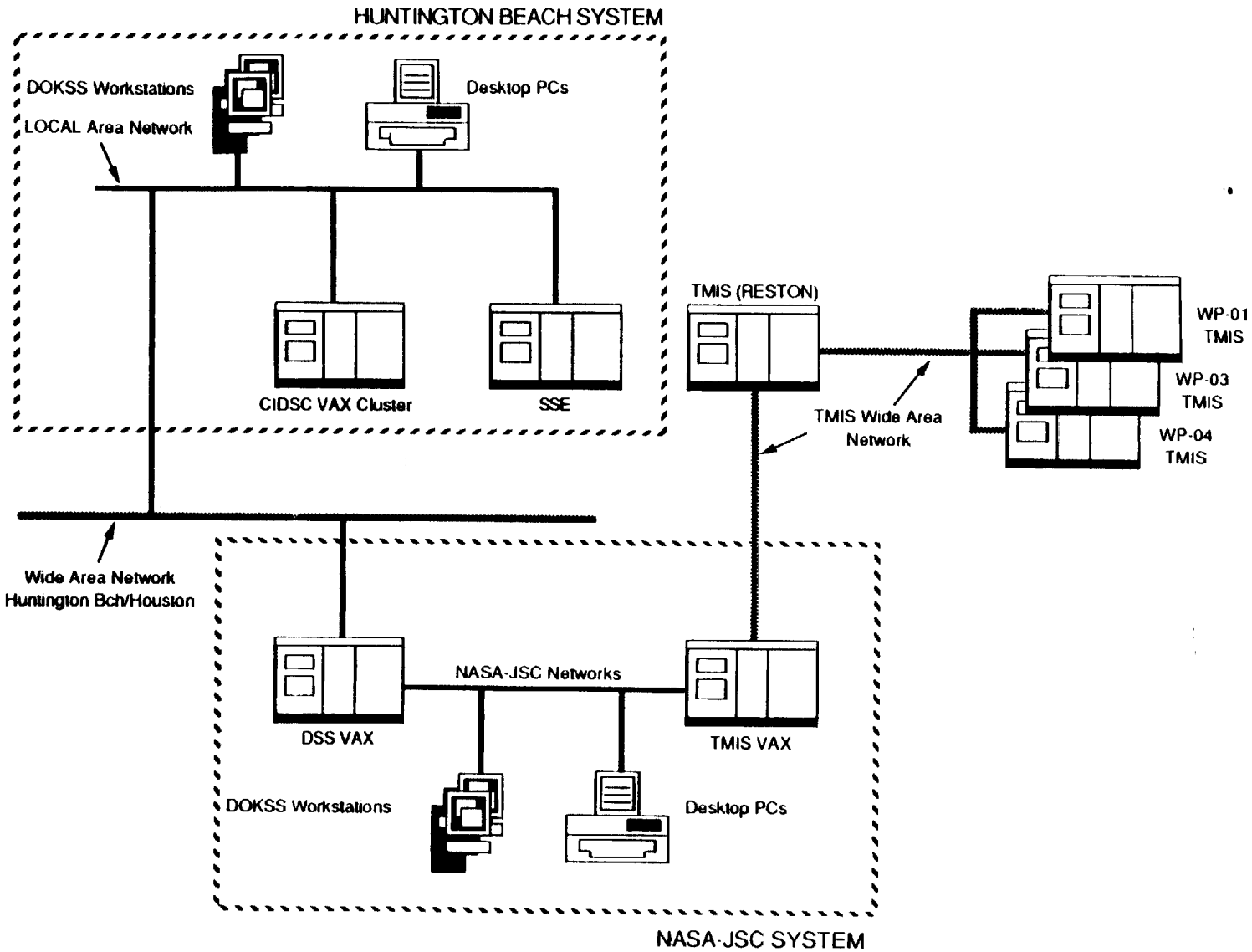

Figure 5.- Future program-wide DOKSS architecture.

CONCEPTUAL DESIGN OF DOKSS FOR JSC/WP-2

Figure 6 shows the conceptual design of a DOKSS supporting both the JSC portion of SSF activities and the WP-2 portion of MDSSC and its subcontractors. Figure 7 shows the detail of the MDSSC portion at Huntington Beach, California. In the context of the SSF program DOKSS shown in figure 5, these concept diagrams provide more detail and some implementation choices. A detailed set of selection criteria for software based on initial DOKSS requirements was developed, and three software options were studied before selection. The software selection was a major driver in hardware selection. These studies are documented in MDSSC's DOKSS Plan (SY-08.1, November 1988). A major difference between the JSC and WP-2 systems is that the WP-2 system does not extend beyond the current contract and does not cover the operations phase of the SSF program, as the JSC systems must. The rationale for the development choices for this conceptual design are as follows:

a. It is compatible with our view of the SSF program design concept that data accuracy is to be maintained at its source.

b. It takes advantage of JSC and MDSSC existing and planned computer systems (EDB, CAD VAX, Apollo, Mac II, etc.) and access is supported through a variety of hardware and operating system combinations.

c. Local networking at JSC and MDSSC is enabled by a file server on a SUN 3 and open network communications with SUN's Network Services Architecture and Network File System implemented by many vendors.

d. User friendly interfaces of icons, windows, menus, text, and graphics and object-oriented DBMS software needs are met using COTS software: Analyst, Smalltalk, and Gemstone.

e. The JSC network and MDSSC network communicate through TMIS. JSC/WP-2 also communicates with other SSF program elements through TMIS.

f. The development of the DOKSS user interface and demonstration must start early in 1989 and SUN 3 is currently the only platform simultaneously supported by Analyst, Smalltalk, and Gemstone.

Gemstone is a multiuser object-oriented data base management system that combines the functionality and security of a mainframe DBMS and the power of the object-oriented paradigm. Virtually any kind of data structures (text, numbers, drawings, maps, computer programs, expert system rules, etc...) can be created and stored. Users define and manage classes of objects and associated operation handles or methods. Gemstone is developed based on the Smalltalk programming language. Development work in Gemstone may be done in its front-end programming language called OPAL (or TOPAZ for VAX/VMS terminals). Comprehensive applications may also be built in Smalltalk or C and interface with the Gemstone data base.

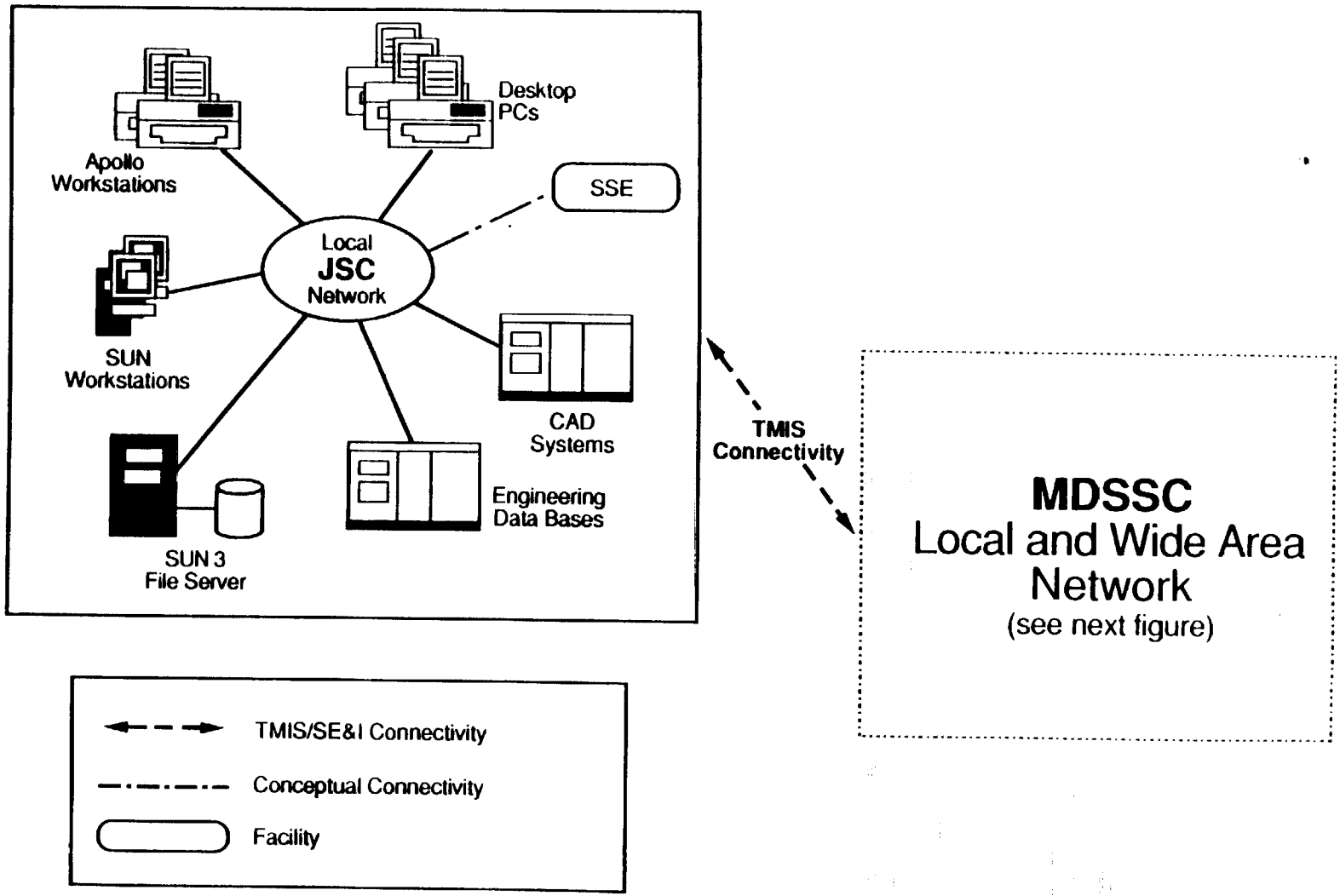

Figure 6.- JSC and WP-2 DOKSS network concept configuration.

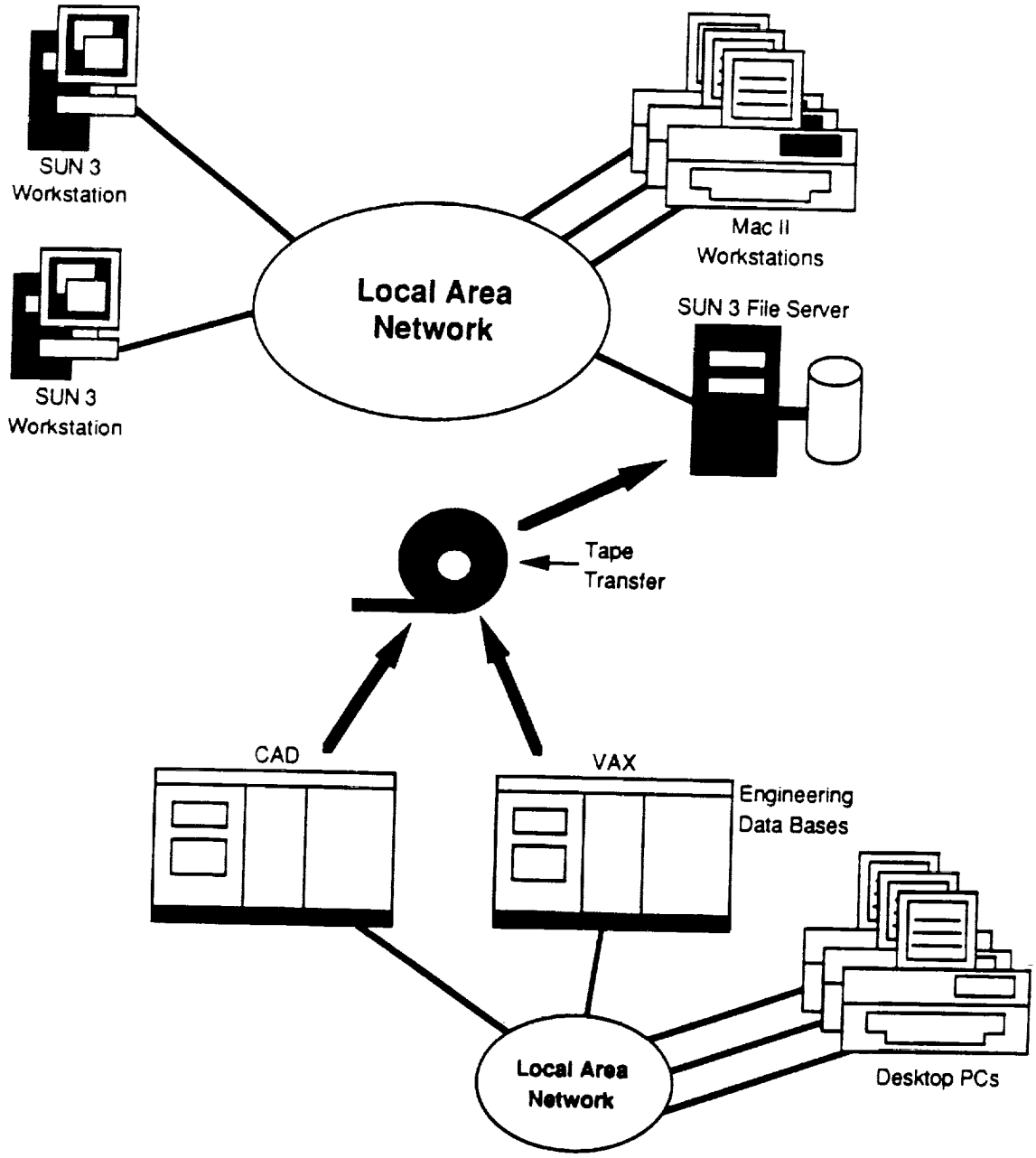

Figure 7.- DOKSS hardware architecture.

Integration with Engineering Data Bases

An important part of the initial architecture is to provide for the integration of the DOKSS with existing and planned engineering data bases. Data base integration will be implemented in three phases. During the initial implementation period, no physical connection will exist between the DOKSS and the engineering data bases. Data will be transferred by magnetic media.

As improved network capabilities are brought online, the DOKSS will be integrated with the engineering data bases using a standard 2-schema approach as shown in figure 8a. In this approach, the DOKSS is a single view into the engineering data bases.

When a full 3-schema approach is implemented as shown in figure 8b, the DOKSS workstations become multiple views in the external schema, while the DOKSS knowledge server is represented by a single internal schema that maps the physical structure of the captured design knowledge contained in the DOKSS. This approach will provide transparent access to captured knowledge from any networked location.

Functional Decomposition

An initial functional decomposition from the implementation point of view has been developed. Figure 9 shows the DOKSS top-level functions. The system architecture, centered around an object management system, identifies both user interfaces and interfaces to other computer systems.

Additions and Improvements

The DOKSS systems at JSC and MDSSC are intended to be completed in a phased development recognizing yearly resource availability. This will enable not only additions of the remaining portions of the DOKSS, but also support of additional aspects of the JSC and WP-2 portions of SSF. Potential improvements to the JSC/WP-2 design knowledge support system are intended to be added in an evolutionary manner in response to user needs.

INITIAL OPERATIONAL CAPTURE

During 1989, JSC and MDSSC design knowledge support teams are capturing high-level design decision rationale from sources such as trade studies, design reviews, and various board and panel decisions. At JSC, project office boards and design reviews now include specifically structured guidelines for presenters and compilers of minutes which address the rationale for decisions for capture. The same is true at MDSSC for engineering review boards and the like.

Initially, this set of rationale is not available online. It will be available once the initial DOKSS elements are developed. Initial use of this set of rationale will be in 1990.

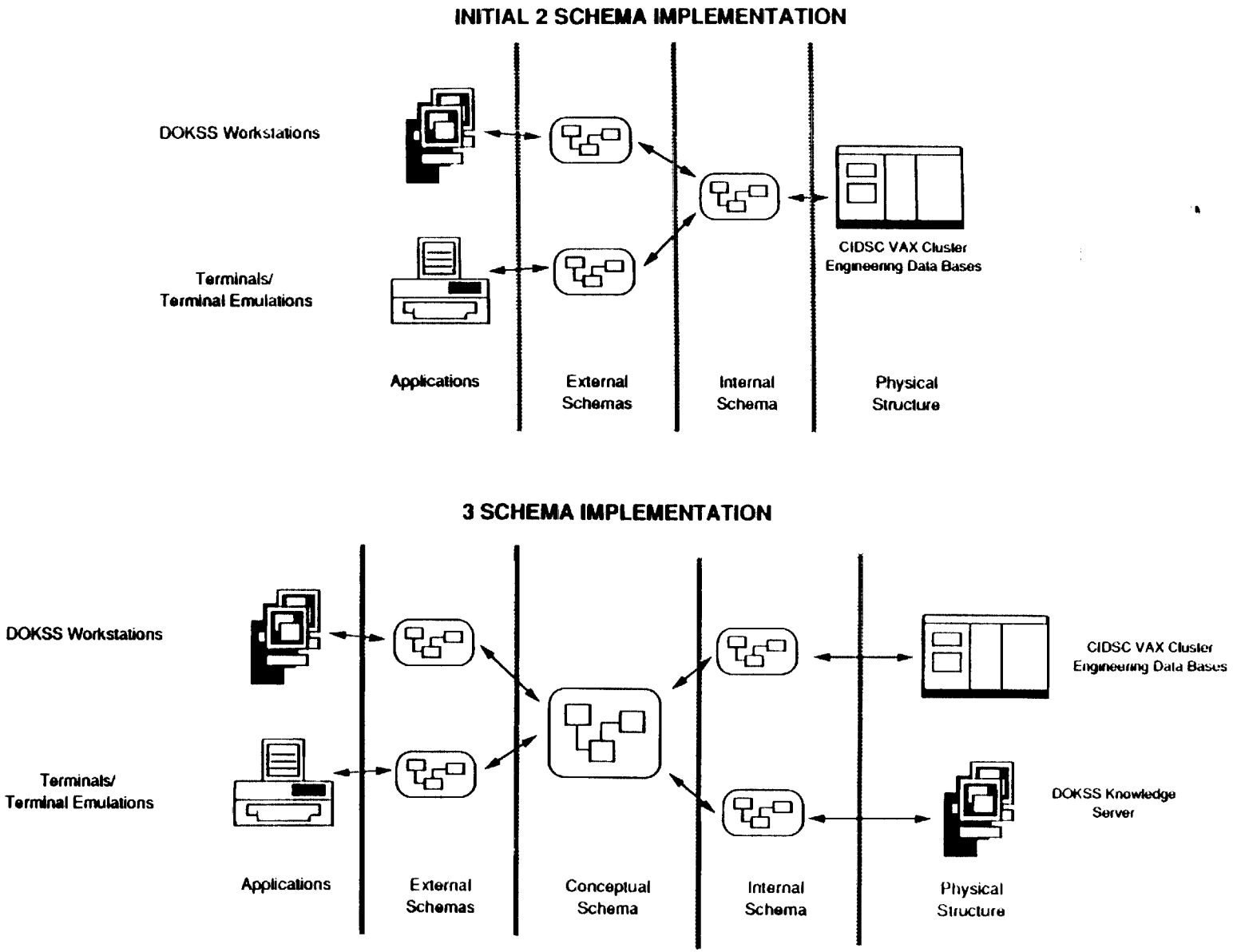

Figure 8.- DOKSS integration with engineering data bases.

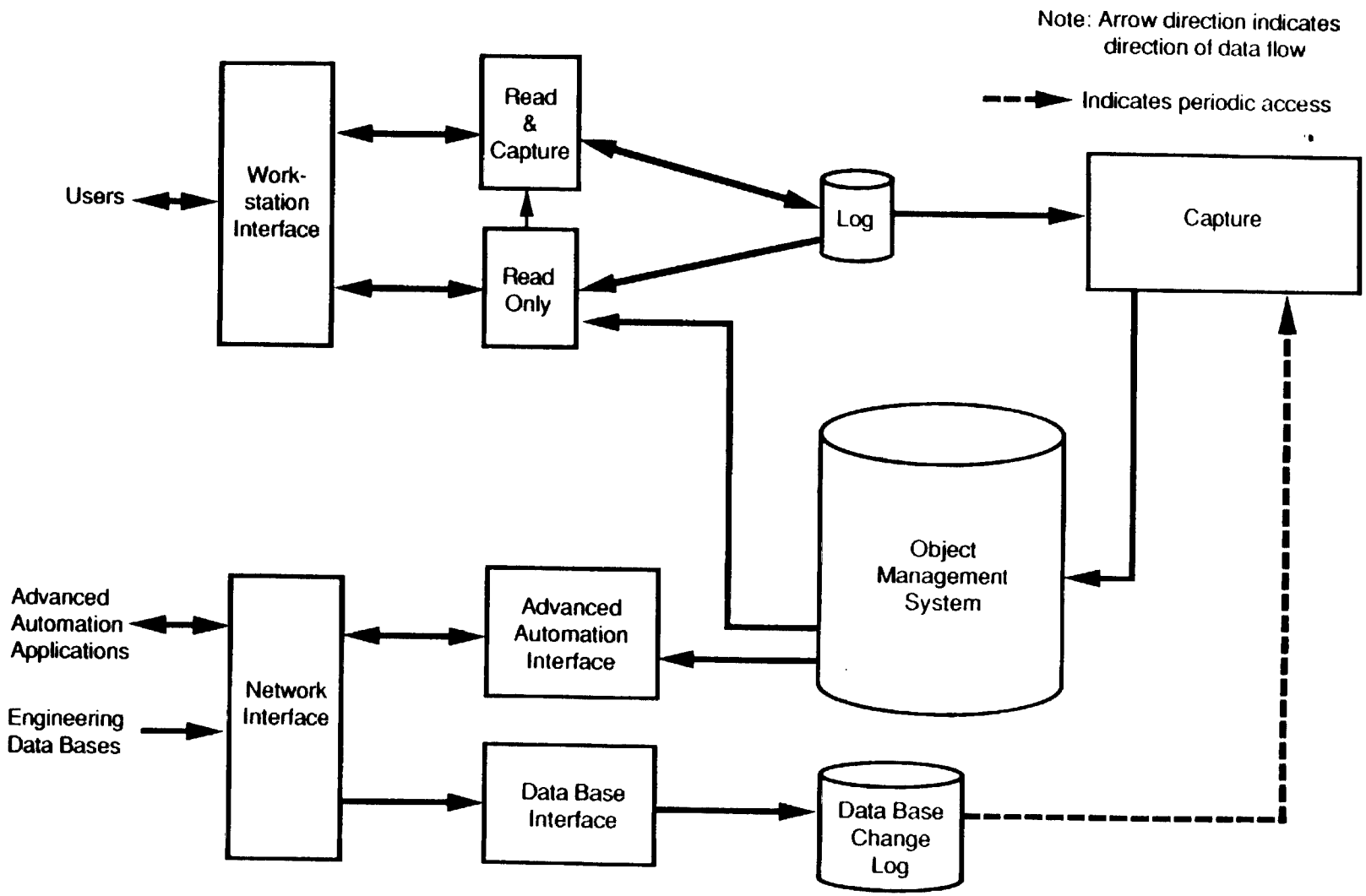

Figure 9.- DOKSS top level architecture.

SECTION 6
CONCLUDING REMARKS

We have presented some of the benefits, such as quality and cost savings, to be obtained from using accurate and authoritative design and operations knowledge about an engineered system. The general requirements for readily accessible design knowledge imply the need for a DOKSS as a convenient-to-use information system delivered and operated with the engineered system. After briefly looking at some example systems, we presented a detailed review of the underlying technologies. This review supports the conclusion that integrating existing COTS products will currently support a DOKSS for the benefit of SSF. Two key aspects are the COTS open systems architecture products and the object-oriented user interfaces and data bases. As with any open system, the DOKSS will provide additional capabilities as the underlying technology is improved. Our concept allows for AI approaches to the DOKSS (such as knowledge engineering and knowledge representation in knowledge bases) to contribute to the support for SSF knowledge-based systems.

A concept of a DOKSS for SSF was given in which possible roles of TMIS, SSE, SSIS, SES, and other elements were outlined. User scenarios are included as examples of the benefits for using a DOKSS on SSF engineering issues and operations.

For those who will use either the SSF DOKSS or other such systems in the future, we have attempted to explain why these systems are useful and to elucidate the basic and essential characteristics and features of such systems with sufficient detail for clarity that sufficient understanding of this new capability is achieved. Because of the importance and value of design and operations knowledge, organizational values and discipline in entering this knowledge are as important as any of the technology.

The significance of the start that has been made in 1989 at JSC and MDSSC on two important activities -- the routine operational capture of key parts of the rationale for design decisions on the JSC/WP-2 portion of SSF (along with CAD drawings and engineering data bases) and the implementation of a DOKSS for the JSC/WP-2 portion of SSF -- is that quality and cost savings benefits are expected to begin accruing in proportion to the degree of use we can generate by meeting user needs and the degree of development we can achieve to support those needs. However, until we are able to reliably replace portions of the paper document system, such benefits will be much less than they might be.

SECTION 7
REFERENCES

Andrews, Timothy and Craig Harris (October 1987), "Combining Language and Data Base Advances in an Object-Oriented Development Environment," OOPSLA '87 Proceedings.

Baker, Mary et al. (May 1986), "Space Station Multidisciplinary Analysis Capability--IDEAS-Squared," Proceedings of the AIAA Conference on Structures, Structural Dynamics, and Materials.

Boas, Ghica van Emde and Peter van Emde Boas (January 1986), "Storing and Evaluating Horn-Clause Rules in a Relational Data Base," IBM Journal of Research and Development.

Boff, Kenneth R. (1987), "Integrated Perceptual Information for Designers," Armstrong Aeromedical Research Laboratory.

Booch, Grady (1986), "Object-Oriented Programming," IEEE Transactions on Software Engineering.

Bunn, Wiley C. (7 February 1989), "Quality, A Recognized Crisis or Where is SR&QA and Where is it Going," Marshall Space Flight Center.

CALS Core Requirements (1987), Phase I.0, Coordination Draft, Department of Defense.

CAM-I News Alert (September 1987), Computer Aided Manufacturing-International.

"CASE Accord Signed by Pansophic, Cadre" (14 September 1987), MIS Week.

Chang, Shi-Kuo (January 1987), "Visual Languages: A Tutorial and Survey," IEEE Software.

Clancy, William J. (Summer 1989), "Viewing Knowledge Bases as Qualitative Models," IEEE Expert, p.9.

Clark, Stephen and Sanford Ressler (1987), The Geometry Modeling System User's Guide, NBSIR 87-3508, National Bureau of Standards.

Computer Aided Manufacturing-International (1985), Requirements for Support of Form Features in a Solid Modeling System, R-85-ASPP-01.

"Concept CAD on a Desktop" (May 1987), Computer Aided Engineering.

Coolidge, Robert C. (31 August 1987), "CASE Environment Window to Future," MIS Week.

Cox, Brad J. (1986), Object-Oriented Programming, Reading, Massachusetts: Addison-Wesley.

Cummings, Steve (28 July 1987), "Graphics Chip Advances Expand Software Possibilities," PC Week.

Datapro Research Corporation (June 1987), "Users Rate Their LAN's," Data Communications.

Derfler, Frank J. (9 December 1986), "LAN's and Beyond," PC Magazine.

Digitalk, Inc. (1986), Smalltalk/V Tutorial and Programming Handbook, Los Angeles, California.

Draft Prospectus for an Industry/Government PDES Cooperative Project (1987).

Electronic Industries Association (1987), Electronic Design Interchange Format, Version 2.0.0.

Esplin, Kathryn (7 September 1987), "Cognition Finishing Work on Expert System for Engineers," Digital News.

Fenves, Steven J. et al. (1987), Introduction to SASE: Standards Analysis, Synthesis, and Expression, NBSIR 87-3513, National Bureau of Standards.

Flynn, Laurie (1 June 1987), "Computer Aided Design Use on LAN's Increasing," InfoWorld.

Foster, Dennis L. (1985), The Practical Guide to the IBM PC/AT, Reading, Massachusetts: Addison-Wesley.

"Front-End Mechanical Engineering Solutions: The Next Wave" (31 May 1987), CAD/CIM Alert.

Gantz, John (1 June 1987), "Sun Begins to Shine on Technical PC-LAN Market," InfoWorld.

Gantz, John (7 September 1987), "Technical Workstations Begin to Clash with PC's," InfoWorld.

Goering, Richard (1 September 1987), "Interchange Standard for CASE Gathers Support," Computer Design.

Goodstein, David Henry (July 1987), "Publishing Standards: Everyone's An Expert," Computer Aided Engineering.

Greenstein, Irwin (20 April 1987), "Is AI/DBMS Meld Big Catalyst for Software Shift," MIS Week.

Henderson, Mark R. (1984), Extraction of Feature Information from Three Dimensional CAD Data, PhD. Thesis, Purdue University.

Index Technology Corporation (1986), Excelerator/RTS User Guide, Cambridge, Massachusetts.

Industrial Automation Standards Workshop (1987), CAM-I.

"Integrated Design Support Prospectus" (1987), IDS Program Office.

Kaehler, Ted and Dave Patterson (August 1986), "A Small Taste of Smalltalk," Byte.

Kennedy, Don (September 1988), "AI Tool Allows Expert Systems On Industry-Standard Hardware," Digital News.

Ketabchi, Mohammad A. (April 1987), "A Matrix of CAD Applications Requirements/DBMS Capabilities," Proceedings: CAD/CAM Data Bases '87 Management Roundtable.

"Large Organizations Turning to PC CAD" (May 1987), Computer Aided Engineering.

Latombe, Jean-Claude and Mark S. Dunn (1984), XPS-E: An Expert System for Process Planning, Proceedings of CAM-I'S 13TH Annual Meeting.

Laub, Leonard (May 1986), "The Evolution of Mass Storage," Byte.

Linn, Joseph L. and Robert I. Winner, editors (1986), The Department of Defense Requirements For Engineering Information Systems (Volume 1: Operational Concepts), The Institute for Defense Analysis.

Lockwood, Russ (June 1987), "Reading Pictures Into Your Computer," Personal Computing.

Long Range Plan for the IGES Organization (1986).

Lopez, Leonard A. et al. (1985), Mapping Principles for the Standards Interface for Computer Aided Design, NBSIR 85-3115, National Bureau of Standards.

Luby, Steven C. et al. (November 1985), "Creating and Using a Features Data Base," Computers in Mechanical Engineering.

MacLean, Allan et al. (May 1989), "Design Rationale: The Argument Behind the Artifact," CHI '89 Proceedings ACM, p. 247.

Maier, David et al. (September 1986), "Development of an Object-Oriented DBMS," OOPSLA '86 Proceedings.

McCaskey, John (19 March 1987), "Object-Oriented Data Base Keeps the House in Order," Electronic Design.

Meyer, Bertrand (March 1987), "Reusability: The Case for Object-Oriented Design," IEEE Software.

Mitchell, Tom and Jack Mostow (14 July 1987), "Artificial Intelligence and Design," AAAI-87 Tutorial TP-2.

Mladejovsky, Michael (19 February 1987), "Hierarchical-Level CAE System Features Snap and Simplicity," Electronic Design.

NASA Space Station Program Office (September 1988), Process Requirements for Design Knowledge Capture, SSP-30471, Rev A.

NASA Space Station Program Office (September 1988), TMIS Interface Description Document (TIDD), SSP-30519, Rev. A.

National Bureau of Standards (1986a), IGES Version 3.0: Initial Graphics Exchange Specification

National Bureau of Standards (1986b), A Reporting of the PDES Initiation Activities.

National Research Council, Committee on CAD/CAM Interface 1984), "Computer Integration of Engineering Design and Production, A National Opportunity," National Academy Press, Washington, D.C.

Orr, Ken (August 1987), "Planning for the Software Factory," Infosystems.

Pascoe, Geoffrey (August 1986), "Elements of Object-Oriented Programming," Byte.

Patch, Kimberly (August 1988), "Program To Let Users Integrate AI Infor in Relatinal Data Base," Digital Review.

Peterson, Robert W. (March 1987), "Object-Oriented Data Base Design," AI Expert.

Prince, David M. (1971), Interactive Graphics for Computer-Aided Design, Reading, Massachusetts: Addison-Wesley.

Rawsthorne, Dan (May 1987), "Synergist, A Fault Diagnosis and Schematic Capture System," Intelligent Applications Ltd., Columbia, Maryland.

Rehak, Daniel R. et al. (1984), "Architecture of an Integrated Knowledge-Based Environment for Structural Engineering Applications," Working Conference on Knowledge Engineering in Computer Aided Design, Amsterdam, The Netherlands: Elsevier Science Publishing Company, Inc.

Ressler, Sanford (1987), Using the AMRF Part Model Report, NBSIR 87-3531, National Bureau of Standards.

Rich and Waters, eds. (1986), Artificial Intelligence and Software Engineering, Morgan Kaufmann.

Rinaldi, Damian (August 1987), "A Question of Timing," Software News.

Rosenfeld, Lawrence W. and Avrum P. Belzer (1985), "Breaking Through the Complexity Barrier--ICAD," Proceedings of Autofact '85.

Ross, Ronald (1987), Entity Modeling: Techniques and Application, Boston, Massachusetts: Data Base Research Group.

Schindler, Max (14 May 1987), "Expert System Fits with Conventional Software," Electronic Design.

Schindler, Max (9 July 1987), "Designers Build Smarts into Their CAE Tools with Expert System Shells," Electronic Design.

Stamps, David (1 July 1987), "CASE: Cranking Out Productivity," Datamation.

Stanton, Tom (13 October 1987), "Scanners Take Off," PC Magazine.

Stanton, Tom et al. (30 September 1986), "Page-to-Disk Technology," PC Magazine.

Stefik, Mark and Daniel Bobrow (1986), "Object-Oriented Programming: Themes and Variations," AI Magazine.

Steinke, George C. and Martin D. Schussel (November 1985), "Engineering by the Book and Online," Mechanical Engineering.

Stone, Paula S. (14 September 1987), "Many Users Choosing Xenix Over MS-DOS," InfoWorld.

Stover, Richard N. (1984), An Analysis of CAD/CAM Applications, Englewood Cliffs, New Jersey: Prentice-Hall.

Wechsler, Donald B. (August 1987), "Product Life Cycle Knowledge Modeling," OOPSLA '87 Workship for Methodologies and Object-Oriented Programming.

Wechsler, Donald B. and K. R. Crouse (1986), An Approach to Design Knowledge Capture for the Space Station, SPIE, Vol. 729, Space Station Automation.

Weston, Charles D. and George A. Stewart (February 1987), "Workstations," Byte.

Williams, Robert, John Quintanilla, and Kevin Shaum, "Commercial DDBMS Technology Assessment Report," ASD, NASA Johnson Space Center, September 28, 1988.

Zengerle, Patricia (18 May 1987), "CAD/CAM Outlays Head Down," MIS Week.

Zoellick, Bill (May 1986), "CD-ROM Software Development," Byte.

APPENDIX: GLOSSARY

Attribute:
Property or characteristic of an entity.

Data:
Discrete recorded facts about phenomena in the data base enterprise.

Data Model:
A model which provides a methodology for capturing how data about (some part of) the real world are related.

Data Modeling:
The process of using a data model to construct a model of a data base enterprise.

Data Base:
A collection of data and "associations" or "relationships"among those data.

Data Base Management System:
A system which provides facilities for defining and retrieving stored information and also provides a basic protection mechanism for users and data.

Design Knowledge:
There are three kinds of design knowledge:

a. Physical design knowledge -- descriptions of the physical characteristics and system properties of a system.

b. Design decision rationale -- provides substantiating information concerning design decisions.

c. Functional/behavioral knowledge -- describes the function and physical behavior of systems and their components.

Design Knowledge Capture:
The process of capturing, analyzing, and maintaining design knowledge in a systematic machine-interpretable form.

Design and Operations Knowledge Support System (DOKSS):
The DOKSS supports the design knowledge capture process and use of the captured knowledge throughout the development and life of the engineered system (Space Station program).

Designer's Knowledge:
The reasoning behind the design, construction, and operation of a product or system.

Distributed Data Base:
A data base kept in dispersed locations on a computer network. Access to different parts of the data is controlled by several different computers.

Engineering Data:
Data pertinent to the analysis, design, and construction of the system (Space Station Freedom).

Entity:
An identifiable concrete or abstract item about which information is recorded.

Host:
A computer which provides a (computing) resource for itself and other machines on a local-or wide-area network.

Information Asset Management Philosophy:
A philosophy which advocates treating information like any other capital asset with well structured controls for its acquisition and disposal.

Integrity:
More precisely, the act of maintaining integrity is a process for ensuring -- as much as possible --that the data in the data base are accurate and consistent at all times.

Network:
An interconnected collection of autonomous computers. Two computers are said to be interconnected if they can exchange data and/or share some computing resources with other.

Product:
A physical object created using the designer's knowledge.

Relation (informal data base related definition):
A table in which each "row" represents a "record" and each "column" represents an "attribute" whose value is taken from a predetermined domain. Note for a given table, the number of columns is fixed.

Relational Data Base:
A collection of time-varying tables, i.e., rows may be changed by insert, delete, and update operations.

Spiral Approach:
An iterative requirements/development/evaluation process for software system development.

Subsystem:
An identifiable constituent part of a system (Space StationFreedom) consisting of one or more system components.

System:
A decomposable (and possibly distributed) collection of tasks or physical objects which together fulfill a single function (or physical part of the Space Station Freedom).

System Component:
A nondecomposable unit of a system. It may be a piece of equipment, construction material, data, software, a service, or personnel.

REPORT DOCUMENTATION PAGE

1. Report No. NASA TM 102 156	2. Government Accession No.	3. Recipient's Catalog No

4. Title and Subtitle Considerations for a Design and Operations Knowledge Support System for Space Station Freedom	5. Report Date October 1989
	6. Performing Organization Code EF
7. Author(s) Jon D. Erickson, Kenneth H. Crouse, Donald B. Wechsler, Douglas R. Flaherty	8. Performing Organization Report No S-596
9. Performing Organization Name and Address Lyndon B. Johnson Space Center Houston, Texas 77058	10. Work Unit No
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	13. Type of Report and Period Covered Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes
Jon D. Erickson, Kenneth H. Crouse - Lyndon B. Johnson Space Center
Donald B. Wechsler - The MITRE Corporation
Douglas R. Flaherty - McDonnell Douglas Space Systems Company

16. Abstract

Engineering and operations of modern engineered systems depend critically upon detailed design and operations knowledge that is accurate and authoritative. A design and operations knowledge support system (DOKSS) is a modern computer-based information system providing knowledge about the creation, evolution, and growth of an engineered system. The purpose of a DOKSS is to provide convenient and effective access to this multifaceted information.

The complexity of Space Station Freedom's (SSF's) systems, elements, interfaces, and organizations makes convenient access to design knowledge especially important, when compared to simpler systems. The life cycle length, being 30 or more years, adds a new dimension to space operations, maintenance, and evolution.

This technical memorandum provides a review and discussion of design knowledge support systems to be delivered and operated as a critical part of the engineered system. A concept of a DOKSS for SSF is presented. This is followed by a detailed discussion of a DOKSS for the Lyndon B. Johnson Space Center and Work Package-2 portions of SSF.

17. Key Words (Suggested by Author(s)) Knowledge representation Information systems Computer systems design Space stations	18. Distribution Statement Unclassified - Unlimited Subject category: 60

19. Security Classification (of this report)	20. Security Classification (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	79	

For sale by the National Technical Information Service, Springfield, VA 22161-2171