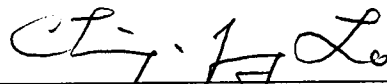


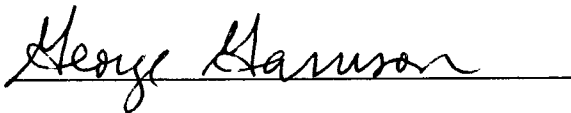
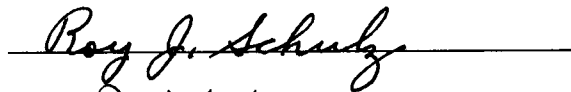
To the Graduate Council:

I am submitting herewith a thesis written by William Andes Hoyt Jr. entitled "Knowledge Engineering Methodology Investigations for Intelligent Hypertext Manuals With A Prototype Aerospace Implementation." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Aerospace Engineering.



Ching F. Lo, Major Professor

We have read this thesis
and recommend its acceptance:



Accepted for the Council:



Vice Provost
and Dean of the Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirement for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my permission.

Signature William Andrew Hoff Jr
Date _____

KNOWLEDGE ENGINEERING
METHODOLOGY INVESTIGATIONS
FOR INTELLIGENT HYPERTEXT MANUALS
WITH A PROTOTYPE
AEROSPACE IMPLEMENTATION

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

William Andes Hoyt, Jr.
December, 1990

Dedicated to Susan,
my wife, best friend and greatest supporter.

ACKNOWLEDGEMENTS

Through his advice and guidance as my director during this research project, Dr. Ching F. Lo contributed immensely to the subject matter contained herein. In the course of the research I also received timely and scholarly advice from Dr. Ahmad Vakili, Dr. Roy Schulz and Dr. George Garrison who served as my committee members.

Part of the work contained herein was funded by the NASA Ames Research Center Contract NAS2-13203.

To the staff at The University of Tennessee-Calspan Center for Aerospace Research, of which the Center for Advanced Space Propulsion (CASP) is a division I express gratitude. They are Dr. Fred Speer, Joe Pawlick, Steve Matherly, Paula Reed, Linda Nelson, Verlin Fiendly, Brice Bible and Wim Ruyten.

A third group of people also receive my thanks. Dr. Susan Wu, my employer and president of Engineering Research & Consulting, Inc., provided a flexible and nurturing work environment, not to mention a touch of well-meant prodding. Beverly Powers, Richard Lin and Betty Hackett endured my many comings and goings, phone calls and interruptions. Mrs. Powers and Mrs. Hackett, in particular, helped input the text into the computer for the hypertext data base.

Dr. George Shi proved valuable in giving me expert and timely advice during the construction of the prototype aerospace implementation.

ABSTRACT

Investigations advancing the methodologies for implementation of artificial intelligence to document retrieval, applied to aerospace systems, are consummated with this thesis. The result is a unified approach to software implementation of an intelligent hypertext manual using hypertext programs and expert system shells. The idea behind the intelligent hypertext manual, its development and architecture are discussed. This thesis develops standard principles and rules for hypertext document construction, and knowledge implementation. The research then focuses on, in a general sense, forming a standard method to link hypertext software and expert system shells. A specific application of the methodologies originated into a prototype intelligent hypertext manual is discussed. This prototype is for reference documents used in conjunction with the size determination of wind tunnel models. This thesis presents hardware and software recommendations for implementing a turn-key intelligent hypertext manual.

PREFACE

This thesis resulted from work conducted in aerospace applications of expert systems started at the Center for Advanced Space Propulsion (CASP) and completed at Engineering Research & Consulting, Inc. CASP is part of The University of Tennessee-Calspan, Center for Aerospace Research. The author worked as a graduate research assistant at the Center for Advanced Space Propulsion with Ching F. Lo acting as the work supervisor from June 1989 to June 1990.

Much of the work contained herein has been applied to a project at Engineering Research & Consulting, Inc, of which the author is a team member. This company is one of several entrepreneurial concerns conducting business in The University of Tennessee Space Institute Research Park located in Tullahoma, Tennessee.

Dr. Lo, with good reason, deserves to claim the original idea behind this work¹. Even, if he does not receive pioneering credit for the idea of an intelligent hypertext manual, his support and investment in the concept rank him as a leader in intelligent hypertext manual applications.

¹Ching F. Lo and F. W. Steinle proposed the idea in a paper in January 1988 (Reference 3) while M. Williamson proposed the idea in "Computers in Science," January/February 1988.

During the course of working for CASP the author expected to write a thesis based on his work for a NASA sponsored project. The project would have resulted in a prototype intelligent hypertext manual for the Space Shuttle Hazardous Gas Detection System. However, funding for the project did not materialize when expected. Thus, the content of this thesis was changed so the author could attain a 1990 graduation date.

Thus, in an effort to complete the thesis, this work looked at the intelligent hypertext manual implementation in a general sense for broad aerospace applications. The author hopes this work will lay some groundwork for future intelligent hypertext manual projects at the University and in industry.

The foundation for this research lies in assignments accomplished during courses taken for the degree sought, work performed at CASP and the author's present job with Engineering Research & Consulting, Inc.

TABLE OF CONTENTS

CHAPTER	PAGE
INTRODUCTION.....	1
Prototype Aerospace Implementation.....	4
1. THE INTELLIGENT HYPERTEXT MANUAL.....	5
Historical Development.....	5
Existing Prototypes.....	6
Intelligent Computer Code Reference Manual.....	6
Project Engineer's Intelligent Assistant.....	7
Architecture and Components.....	7
Implementation and Control.....	8
Expert System Control.....	9
Hypertext Control.....	9
User Interface/Operating System Control.....	10
2. KNOWLEDGE ACQUISITION.....	12
3. STANDARD HYPERTEXT IMPLEMENTATION PROCEDURE (SHIP)...	15
The Data Base.....	15
Organization.....	16
Hypertext Links for SHIP.....	16
Data Base/Document Storage Methodology.....	20
Operating System and User Interface.....	25
Fonts.....	26
Information Formatting and Layout.....	27
Hypertext Link Visual Clues.....	29
4. STANDARD KNOWLEDGE IMPLEMENTATION PROCEDURE (SKIP)...	30
The Knowledge Base (Rules and Facts).....	31
Knowledge Base Storage Methodology.....	31
Rules for Rules.....	32
Consultation Displays.....	34
5. STANDARD HYPERTEXT INTELLIGENCE INTEGRATION METHOD (SHIIM).....	35
Integration Method.....	35
Organization.....	35
Components.....	36
Control and Hypertext Scripts.....	38
Operation.....	39

CHAPTER	PAGE
6. WIND TUNNEL MODEL SIZING PROTOTYPE IMPLEMENTATION....	41
Domain and IHM Platform Selection.....	41
Model Sizing Knowledge Acquisition.....	42
Domain Understanding.....	43
Documentation Source Identification.....	48
The Model Sizing Intelligent Hypertext Manual.....	49
The Model Sizing Hypertext Document.....	49
The Documentation Location Expert System.....	50
The Advisory Expert System.....	52
Test and Evaluation.....	56
7. HARDWARE AND SOFTWARE EVALUATION.....	57
Hardware Platform.....	57
Random Access Memory and Clock Speed.....	58
Peripherals.....	60
Operating System Software.....	66
8. CONCLUSIONS AND RECOMMENDATIONS.....	67
BIBLIOGRAPHY.....	70
APPENDICES.....	74
1. HYPERTEXT.....	75
2. EXPERT SYSTEMS.....	88
3. FIGURES.....	97
4. TRADEMARKS.....	114
VITA	116

LIST OF TABLES

TABLE	PAGE
1. SHIP links.....	19
2. File Names for Four-Level Structure with up to 99 Sections per Level.....	22
3. File Names for Eight-Level Structure with up to 26 Sections per Level.....	24
4. NASA Ames Aerodynamic Division Facilities (Wind Tunnels)w.....	46
5. Intelligent Hypertext Manual Computer Platforms.....	58
6. Memory Requirements for Expert System Shells and Hypertext Programs Used in Intelligent Hypertext Manuals.....	59
7. Software and File Loading Times for Intelligent Hypertext Manual Components.....	61
8. Typical File Storage Requirements for Data Base Files of an Intelligent Hypertext Manual.....	63
9. Uncompressed and Compressed File Sizes Using PK-Pak and PK-UnPak File Compression "Shareware".....	64
10. Hypertext Software Surveyed.....	82
11. Expert System Shells Surveyed.....	93

LIST OF FIGURES

FIGURE	PAGE
1. The Architecture for an Intelligent Hypertext Manual.....	98
2. Node to Node Links.....	99
3. SHIP Structure and Links.....	100
4. Hypertext Data Base Setup File in HyperCard.....	102
5. Helvetica and Times Fonts.....	102
6. Typical SHIP Table of Contents Display for the Model Sizing Documentation.....	103
7. Typical SHIP Text Screen for the Model Sizing Documentation.....	103
8. Typical SHIP Glossary Screen.....	104
9. Typical SHIP Index Screen.....	104
10. Typical SHIP Graphic Screen.....	105
11. Sample Rules from the Model Sizing Document Location Expert System.....	106
12. Sample Consultation Displays from the Model Sizing Document Location Expert System.....	107
13. Intelligent Hypertext Manual Data Flow and Organization.....	108
14. Specific Flow of Information and Control for the PEIA Model Sizing Module.....	109

FIGURE	PAGE
15. Hypertext Data Base Organization for Model Size Determination.....	110
16. Model Size Documentation Location Expert System Knowledge Base Organization.....	111
17. Model Size Advisory Expert System Knowledge Base Organization.....	112
18. Typical Hierarchic Structure.....	113
19. Cross-Referenced Data Base Structure.....	113
20. Combination Data Base, Hierarchic with Selected Cross References.....	113
21. Combination Data Base, Cross-Referenced with Hierarchies Existing at Selected Nodes.....	113

INTRODUCTION

Aerospace vehicles, support systems, test facilities, etc., require voluminous amounts of documentation. Information must be up-to-date and accurate.

Computers and mass storage technology offer the means to place typically extensive aerospace systems documentation into digital data bases. Two separate, proof-of-concept projects demonstrated that forging the link between a document data base and an artificial intelligence computer program would hasten information recovery. These systems were the result of a class project by this author [2] and a NASA Ames Research Center project [5]¹. The resulting prototypes are designated as Intelligent Hypertext Manuals (IHM) and are summarized in pages 6 through 7 in this Introduction. Intelligent hypertext manuals are used for efficient document retrieval, to make easy updates in documentation, to give advice for problem solving, and in computer aided instruction.

A means of rapidly implementing intelligent hypertext manuals merited investigation. It was the author's goal to increase the efficiency of engineers who form intelligent hypertext manuals for end-users. A unified approach for engineering intelligent electronic documents resulted from the work for this thesis.

¹Numbers in brackets refer to similarly numbered references in the bibliography.

Often, the end-user of an intelligent hypertext manual will specify software and/or hardware for an implementation. Since different software/hardware configurations require dissimilar structures, syntaxes, commands, operating systems etc., intelligent hypertext manual developers may be forced to "start from scratch" for each new application.

To avoid the need to start anew for each application, a look into means of hastening the intelligent hypertext manual building process was conducted. Foundational aspects of the existing proof-of-concept systems provided the impetus to advance the methods of building intelligent hypertext manuals. Therefore, the methodologies necessary for three areas common to any intelligent hypertext manual execution were studied. They were the hypertext document (data base) structure, knowledge (expertise) acquisition and implementation processes, and integration of the two. The knowledge engineer building an intelligent hypertext manual must make conscious decisions about each of these three foundational tasks.

This thesis resulted in three guidelines for forming intelligent hypertext manual structures and integrating its components as follows:

1. Standard Hypertext Implementation Procedure
(SHIP).
2. Standard Knowledge Implementation Procedure
(SKIP).

3. Standard Hypertext Intelligent Implementation Method (SHIIM).

Together these methodologies form the Intelligent Hypertext Manual Methodologies (IHMM).

Acquiring the knowledge of human experts is essential to the development of any system utilizing artificial intelligence. Knowledge engineers employ two means for knowledge acquisition, literature searches and interviews with known experts. Not only must the final intelligent hypertext manual contain specific knowledge that leads to solutions and answers, it must be "user friendly." For successful intelligent hypertext manuals, minimal user interaction with the interface is required. It is important that the end user does not need to learn how to run another complicated computer program. Thus, the ubiquitous requirement "of keeping it simple" for the developer and user played an important role during methodological studies.

The research herein, even further, investigated what unique foundational problems and solutions exist for aerospace applications. Using these standard methodologies an implementation for a sample aerospace application was built using two different hypertext packages.

The overall goal of this thesis is that methodologically formed implementations of intelligent hypertext manuals will result in maximized productivities of aerospace engineers working as knowledge engineers.

The techniques developed during this research were applied to a prototype intelligent hypertext manual using an aerospace domain. This example effort develops an intelligent hypertext manual to help engineers determine the size of models for wind tunnel tests.

The resultant intelligent hypertext manual consists of a hypertext program² and an expert system shell³ operating on a Macintosh computer. A comparison between two hypertext program implementations provides significant insight into efficient systems development. Full details of the example application appear in Chapter 6.

This example, in addition to locating pertinent documentation, contains an expert system that uses one-dimensional isentropic compressible flow theory and rules of thumb to advise the user what size a wind tunnel model should be for minimum wall correction calculations.

²Although a misnomer, the term hypertext describes computer-accessible electronic documents that contain text, tables, figures and graphics. Hypermedia is sometimes used to identify electronic documents that add computer-accessible video and audio information.

³A shell is a computer program with structures, and tools that is used to construct an expert system.

CHAPTER 1

THE INTELLIGENT HYPERTEXT MANUAL

Historical Development

The recent developments of expert systems⁴ and hypertext programs can be utilized to offer a conduit to quickly access the many volumes of information typical in aerospace systems. Expert systems enable experts to build systems that contain expertise without extensive help from computer scientists. Hypertext programs allow easy access to large amounts of textual and graphical information via micro-computers. Lo and Steinle [3] proposed the idea of combining expert systems and hypertext software into an intelligent hypertext manual. Descriptions of hypertext and expert systems are included in Appendices 1 and 2.

Hypertext serves as an efficient means to store and retrieve text and graphics. However, for users to take advantage of shortcuts offered, they must be intimately familiar with the hypertext data base. If not the user gets lost! In fact, one reviewer of hypertext programs commented, "Some hypertext systems give you control when what you may need is guidance [11]." That guidance is just what the intelligent hypertext manual gives to users.

⁴MYCIN, a consultation system for physicians, is considered to be the first expert system. Work on MYCIN Began in 1972 and culminated a decade later [17].

The ability to place experts' knowledge into computer programs offers a wide range of applications for the aerospace industry. In the intelligent hypertext manual the expert system serves two purposes. It advises the user where to find specific information in the hypertext data base and returns advice to the user about specific engineering problems.

Existing Prototypes

Two separate efforts at The University of Tennessee Space Institute confirmed the viability of the intelligent hypertext manual concept.

Intelligent Computer Code Reference Manual

In previous work this author built a prototype intelligent hypertext manual in an effort to prove the viability of the concept [2]. The system displayed documentation for computer codes utilized in wind tunnel testing at the Von Karman Gas Dynamics Facility at Arnold Engineering Development Center. The resultant intelligent hypertext manual used KMS, a hypertext program, and EXSYS, an expert system shell. Though the size of this intelligent hypertext manual was small, the experience gained by forming the system contributed immensely to the methodologies hypothesized by this thesis.

Project Engineer's Intelligent Assistant

A prototype intelligent hypertext manual was built for the NASA Ames Research Center based on wind tunnel testing documents [5]. The work was performed at ERC, Inc., a company located in the University of Tennessee Space Institute (UTSI) Research Park, and at UTSI. An expert system shell, VP-Expert, controlled the system and provided the user interface. Guide was the hypertext software utilized. This effort was funded through a Phase I Small Business Innovation Research Project. The NASA Ames Research Center continued funding with a Phase II contract for building a Project Engineer's Intelligent Assistant [8]. This intelligent hypertext manual will assist project engineers who conduct wind tunnel tests at NASA Ames.

The intelligent hypertext manual is a tool for engineers and technical personnel who deal with large volumes of reference documents and manuals. When dealing with large and complex aerospace vehicles and systems, people spend a considerable amount of effort locating, reviewing and analyzing the correct documentation.

Architecture and Components

Lo, et al, developed an architecture for the intelligent hypertext manual (Figure 1, Appendix 3)⁵ [8].

⁵Figures appear in Appendix 3 beginning on page 97.

This structure consists of three layers of software and data bases built on top of a particular computer operating system and platform.

The bottom layer contains the commercial software packages that the developer (and user) execute. The two required packages are a hypertext program and expert system shell. For developing and accessing existing textual, tabular graphical and relational data bases; word processor, spreadsheet, drawing and data base programs can be included.

The middle layer contains the different types of data bases that may exist or need to be built to sufficiently represent the domain. Additionally, if the intelligent hypertext manual needs input based on engineering computer codes, these codes reside in the middle layer.

At the top layer a user interface presents information to the user via the display, printers, etc. The user interface can be separate from all the programs or it can utilize the existing interfaces of each program as it executes.

Implementation and Control

Numerous combinations of the programs and data bases, as based on the architecture of Figure 1, Appendix 3, might constitute an intelligent hypertext manual.

Expert System Control

Calling the hypertext program, spreadsheet and data base programs from an expert system to provide information to the user constitutes the basis of using expert system control. Some of the first additions to second generation expert system shells included the ability to directly access spreadsheet files and relational data base files like those generated by 1-2-3 and dBase IV, respectively.

Thus, the use of expert systems to control the execution of an intelligent hypertext manual seemed natural. The Project Engineer's Intelligent Assistant applied its expert system shell (VP-Expert) as the program of control which called the hypertext program (Guide) to display the hypertext data base. Until expert system shells become fast, operate under control of graphical user interfaces and/or multi-tasking environments their control of intelligent hypertext manuals will remain limited.

Hypertext Control

Controlling the intelligent hypertext manual with a hypertext program offers the user direct access to the hypertext data base. Expert systems give the user little control over where to go and do not provide the wealth of information available in hypertext data bases. Therefore, hypertext control lets the user explore the hypertext data

base freely. When help is needed the user can invoke an expert system for navigational help and/or advice.

This control of the intelligent hypertext manual relies on a script language within the hypertext program to call and read output from the expert system. The script will display the appropriate information within the hypertext data base as a result of the consultation data returned by the expert system.

The Intelligent Computer Codes Reference Manual [2] used hypertext (KMS) as the controlling program which called an expert system (Pro-EXSYS) to provide guidance only as the user wanted help.

Hypertext control offers the best compromise between access to information and development effort.

User Interface/Operating System Control

Expert system or hypertext control of the intelligent hypertext manual leave the developer tied to existing user interfaces and limitations.

Developing the intelligent hypertext manual under the control of a powerful programming language that provides display control, multi-tasking and data exchange features may offer the best overall performance. This programming language, in concert with the existing operating system, actually provides the user interface. As viewed from the user's point of view it becomes the operating system. Based

on input from the user, data bases and external data sources (sensors) the correct program will be called as needed to give the user proper documentation and/or advice.

This third method was not used for the Intelligent Hypertext Manual Methodologies because of the usual long learning curves required for developers to become proficient programmers in these type of languages, e.g. Small Talk.

Developing a new user interface and control program using languages should provide the best system, but with long lead times. However, using commercially available programs with script languages reduces the development effort so engineers and experts themselves can implement intelligent hypertext manuals.

CHAPTER 2

KNOWLEDGE ACQUISITION

Specific steps that the knowledge engineer should take to successfully acquire knowledge for the expert system component of the intelligent hypertext manual are addressed here. The first requirement is that a knowledge engineer with an aerospace background should build the expert system for the intelligent hypertext manual.

Effective knowledge acquisition requires three basic steps: establishing realistic goals for the document retrieval system, specifying the domain and its associated documents, and capturing the thought processes that an expert uses to locate and utilize information within the appropriate documents.

One high-level goal is an inherent part of the intelligent hypertext manual--to develop an expert system that will display documentation of interest. Troubleshooting and problem solving goals can be added.

Overall goals of the aerospace system of interest, its contribution to larger systems, its subsystems, and sub-goals merit exploration by the knowledge engineer. This effort will help the knowledge engineer define the specific expert system domain, and will aid in domain comprehension. Once a working knowledge of the domain is at hand the

engineer can start to build components of the intelligent hypertext manual.

The domain for the intelligent hypertext manual is divisible into two parts, the specific aerospace system of interest and its related documentation. A knowledge engineer starts the knowledge acquisition process by answering the following questions developed for this thesis:

1. What is the name of the system of interest? If a name does not exist, what is an appropriate one?
2. What is the technical goal of the system?
3. If this system is a sub-system, what is the larger system and how do the two relate?
4. Is a goal of the intelligent hypertext manual to help design a new system, plan use of the system, provide advice for problem solving, etc.?

Once a specific, but preliminary, domain is established, the developer needs to determine what documents are applicable to the intelligent hypertext manual. During the knowledge acquisition task the developer must inquire about the specific use of the documents. The expert must explain how he or she determines which part of the document to reference for each situation encountered. The choice of documents is critical to the implementation. In engineering, people document systems, write about theories and ideas of the system workings, but they seldom

record why and which documents are utilized to conduct their job.

For the intelligent hypertext manual, a literature search has two purposes. The search provides valuable background information to the knowledge engineer for conducting interviews with experts, and identifies sources of documents not named during interviews. Using a literature search, the knowledge engineer can start establishing the size, and can think about the organization of the hypertext data base.

Interviews are the predominant method for acquiring knowledge from experts. Planning the interview is important to avoid unfruitful sessions. A well conducted interview demonstrates to the expert that a structured method within control of the knowledge engineer will lead to an applicable and functional intelligent hypertext manual [24]. An outline of every session should be formulated by the knowledge engineer and given to the expert.

The knowledge engineer is addressing two facets of knowledge acquisition when developing the intelligent hypertext manual. Knowledge is acquired to find out the "why" of documentation use, and the "which" for identifying the documents used by the expert.

CHAPTER 3

STANDARD HYPERTEXT IMPLEMENTATION PROCEDURE (SHIP)

Builders of any hypertext document, whether it contains intelligence or not, need to establish a consistent presentation for the user. This consistency comes from the paradigm utilized to build the user interface and data base.

A model is developed here that applies to basic aspects of the intelligent hypertext manual. This chapter puts forth a standard procedure that facilitates the actual hypertext implementation so that the organizational model, linking model and formatting/layout decisions need not be made anew for each new system developed. This paradigm is designated the Standard Hypertext Implementation Procedure (SHIP) and addresses the requirement of the larger task to integrate hypertext into an intelligent hypertext manual.

A review of hypertext and three commercial hypertext programs is presented in Appendix 1.

The Data Base

The Standard Hypertext Implementation Procedure gives the hypertext data base for intelligent hypertext manuals enough structure to aid in its composition, and to present familiarity to the user. Its structure allows flexibility for adapting to different applications.

Organization

Printed documents naturally follow hierarchic structures and provide the reader with the most familiar type of an information presentation scheme. Thus, the hypertext data base should follow a hierarchic structure. Users may never realize that they are using a hierarchy, but they will sense its familiarity based on experience with printed documents. However, to be successful the hypertext document must provide advantages in speed and ease of use over printed material.

The Standard Hypertext Implementation Procedure specifies the use of distinct screen-size pieces of information to provide the best support on all platforms and operating systems. Some hypertext programs only support this type of display (e.g. KMS and HyperCard). The organization will follow that of printed books with a preamble, body, references, appendices, glossary and index. The preamble can include the following: title page, copyrights, acknowledgements, preface/foreword, table of contents, list of figures, list of tables and nomenclature.

Hypertext Links for SHIP

Active links between parts of the hypertext data base provide the key to rapid document retrieval (Appendix 3, Figures 18-21). In addition to hierarchic links, accessible with a simple mouse click, the hypertext builder can add

numerous cross references from one part of the data base to another. A natural cross-reference links an index entry to the word in the document. However, if too many and haphazard cross references are added, the hierarchy becomes meaningless. Too many links seriously hamper development efficiency.

The number of links possible with hypertext easily becomes overwhelming. Referencing each node in the data base once, from every other node, builds an enormous progression of links. These types of links are called node-to-node links (Figure 2, Appendix 3). The formula governing the possible number of node-to-node links (L) was derived by the author is as follows:

$$L = \sum_{i=1}^{n-1} (n-i) \quad (1)$$

Where:

L = number of hypertext links

n = number of nodes in the data base

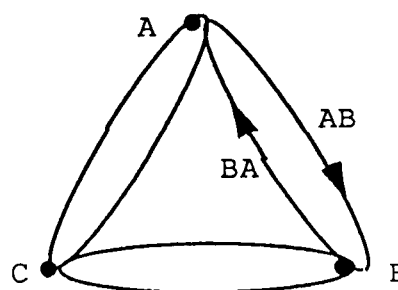
i = summation counter

After putting in integral form, integrating and substituting in the limits the number of links becomes:

$$L = n^2 - n \quad (2)$$

This equation shows that the number of links increases nearly with the square of the number of nodes in the hypertext data base.

This example of a three node network has six links. As stored in the computer a link from A to B is not the same as a link from B to A.



This author calculated that the Phase I intelligent hypertext manual for the Project Engineer's Intelligent Assistant contained 22 files (nodes). Therefore, the number of possible node-to-node links would be 462. . By using the organization described above and the links that follow, this author built the same data base with 114 links, while providing the user with significantly better control.

The links in the data base should be defined in a manner that enables the user to understand the result of an action (i.e. where will a link direct the user and from whence in the data base did it bring him or her?).

The Standard Hypertext Implementation Procedure incorporates only one mandatory link to the expert system. This procedure suggests four hierarchic links and adds to it a predefined type and number of cross-referenced links, making a total of nine suggested, and six optional links (Table 1). For the intelligent hypertext manual, a link to activate the expert system to help the user navigate within hypertext is the mandatory link. Otherwise the system would simply be a hypertext document, at best, without any guidance.

Table 1. SHIP links.

Link #	Description/Location	Type/Use
1	Consult (Gets Expert Help) Link Menu	N/A Mandatory
2.	Step-down Link Menu	Hierarchic Suggested
3.	Step-up Link Menu	Hierarchic Suggested
4.	Jump-down Link Menu	Hierarchic Suggested
5.	Jump-up Link Menu	Hierarchic Suggested
6.	Table of Contents Link Menu	Cross-referenced Suggested
7.	Index Link Menu	Cross-referenced Suggested
8.	Gloss Link Menu	Cross-referenced Suggested
9.	Back (Previous or Last) Link Menu	Mixed Suggested
10.	Numbered Section Tables of Contents	Mixed Suggested
11.	Figure Body Text/List of Figures	Cross-referenced Optional
12.	Table Body Text/List of Tables	Cross-referenced Optional
13.	Index Entry Index	Cross-referenced Optional
14.	Gloss Entry Body Text	Cross-referenced Optional
15	Top of File Body Text	Cross-referenced Optional
16	Help File Link Menu	Cross-referenced Optional

All screens displayed must contain the mandatory "consult" link. It is strongly suggested that from anywhere in the data base the user should have access to the table of contents, nomenclature, index and glossary. Therefore, links to these entries, contained in a "link menu," should be placed at the top of every display. Other links that reside in the link menu include help, step-up, step-down, jump-up, jump-down, and back.

Diagrams of the data base structure and each link are shown in Figure 3, Appendix 3. Note that some of the links are mixed (hierarchic or cross-referenced) depending on what file of the data base is currently displayed. The "help" link extends the system to the built-in hypertext help facility, if one is available. Cross-referenced links are represented in two ways, text or graphic elements.

To eliminate redundant links, only links with possible actions are presented to the user. For example, if the user is at the table of contents there are no links up to a higher level for that document, thus the links step-up and jump-up are not included in the table of contents link menu. The table of contents is the default entry point and will have one hierarchic link to the data base, "step down."

Data Base/Document Storage Methodology

For the Standard Hypertext Implementation Procedure each node of the data base will be stored in a file on a

mass storage device. Since the methodology is to be device and operating system independent the file structure will take on the form of the least capable operating system.

Of the operating systems contemplated as environments for the intelligent hypertext manual, IBM/MS DOS imposes the most strict method for naming files. File names are limited to 11 characters, eight characters followed by a period and three for an extension. Therefore, when the user has control of hypertext file names⁶ the Standard Hypertext Implementation Procedure will limit names to the first eight characters of the DOS syntax. Names will always appear in upper case since DOS does not distinguish character case. Using the DOS syntax, two methods of file storage are specified by the Standard Hypertext Implementation Procedure. Both methods include identical document preambles. The first uses numeric names for the hypertext files (sections), with the exception of the glossary and index, as shown in Table 2. Each file is represented by a progression of numeral pairs. Each pair, reading from the left, identifies the level of the particular section (i.e. six numerals, three pairs, indicate the third level of the document hierarchy). Four levels are possible (8/2).

⁶The user cannot always control file names in hypertext. For example, KMS assigns file names to each frame of information created by the user based on a "tree" name given by the user. If the user names a tree "aero" then each frame will be named aero1, aero2, aero3, ... aero(n-1), aero(n), aero(n+1), ...

Table 2. File Names for Four-Level Structure with up to 99 Sections per Level.

Document Section	Example	Name	Notes
Preamble	Table of Contents Following Screen Title Page List of Figures Following Screen List of Tables Following Screen Copyrights Acknowledgements Preface/Foreword Nomenclature Following Screen	0 0A 00 000 000A 0000 0000A 00000 000000 0000000 0000000A	Entry Pt. Follow these entireties by A, B, etc., if needed
Body	Section Subsection sub-subsection sub-sub-subsection . Intermed. level 3 . Intermed. level 1 . . . Bottom-most section	01 0101 010101 01010101 . 040103 . 07 . . . 99999999	First Example Example Last
Appendices	First Page Following Pages	APP01 APP02	
References or Bibliography	First Page Following Pages	BIB01 BIB02	
Glossary	First Page Following Pages	GLOSS01 GLOSS02	
Index	First Page Following Pages	INDEX01 INDEX02	

The possible number of sections is thus 99^4 or almost 100 million sections for the document.

The second file-naming scheme uses letters. This hypertext method allows for eight levels of up to 26 sections each. Each file is represented by a progression of single alphabetic characters. Each character, reading from the left to right, identifies the level of the particular section (e.g. seven letters indicates the seventh level of the document hierarchy, eight levels are possible). This provides the possibility of 26^8 or over 200 trillion sections for the body (Table 3).

These structures include provisions for the nomenclature, lists of tables and figures, references, indices and appendices. The Standard Hypertext Implementation Procedure will not address directory structures for hypertext other than to specify that each unique document exist within a unique unit like a file or directory.

Building on this data base structure allows the Standard Hypertext Implementation Procedure to contain setup files for each portion of the data base (i.e. the link menu, display options, section numbers, text fields, etc.). Figure 4, Appendix 3, shows an example hypertext setup file.

Table 3. File Names for Eight-Level Structure with up to 26 Sections per Level.

Document Section	Example	Name	Notes
Preamble	Table of Contents Following Screen Title Page List of Figures Following Screen List of Tables Following Screen Copyrights Acknowledgements Preface/Foreword Nomenclature Following Screen	0 0A 00 000 000A 0000 0000A 00000 000000 0000000 0000000A	Entry Pt. Follow these entires by A, B, etc., if needed
Body	Section Subsection sub-subsection sub-sub-subsection . . . Eighth subsection . Intermed. level 1 . Intermed. level 5 . . . Bottom-Most section	A AA AA AAA . . . AAAAAAA . G . RABCA . . . ZZZZZZZZ	First Example Example Last
Appendices	First Page Following Pages	APP01 APP02	
References Bibliography	First Screen Following Screen	BIB01 BIB02	
Glossary	First Screen Following Screen	GLOSS01 GLOSS02	
Index	First Screen Following Screen	INDEX01 INDEX02	

A very important software note must be made. Since some hypertext programs only display one window, the Standard Hypertext Implementation Procedure allows only one window. This limitation actually simplifies the process and gives the user a more consistent and less confusing display.

Since the purpose of this thesis is to develop the methodology for building an intelligent hypertext manual independent of the computer hardware and operating system, the Standard Hypertext Implementation Procedure does not specify an operating system choice. For the most efficient execution of the intelligent hypertext manual the system must operate under control of a true multi-tasking environment. The very nature of the intelligent hypertext manual architecture which uses available commercial programs requires that separate programs must run concurrently.

Once the intelligent hypertext manual builder knows the operating system and platform to be used, specific steps can be taken to introduce common elements into the user interface as follows:

1. Fonts applied to text.
2. Information formatting and layout.
3. Visual cues to show the hypertext link points.

These elements, when applied by publishers of printed material (except for the links), describe the make-up and design of a publication. Thus part of the Standard

Hypertext Implementation Procedure involves the make-up and design of the hypertext displays. The user interface contains many other elements such as operating system menus, mouse pointer shapes, and display colors. These items will not be included in the Standard Hypertext Implementation Procedure since some hypertext programs do not allow their control.

Fonts

Hypertext programs allow the user to pick the size and the style of the type used for text. The style or design of type is its face. A specific size and face define the font. Based on this author's experience no more than three faces should appear in a given visual unit (the display screen). When three faces are employed, two predominant faces are preferable with a third used sparingly to add emphasis to specific items.

To follow this typographic rule of thumb, the Standard Hypertext Implementation Procedure specifies that text use a sans serif typeface with regular and bold faces. Sans serif faces, as opposed to serif, appear "cleaner" and are easier to read on a display screen. To compromise between the amount of information that can be displayed on the screen and readability, the type should be near 12 and not less

than 10 points⁷. Figure 5, Appendix 3, contains samples of regular and italic sans serif and serif faces.

Information Formatting and Layout

A guideline for the format and layout of information on the display screen is included as part of the Standard Hypertext Implementation Procedure. All documents, printed and electronic, need formatting guidelines.

The Standard Hypertext Implementation Procedure designates five distinct display screens, as follows:

1. Table of Contents, Subsequent Tables of Contents.
2. Text.
3. Graphics.
4. Glossary.
5. Index.

Table of Contents. The table of contents is a hierarchic structure with chapters, sections, subsections, sub-subsections, etc. Each subsequent division should be indented two to five additional spaces from the left margin. Each entry in the Table of Contents contains a descriptive term of the referenced entry which is preceded by either a number or letter designator. This designator corresponds to the particular section of the document as well as serving as

⁷A point is approximately 1/72 of an inch and is the standard of measure used for sizing type. Type is measured from the bottom of the descender on a character like "y" to the top of the ascender on a character like "h."

the hot spot for the link to that section (Figure 6, Appendix 3, and Tables 2 and 3).

Subsequent tables of contents are for sections that have further subdivisions. For example, if the user goes to Section 2 and there exists sections 2.1, 2.2, 2.3, etc., then a Section 2 table of contents showing its entries and links to those entries is displayed.

If the display screen is much wider than it is long, as in HyperCard displays, the table of contents may be displayed in two columns to take advantage of the display shape.

Text (Body, Glossary and Index). Textual displays should look very much like a book. The screen will first have its link menu at the top. Next, the section numbers, from the first level to the level of this particular section are listed with their links and headings. Last, the text for this particular entry appears. Text is arranged in un-indented paragraphs with double spacing between paragraphs. The left and right margins are within one to five character spaces from the screen edges.

The Glossary and Index are considered text screens. As an addendum to the document, they do not have section numbers listed. Just below the link menu at the top of the glossary or index screens will be a link to additional glossary or index information as needed. The link is represented by the alphabetical range each screen of

information contains (e.g. "A-G," "G-P," and "Q-Z"). If the hypertext program supports multiple columns, index entries should be placed in such an arrangement. Examples of text screens are shown in Figures 7 to 9, pages 103 to 104.

Graphics. To make graphics useful, most engineering drawings should be displayed on a full screen. The common link menu should be placed at the top of the graphics screen. Black on white displays are recommended. As with textual screens the font for explanations, dimensions, etc., should be sans serif, 10 to 12 point type where possible. A sample graphics screen is shown in Figure 10, Appendix 3.

Hypertext Link Visual Clues

Links to different parts of the hypertext data base must stand out from other elements in the display. All links should be in bold text and boxed. Actions represented by the link menu contain visual clues. If a link takes the user up or down in the structure then an arrow pointing in the appropriate direction should accompany the link (e.g. Step $\wedge$, Step $\vee$).

CHAPTER 4

STANDARD KNOWLEDGE IMPLEMENTATION PROCEDURE (SKIP)

Knowledge implementation for an intelligent hypertext manual consists of organizing the knowledge base and writing its rules and control strategy for the expert system. This chapter outlines procedures developed here for construction of the knowledge base and user interface display.

Expert system shells⁸ usually do not provide the user with step-by-step means to control the flow of a consultation. Unlike traditional programming where flow control is very important to the developer, the shell uses the inference engine to control system execution. The knowledge engineer who has experience in programming with high and/or low level languages must realize this fact. The developer specifies goals, the type of chaining to be executed, output display formats, and some input required from the user. The knowledge engineer need not, and should not try to place procedures into the system. The inference engine should carry out procedures [24].

The expert system within the intelligent hypertext manual is goal oriented and should be built using backward chaining, leaving open the possibility of some forward chaining rules for limited control. The goal is to provide

⁸Expert system shells for the remainder of this chapter will be referred to as shells.

locations representing documentation of interest to the user, based on input from the user and/or data provided by the aerospace system. Shells provide control of serial ports that can be used to pass data to the inference engine when needed. Ancillary goals will advise the user about a course of action to take in solving problems. The resulting expert system is considered advisory because it does not take any action affecting the application aerospace system.

The following sections use generic terms to describe shell elements since different shells vary terminologies even though referenced items are essentially identical from one shell to the next.

The Knowledge Base (Rules and Facts)

Knowledge Base Storage Methodology

Shells usually do not require that rules be placed in any particular order. However, rule firing is sensitive to the order in which rules reside. The Standard Knowledge Implementation Method specifies that the knowledge engineer organize the rules in a manner that eases implementation and maintenance. The following list provides some guidelines for building the knowledge base:

1. Group rules of like nature together [18].
2. Place goals and facts first. Usually shells will require goals in a certain location [18].

3. Place display and formatting statements together.
Usually these will go with the control statements.
4. Put control rules together at the end of all other rules when possible.
5. Prioritize rules and put those that can quickly reach goals after control rules and goal declarations. The goals sought for the intelligent hypertext manual are documentation locations.
6. Use the commenting feature of the shell to add relevant information about the knowledge base.
7. Well placed comments with blank lines, special characters to create boxes, dividers, etc., can make the knowledge base easier to read, debug and maintain.
8. Use comments to explain how the developer introduced control into the expert system.

Rules for Rules

All rule semantics should use wordings that are easily understandable. Do not make rule names, clauses, actions, etc., cryptic. With shells the developer is able to use natural language construction in most cases.

Rule Names. For ease of documentation and referencing, name rules when the shell allows. Avoid cryptic names. They need to reflect terms commonly used in the hypertext

data base. Rules that return actual hypertext data base locations should have names that reflect either the associated hypertext file name or be descriptive.

Rule Syntax. For consistency among knowledge bases, rules should utilize the following format standards developed for this thesis when possible (the standard assumes only one font is available):

1. Capitalize rule names: the words "If," "Then," "Else," "Display," "And," "Or" and any other key words provided by the expert system.
2. Place the rule names, "If," "Then," "Else," and other clauses with specific expert system supplied key words on lines by themselves.
3. Use hanging indents to keep all clauses of rules the same distance from the left margin.
4. If clauses have more than one line, indent the second and subsequent lines more than the preceding level.
5. Use "Else" statements when a rule should always fire for control.

Sample formats of expert system rules are shown in Figure 11, Appendix 3.

No comments on uncertainty and confidence factors have been included here. With the speed of computers today and through the quick access offered by hypertext, the need for confidence factors for document location specifications is

eliminated. Let the user see all possible references that the expert system suggests. It is the job of the knowledge engineer to ensure that the intelligent hypertext manual user is not given too many choices. The order of documents suggested by the expert system should prioritize choices.

Consultation Displays

In general, displays construction should be kept simple. Dazzling displays that do not provide guidance in using the expert system do not help the user complete his or her task. If color is applied to the display, use simple color schemes. One display tested used red characters on a bright blue background and was nearly impossible to read [20]. Keep backgrounds muted and in sharp contrast to the information displayed. Do not make the user wade through repetitive screens of information at the beginning of each session. Get to the point and start the consultation with as few keystrokes as possible.

Make the user interface concise. Succinct functional displays require the most amount of thought to construct, and result in elegance through simplicity. If the expert system results are not useful or gained rapidly, a dazzling display will fail to make the end user a regular user. Sample consultation displays are shown in Figure 12, Appendix 3.

CHAPTER 5

STANDARD HYPERTEXT INTELLIGENCE

INTEGRATION METHOD (SHIIM)

The ubiquitous representation of the intelligent hypertext manual developed for the Project Engineer's Intelligent Assistant [8] serves as the architecture for the Standard Hypertext Intelligent Implementation Method (SHIIM) developed in this chapter (See Figure 1, Appendix 3). Based on that architecture, and using an information and control flow chart (Figure 13, Appendix 3), this chapter addresses the relationships of the user, knowledge engineer, expert and components of an intelligent hypertext manual. This flow chart provides the organization to follow for integration of the hypertext documents and expert systems into an intelligent hypertext manual. The intelligent hypertext manual uses a script to start, control, and end the consultation with the expert system.

Integration Method

Organization

The integrated environment of the intelligent hypertext manual includes the user, developer, expert, original documents, document input system, hypertext program and data base, expert system shell and knowledge base, aerospace

system data sources, and the document location and navigation information.

A four level organization exists for the Standard Hypertext Intelligent Implementation Method. The typical user (first level) accesses only the hypertext program and expert system shell as denoted by the bold lines of Figure 13, Appendix 3. Below that, the hypertext program and expert system shell and their data bases form level two, the executable portions of the intelligent hypertext manual. The third level includes data and information input components. The knowledge engineer, domain expert and document sources comprise the bottom level. The user who maintains the intelligent hypertext manual will also enter the system from this last level.

Components

First and Fourth Levels. Inputs from personnel make up the first and fourth level. The user accesses the hypertext program immediately upon entering the intelligent hypertext manual. The user interface is thought of as level 1 and utilizes the hypertext and expert system interfaces from those programs on level two. The knowledge engineer will work from levels one and four during the system construction, check out and maintenance. To test the system both the knowledge engineer and expert should assume the role of the user at level 1.

Second Level. Both the hypertext program and expert system, and their respective data bases make up the second level. The hypertext data base comes from the original documents chosen. The expert system knowledge base comes from the domain expert with reliance on the original documents for background knowledge. To allow for modular development and easier checkout, multiple knowledge bases are desirable for a complete intelligent hypertext manual.

Third Level. The Document Location Information and Document Navigation Information serve as the integrating feature between the expert system and hypertext program. The expert system reads information from the Document Location Information to create the Document Navigation Information. The knowledge engineer creates an entry in the Document Location Information for each document location that the expert system returns to the user. The Document Location Information holds the link name of each hypertext reference link.

The Document Navigation Information is an extraction of the Document Location Information for the documentation that the expert system determines will be of interest to the user. These results are based on user input and external input sources. The expert system performs the extraction based on results of its execution.

Another component on this level is the document input system. Currently the system is no more than a typist

retyping documents into the data base. The use of optical scanners as automatic document input systems is a possibility. Users who need to maintain hypertext documents will use the document input system and program editors.

In addition, the third level employs means for external input to the expert system and hypertext program. Input may come from the following: data from the aerospace system sensors, video and/or audio, spread sheet files, relational data bases, and/or executable engineering programs.

Control and Hypertext Scripts

Experience gained from work on the Intelligent Computer Codes Reference Manual and Phase I of the Project Engineer's Intelligent Assistant both completed, and the Hazardous Gas Intelligent Hypertext Manual and Phase II of the Project Engineer's Intelligent Assistant both under development' offer insight into the best integration method for the intelligent hypertext manual [2] [5] [4] [8]. Using the hypertext program as the controlling mechanism and primary interface gives a general integration method with flexibility, yet control, as needed.

Two reasons contribute to the choice of hypertext as the controlling program. The first comes from the overall goal of the intelligent hypertext manual which is to make the retrieval of pertinent information quick. The bulk of the information to be presented exists in the hypertext data

base. Thus the user should spend the most time using hypertext. Information is displayed directly and expert help is available when desired by the user.

As investigation continued, the requirement of using a script language became abundantly clear. The various script languages that exist in hypertext programs provide the second reason for using hypertext control. Scripts proved powerful in past implementations--enabling the expert system and hypertext program to communicate. Scripts provide a tool to add control to the system.

The Project Engineer's Intelligent Assistant [8] serves as an example of the importance of script languages. In this system the hypertext data base is accessed with a version of Guide that does not have a script language. Therefore to successfully implement the system, HyperCard with its script language (HyperTalk) controls the session. Guide is used to display enhanced screens of information as needed.

Operation

A multi-tasking operating system is a necessity for implementing intelligent hypertext manuals. As the memory requirements of expert systems and hypertext programs grow, the need to operate in a true multi-tasking environment becomes apparent. A new version of LEVEL5 takes a full megabyte of memory to operate efficiently (see Chapter 7).

To operate the automated consulting scheme the user clicks with the mouse on the Link Menu (see Chapter 3) hot spot labeled "Consult." The script language takes control. The exact program flow for the prototype aerospace implementation (Chapter 6) is shown in Figure 14, Appendix 3. The consult button is mandatory for all screens, thus the user can get help from anywhere, even when lost. After clicking on "consult," the script through the operating system calls a command which directly opens the expert system and appropriate knowledge base. During the consultation session the user will also use the expert system interface.

Within the expert system whenever a document location rule's "if" statement is determined to be true "then" a command is executed that reads the Document Location Information. This information is then passed to the Document Navigation Information. The resulting Document Navigation Information contains the cumulative results of all the possible locations where the user may find germane information in the hypertext data base. Control is passed back to the hypertext program and script to display the documentation as suggested by the expert system.

WIND TUNNEL MODEL SIZING PROTOTYPE IMPLEMENTATION

To test, verify and identify areas needing improvement, an intelligent hypertext manual for an aerospace related application was implemented utilizing the Intelligent Hypertext Manual Methodologies (IHMM). Providing documentation and expert advice on how to determine the appropriate sizes of models for wind tunnel tests served as the domain of this prototype implementation.

Domain and IHM Platform Selection

The prototype implementation came from the Phase II Project Engineer's Intelligent Assistant for a NASA Ames Research Center Project [8]. The NASA Ames task was chosen because this thesis would help the progress of the project and a "real-world" example was needed to test the methodologies of this treatise. Rightly so, the author, with the advice of professors, his own experience and knowledge gained from documents, served as the principal domain expert.

The model sizing task lends itself to using the Intelligent Hypertext Manual Methodologies. Text, graphs, figures and tables exist in available documentation which were input into the hypertext data base. The domain is specific, yet needs real engineering judgment to reach final conclusions. Rules of thumb exist for sizing wind tunnel

models but clear-cut formulas do not. This application required use of all the components of the intelligent hypertext manual environment (Figure 13, Appendix 3) except for external data inputs, e.g. sensor, video or audio.

The Macintosh personal computer was specified for the NASA Ames Research Center Phase II Project. Thus, HyperCard was selected as the hypertext program since it comes with all Macintosh computers and offers a powerful script language, HyperTalk. Guide was selected as a supplemental hypertext program because HyperCard is limited in its display capabilities. Guide gives better displays and easier document setup, but does not provide enough control to the developer. LEVEL5 served as the expert system shell since it was already available and runs on the Macintosh.

Model Sizing Knowledge Acquisition

Defining the goals and domain for the model sizing intelligent hypertext manual are two of the three steps required to define task. Therefore, based on the high-level goal of the intelligent hypertext manual and the needs of the NASA project, a two-part goal for this implementation was established as follows:

1. Establish which documents, or portions thereof that deal with model sizing, are pertinent to a specific wind tunnel test program (the high-level goal of any intelligent hypertext manual).

2. Establish limits and criteria for determining the size of the model to be used for the specific wind tunnel test. These limits are returned to the project engineer as advisory output.

The hypertext data base contains information which can help the user reach each goal. The user receives specific documentation or advice from separate expert systems using distinct knowledge bases for each goal above.

Domain Understanding

The next step is to understand the domain and identify relevant knowledge. Therefore, an understanding of the under-lying goal of the model sizing task was established.

Model size determination is vital so that results from the wind tunnel tests will be as close to free-flight conditions as possible, or applicable to analysis of full-scale problems. If the model is too small it may not accurately represent the full-scale vehicle because manufacturing tolerances cannot be held tightly enough as the model scale decreases. If the model is too large interference effects with the wind tunnel walls will cause discrepancies in data. The advisory expert system returns model size guidelines so that interference corrections can be a minimized.

In the effort to match free-flight conditions in wind tunnels, engineers have pushed for larger models. For many

large scale effects, such as lift and drag, matching the Reynolds number between the test and free-flight cases is a fundamental and elementary requirement of wind tunnel testing. Larger models result in larger test Reynolds numbers. Typically, the test Reynolds numbers are lower than those of free flight, mainly due to model scaling.

Several criteria can be established to help the engineer determine the design and size of a particular wind tunnel model. Some examples follow:

1. Tunnel to be used.
2. Velocity regime--subsonic, transonic, supersonic, hypersonic.
3. Type of model--wing, body, wing-body, inlet, ground plane used, reflection model, non-aerospace (buildings, bridges, etc.), blunt body, etc.
4. Type of test--pressure, force, flow-visualization, stability and control, etc.
5. Pitch and yaw angle limits for the test.
6. Desired Reynolds number for the test.
7. Actual full-scale vehicle dimensions--span, length, planform area, height, etc.

Wind Tunnel Considerations. It is obvious that information for each wind-tunnel must be placed into the knowledge base. Table 4 contains a list of the wind tunnels applicable to this task [36]. For instance, when using the Ames 11x11-Foot Transonic Wind Tunnel, the expert system must account for the longitudinal location of the model in the test section based on the Mach number of the test to assure that the model is out of flow regimes created by expansion and compression shock waves from the tunnel nozzles. These forward and rear limits of the model location vary from 60 to 100 and 175 to 220 inches respectively, relative to the beginning of the tunnel test section. Thus, model length must also be based on specific limitations of the test section.

For the expert system to provide adequate conclusions, certain assumptions and limitations about the test and the model criteria must be established. One such assumption is that the project engineer may know which tunnel will be used for the test. In addition, the test speed regime might be considered as a known value. By stating that these facts are known, it is meant that the project engineer knows the information for input into the expert system. Based on either of these two criteria alone, the correct wind tunnel technical information can be flagged. Also, the model size determination documentation, as based on the Mach number regime, e.g. transonic, can be flagged by the expert system.

Table 4: NASA Ames Aerodynamic Division Facilities
(Wind Tunnels).

Tunnel	A ft ²	L ft	Re x10 ⁶ /ft	M	P _T psia	T _T °R
11x11-Foot Transonic	121	22	0.3-9.6	0.3-1.5	3-32	600
9x7-Foot Supersonic	63	18	0.9-6.5	1.55-2.6	4.4-29.5	600
8x7-Foot Supersonic	56	16	0.7-5.2	2.45-3.5	5.0-28.3	600
2x2-Foot Transonic	The 2-by 2-Foot Transonic Wind Tunnel is presently out of service.					
12-Foot Pressure	116	18	0-3.2	0-0.9	2.5-14.7	625
14-Foot Transonic	185	34	2.8-4.2	0.5-1.15	14.7	640
6x6-Foot Supersonic	36	14	1.0-5.0	.25-2.2	4.4-16.0	580

Subsonic Flow Considerations. A rule of thumb that applies to low-speed wind tunnel testing is that the model be no more than 80 percent of the tunnel test section width [32]. The reason is two fold. First all of the model should be kept out of the wind tunnel boundary layer, which is reduced energy flow that would not be encountered in free flight. Second, wing-tip vortices should be kept sufficiently far from the wall to minimize interference effects.

Compressible Flow Considerations. Percent of the wind tunnel cross-sectional area blocked by the model is known to be a very important parameter in the case of compressible flow [28] [31]. This parameter can thus give the expert system a subgoal in establishing the model size. If an acceptable percent blockage is part of the knowledge base then, based on the physical parameters of the model and test objectives, the amount of blockage can be calculated.

Blockage is particularly important when designing to accommodate the tunnel transient. Documentation shows that the unobstructed area in the wind tunnel test section should be equal to or greater than the area of the second throat of the tunnel [28] [33]. Also, the yaw and pitch angle limits must be considered to calculate the true projected area of the model during testing. This projected area must not cause the tunnel to "un-start" and thus must be accounted for by the expert system. The tunnel will be started with

the model in an orientation for minimum projected area-zero yaw and pitch for an axisymmetric model.

Another important consideration for model size is the length of the model as it relates to the reflected shock wave [33] [35]. One method uses the Mach angle to calculate the downstream locations of the shock wave. The model should be at least four to five base diameters ahead of this reflected shock [27]. These analyses will be undertaken by the expert system to return model size advice to the project engineer.

Documentation Source Identification

As a result of a literature search and consultation with Dr. Lo and Dr. Vakili, several documents dealing with wind tunnel model sizing were found. Although not considered complete or comprehensive, the documents do provide sufficient information to start building the intelligent hypertext manual and its data bases. These documents are listed in the Bibliography under the "Example Aerospace Application" heading.

It is interesting to note that of two engineers with doctorate degrees who were consulted, both said they used experience and rules of thumb as opposed to documentation sources when designing and sizing wind tunnel models [33].

The integrated Intelligent Hypertext Manual made up of HyperCard, Guide and LEVEL5 follows the organization shown in Figure 13, Appendix 3. This integration follows the procedures outlined in Chapter 5. Control of the intelligent hypertext manual transfers from HyperCard to the other parts of the system. Two separate expert systems were developed for this thesis, a Model Size Document Location Expert System, and a Model Size Determination Expert System. They were constructed according to the guidelines of the Standard Knowledge Implementation Procedure.

The Model Sizing Hypertext Document

Once appropriate model sizing documents were located, a hypertext document was built using Guide and HyperCard according to the Standard Hypertext Implementation Procedure. The resulting organization of the hypertext data base is shown in Figure 15, Appendix 3.

Four document groupings were used for the hypertext document with an additional opening screen. This first display is a HyperCard card containing some explanatory information and links to the four hypertext documents. In short, these groupings are as follows:

1. Portions of "Model Selection and Design Practices Applicable to the Development of Specific Aircraft" using HyperCard [28].

2. "Model Size Selection in Wind Tunnel Testing," using a Guide hypertext file [35].
3. Short excerpts from several papers and textbooks that contain model sizing information and examples in a Guide hypertext file [30] [31] [32] [33] [27].
4. Excerpts from the "NASA Ames Aerodynamic Division Facilities" that list pertinent wind tunnel information to the model sizing problem using a Guide hypertext file [36].

The Documentation Location Expert System

The document selection expert system is activated when the user chooses the "Consult" button in HyperCard. Using all or part of the following information, appropriate sections of the hypertext data base can be displayed for the user: wind tunnel to be used, Mach number regime, type of test and model type. Once the Documentation Navigation Information is found the program returns to HyperCard where the user chooses a "Read Next" button which replaces the "Link Menu." The user is prevented from navigating to other sections of the hypertext data base until all the documents suggested by the expert system have been accessed.

The following discussion refers to Figure 16, Appendix 3, which shows the knowledge base organization for the expert system. It pursues the goal of "Card Name" (i.e.

which HyperCard card contains germane information). An excerpt of the knowledge base listing for this expert system is presented in Figure 11, Appendix 3.

First, the documentation location expert system seeks to determine the wind tunnel to be used for the test by asking the user for the tunnel name. If known, the Mach number capabilities of the specific tunnel are retrieved from the knowledge base. Using the Mach number regime alone, the expert system returns card names to the Document Navigation Information which is an ASCII text file. If the tunnel is not specified the system asks the user for the test Mach number to establish the Mach number regime.

The user is always given the choice of "unknown." If unknown, which is different from unspecified, the expert system will try to reach the goal by going to another grouping of rules as shown in Figure 16.

Next, the system seeks to determine what type of test will be run (e.g. force, pressure, flow visualization) and returns document locations based on this input. The program then inquires about the model type (e.g. "inlet" or "not inlet") and can return card names based on this information, if appropriate.

Last, this expert system displays the user inputs and asks if the user wants to run another expert system session or return to HyperCard to read the documents suggested. If the user wants to change his or her inputs by staying in the

expert system, new Document Navigation Information will replace the current file contents. Thus, the user will need to "consult," then return to the documents for each new set of inputs.

The Advisory Expert System

Model size criteria is returned to the user when he or she picks the "Model Size Advice" button. This expert system uses a knowledge base that tries to solve transonic and supersonic tests together, or subsonic tests separately. Like the document location expert system, this expert system uses, as input, the specific tunnel planned for the test. Other information that the knowledge base processes is the specific test Mach number and the maximum angle of attack or sideslip to be tested. To calculate model sizes the following information about the full-scale vehicle design are processed: projected area and planform area, wing span and length, and the location and height of the vertical tail or longest fins from the center line.

Once the model size criteria is found the expert system displays the information, writes it to an ASCII file, and returns to HyperCard where the user can also read the information if desired.

The following discussion refers to Figure 17, Appendix 3, which shows the knowledge base organization for the model size expert system. This expert system pursues

the goal of "Model Size" with the subgoals of "Model Length," "Projected Areas," "Planform Area" and "Model Scale."

If the user chooses subsonic testing the expert system returns a model size based on a maximum of 80 percent of the tunnel width. Tunnel specific limits that would dictate model length in an effort to keep the model in regimes of "good" flow away from boundary layers, and contracting or expanding streamlines.

For the tunnel transient the following equation based on isentropic relationships for air is used by the expert system to determine the maximum model area ratio [27]:

$$\frac{A_m}{A_w} = \frac{M'_{ch}}{\left[1 + \frac{(M'_{ch})^2 - 1}{6}\right]^3} \quad (3)$$

and

$$A_m = A_w - A_e \quad (4)$$

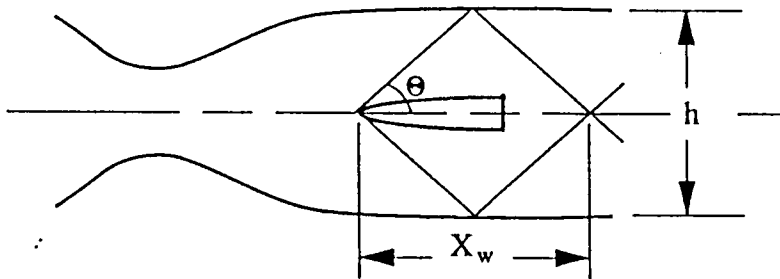
Thus the ratio of A_m/A_w can be calculated for a given apparent choking Mach number M'_{ch} of the flow ahead of the model. The expert system uses this Mach number as the actual test condition to calculate the area ratio. A_w is the area of the test section without the model. A_m is the area of the test section with the model installed. A_e is the effective area of the model taking into account the subsonic boundary layer around the model. In practice the true

projected area of the model is used and the area ratio is reduced by 20 percent to account for this boundary layer [28].

The projected area at the maximum angle of attack or side slip is then calculated and input into the area ratio equation to size the model so that the tunnel does not "unstart." This final calculation gives the resultant model area to tunnel ratio.

For the running tunnel the following equation gives the "reflection length," (x_w) based on the tunnel height (h) due to reflected shock waves [27] and as shown below:

$$x_w = h \cotan \theta = h \sqrt{M_\infty^2 - 1} \quad (5)$$



The above equation is based on the Mach angle and is an unconservative estimate [28]. The Mach angle is given as:

$$\theta = \sin^{-1} \left(\frac{1}{M_\infty} \right) \quad (6)$$

Consideration of the shock wave reflection from the subsonic boundary layer is also taken into account by reducing the actual tunnel height by twice the thickness of the boundary layer (δ) [33]. Use d in the wave reflection length calculation instead of h .

$$d = h - 2\delta \quad (7)$$

In practice the model should be 2 to 3 [28], or 4 to 5 [27] base diameters shorter than the reflected length. The expert system takes these rules into consideration and returns the model length to the user as a range of lengths. For two-dimensional wing models the reflected shock should intersect the model wake at least 50 percent of the chord length behind the wing [27]

Based on the Mach angle, a geometric check is performed by the expert system to ensure that fins or vertical tails do not protrude into the reflected shock wave [28]. If these appendages interfere with this shock, the model size is reduced accordingly. The final answers are compared to rules of thumb such as one to two and one-half percent blockage [31], and 1/50 scale models [33].

Finally any tunnel specific limits for the size of the model due to location restrictions within the wind tunnel test section are checked. If applicable the model size is reduced accordingly before display to the user.

Actual model dimensions are calculated based on the design of the full-scale vehicle, if input, and/or the knowledge base test section dimensions such as tunnel height. If the model is not representative of a full-scale vehicle then the projected areas, lengths and widths are returned based on the wind tunnel test section dimensions.

Test and Evaluation

The hypertext system is tested simply by navigating through it, at will, as a user would. All buttons (links) in each document were tested for correct operation. In addition, the intelligent hypertext manual was demonstrated to engineers for their feedback.

For the two expert systems different cases of input were given to LEVEL5 based on a checklist of all the possibilities and branches of logic that the expert system could follow. The checklist was then used to verify all of the possibilities that could be returned to the user. Model size calculations were verified with hand calculations of the equations used.

CHAPTER 7

HARDWARE AND SOFTWARE EVALUATION

Hardware Platform

Inclusive of the software packages chosen to implement an intelligent hypertext manual, the hardware for the system plays a vital role. Although the thrust of this thesis was to develop methodologies independent of hardware and software, experience garnered during the development process can help engineers decide what systems to use, if the choice is left to the knowledge engineer. This chapter compares different types of hardware on which the author has implemented and/or tested intelligent hypertext manuals.

Hardware considerations for the intelligent hypertext manual include memory requirements, clock speed, mass storage devices, video displays, scanners and printers.

Table 5 lists the different computer systems and their characteristics on which the intelligent hypertext manuals tested by this author reside. Software components of the intelligent hypertext manual were tested on a 10 megahertz 286 PC. This computer is much too slow and does not have enough memory to efficiently run an intelligent hypertext manual. It did prove valuable for benchmark analyses, however.

Table 5. Intelligent Hypertext Manual Computer Platforms.

IHM ⁹	Computer Platform	Op. Sys.	RAM Size (Mb)	Clock Speed (Mhz)	Disk Size (Mb)	Disk Access ¹⁰ (sec)
Benchmark	286 PC	DOS 3	0.64	10	20	14
PEIA (I)	386 PC	Window	4	25	80	16
PEIA (II)	Mac IIX	Mac	4	20	80	8
ICCRM	Sun 386i	UNIX	8	25	327	1
HIHM	386 PC	Window	6	16	80	14

Random Access Memory and Clock Speed

Large memory requirements hold true for the intelligent hypertext manual executing under control of a multi-tasking operating system. Table 6 shows the random access memory requirements for different components of the intelligent hypertext manual. This table does not include the memory needs of the hypertext data base and expert system knowledge

⁹PEIA: Project Engineer's Intelligent Assistant.
 ICCRM: Intelligent Computer Codes Reference Manual.
 HIHM: Hazardous Gas Intelligent Hypertext Manual--only hypertext data base partially developed.

¹⁰For 1 megabyte read and write operation.

base, so total memory requirements listed are thus less than actual minimum memory needs.

Table 6. Memory Requirements for Expert System Shells and Hypertext Programs Used in Intelligent Hypertext Manuals.

IHM	Operating System (kBytes)	Expert System (kBytes)	Hypertext Program (kBytes)	Total Memory (kBytes)
PEIA (I)	Windows 226	VP-Expert 189	Guide 2.0 220	635
PEIA (II)	Macintosh 160	LEVEL5 341	Guide 2.0 192	693
ICCRM	UNIX 400	EXSYS 192	KMS 964	1556
HIHM	Windows 226	LEVEL5 Ob 640	Guide 3.0 111	977

Knowledge bases and textual information need relatively small amounts of memory. Text-only data bases require just one byte per character. Graphic files take considerably more memory. A full-page bit-mapped image (8x10 inches) at 300 dots per inch resolution occupies 1.1 megabytes.

Loading files and programs into memory cuts execution times. On a 286 PC, Windows took four seconds to load and

start when resident in memory as compared to nine seconds when started from the hard disk drive. Table 7 shows the load and start times for key actions taken by intelligent hypertext manual components run on the ten megahertz 286 PC. RAM indicates that the program resides in memory on a virtual disk. The table clearly shows the advantage of keeping programs and files in memory.

If adequate memory is available, the most important performance parameter becomes faster clock speeds.

Peripherals

Mass Storage. After clock speed, mass storage access time most affects the intelligent hypertext manual performance. Magnetic media still offer the quickest access to files with the best access times below 10 milliseconds.

Large intelligent hypertext manuals will require high-capacity mass storage devices. New technology hard disk drives and optical storage devices possess the capacities to satisfy these requirements. Hard disk drives with hundreds of megabyte capacities are available. Mass storage on optical devices now approaches one billion bytes. BYTE gave a good survey of optical mass storage devices [7].

Table 7. Software and File Loading Times for Intelligent Hypertext Manual Components.

Start Point	Called Object	Load Display Time (secs)
DOS (RAM)	Windows from RAM	4
DOS	Windows	9
Windows (RAM)	Guide from RAM	9
Windows	Guide	16
Guide (RAM)	VP-EXPERT from RAM	12
Guide	VP-EXPERT	10
Guide (RAM)	Guide File from RAM	10
Guide	Guide File	21

Two types of graphics files were evaluated for disk storage requirements: the Sun graphics image format, and bit-mapped files. As seen in Table 8, bit-mapped images require the most memory. Text data in ASCII format occupies one bit per character. The size of files using the KMS and Guide hypertext formats were determined. The assessment showed that text storage requirements will not be a problem for an intelligent hypertext manual.

Table 8 also lists typical file storage requirements. Based on these amounts and using average word counts, a 100,000 word Guide document, with links, will occupy about three megabytes of storage. A 100,000 word file is quite large (e.g. the entirety of this thesis is about 20,000 words).

To reduce graphic file storage size, the UNIX operating system offers a compress program which gave significant reductions in graphic file size. "Shareware" programs, PK-Pak and PK-Unpak from PK Ware offer relatively good file compression for PC files at little expense. Table 9 lists a sample of file sizes uncompressed and compressed.

Table 8. Typical File Storage Requirements for Data Base
Files of an Intelligent Hypertext Manual.

File Type	Program Type	Storage Req. (bytes/char)
Executable ¹¹	BASIC	1
ASCII Text	Editor	1.0
Formatted Text	Word	1.5
Hypertext ¹²	Guide	2.3
"	HyperCard	~5
		(bytes/Sq. in)
Graphic ¹³	Bit-map	13680
"	Sun Image	810

¹¹For interpretive programs like BASIC.

¹²Based on average samples taken from files of many links, a few links and no links.

¹³Using 300 dots per linear inch resolution.

Table 9. Uncompressed and Compressed File Sizes Using PK-Pak and PK-Unpak File Compression "Shareware."

File Type	Uncompressed (bytes)	Compressed (bytes)	Savings (bytes)	%
ASCII Text	158,627	73,335	85,292	54
Formatted Text	199,680	95,016	104,664	52
Hypertext	243,779	107,096	136,683	56
Bit-Mapped	113,158	4535	108,623	96
Executable	402,048	267,440	134,608	33

The greatest savings are for bit-mapped graphic files and hypertext files, the type of files stored in hypertext data bases. The least amount of storage saved was a full third which is also significant.

Monitors. The trade-off for monitors is resolution versus speed. Higher resolution monitors are the best for readability but take longer to display. For fast computers (20 megahertz or higher) the redraw speed for graphics screens is acceptable with high-resolution monitors. Of the computers tested the 16 megahertz 386 PC, and 10 megahertz 286 PC significantly impacted the display of a graphics screen while using high-resolution monitors. The SUN 386i

19-inch monitor displayed two full pages of text or one large graphic.

Scanners. An assessment of optical scanners as rapid input devices was conducted as part of preliminary work for the Hazardous Gas Intelligent Hypertext Manual [4]. For graphics, flat-bed scanners scan the best images because the input image is held fixed. Bin-fed scanners offer the fastest throughput since an operator does not need to load individual pages. When developing large hypertext data bases both figures and text will need to be input. The knowledge engineer must determine if throughput or image quality is the most important factor. Ultimately, the decision will be an economic one based on personnel time. An adequate scanner with optical character recognition software can be bought for less than \$5,000. MacUser had a recent article that surveyed and tested a variety of scanners [9].

Printers. Printers are not an important part of the intelligent hypertext manual. For full-blown development of an intelligent hypertext manual, a printer will be helpful during the development effort and for users who want to obtain hard copies of important information.

As stated previously, multi-tasking environments for full-blow intelligent hypertext manuals are a must. UNIX on the SUN, DOS with Windows on various PCs, and the Macintosh Operating System on the Macintosh were tested.

The UNIX system offers the best multi-tasking capabilities among the three tested. Sun integrated its Sun View Desktop with UNIX to provide a graphical user interface. UNIX gives the user a DOS window for running DOS applications. This feature made the SUN very versatile.

Windows 3.0, released in 1990, gives the DOS users their first functional multi-tasking environment. However, to take full advantage of Windows for intelligent hypertext manual development a high-end 386 PC should be employed.

The Macintosh System was the first widely-accepted graphical user interface. It still leads Windows and Sun View in ease of use and intuitiveness. However, the multi-tasking of Macintosh is not as good. The latest System 6.0.5 still has problems that cause applications to fail without explanation. Macintosh offers the most intuitive environment for the user and Windows 3.0 now runs a close second.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

The Intelligent Hypertext Manual Methodologies developed for this thesis give knowledge engineers a preconceived structure and process for implementing intelligent hypertext manuals. The project clearly showed the possibility of implementing an intelligent hypertext manual using the Intelligent Hypertext Manual Methodologies.

Although the wind tunnel model size determination application intelligent hypertext manual uses limited hypertext data bases and limited rules, its successful implementation shows the concepts developed herein are viable. It was found that giving direct advice to the user about the system is equally as important as returning the appropriate documentation. Thus, the goals of the intelligent hypertext manual expert system should include advisory as well as document location information.

The effort in building the model sizing expert system underscored the fact that the domain must be highly restricted for functional systems to mature. Numerous choices (independent variables) were applicable and were adequately handled for the implementation.

The most important development for this work lies in the development of the Standard Hypertext Implementation Procedure. This procedure satisfied the need for controlled

flexibility when implementing new applications into the intelligent hypertext manual.

Another very important result was that Document Location Information for the hypertext data base can be stored outside of, or as part of the knowledge base for the expert system. The Document Location Information provides the key for passing information between several types of hypertext programs and expert systems shells. The Document Navigation Information, stored as a separate file and based on the Document Location Information, allows the hypertext program to navigate based on the expert system output. These two information files are paramount to the success of the Integrated Intelligent Hypertext Manual Methodologies.

Script languages are a necessity for the successful development of intelligent hypertext manuals. If a script language is not available, then some pioneering work will again be needed to form the link to the expert system.

The effort to keep the Intelligent Hypertext Manual Methodologies hardware and software independent was accomplished because it was used for a computer system that is a new platform for intelligent hypertext manuals.

The knowledge acquisition process is fundamental to successful intelligent hypertext manual construction. Additional work on how to conduct interviews and extract the most important information from the expert is necessary.

Fast high-end hardware with lots of memory, and large, fast mass storage devices will give developers and users the best intelligent hypertext manual. Loading the programs into memory gives load and execute time-savings near 50%. The absolute minimum random access memory available for the multi-tasking environment of intelligent hypertext manuals should be at least two megabytes. Four megabytes is a better lower limit. The most important aspect of mass storage devices is access speed. An increase in hard disk access speed will make much more difference than a similar increase in clock speed.

For further study, a logical extension to the effort would be to devise a system that takes artificial intelligence/expert systems to automatically build a hypertext manual from linear documents.

In conclusion, if a user feels as comfortable with the intelligent hypertext manual as with a printed document then the design may be considered adequate.

BIBLIOGRAPHY

BIBLIOGRAPHY

General/Intelligent Hypertext

1. Bush, Vannevar, "As We May Think," The Atlantic Monthly, July, 1945.
2. Hoyt, W. Andes, "Development of a Prototype Intelligent Hypertext Manual," The AIAA Southeastern Regional Student Conference, Atlanta, Georgia, April 1990.
3. Lo, Ching F. and Steinle, F. W., "Application of Intelligent Systems to Wind Tunnel Test Facilities," AIAA Paper No. 88-0193, January, 1988.
4. Lo, Ching F., and Garrison, George W., Development of an Intelligent Hypertext Manual for the Space Shuttle Hazardous Gas Detection System, Proposal, Center for Aerospace Propulsion, The University of Tennessee Space Institute, May, 1990.
5. Lo, et al, "Final Report (Phase I) Intelligent Hypertext Systems for Aerospace Knowledge Representation," Engineering Research & Consulting, Inc., Tullahoma, Tennessee, 1990.
6. Rada, Roy and Barlow, Judith, "Expert Systems and Hypertext," The Knowledge Engineering Review, December 1988.
7. Ruzzo, John, et al, "Erasable Optical Drives, MacUser, Foster City, California, November, 1990.
8. Wu, Y. C. L. Susan, "Intelligent Hypertext Systems for Aerospace Knowledge Representation," NASA SBIR Solicitation 88-1-II 0309, Engineering Research & Consulting, Inc., 1989.
9. "Scanners Build a Better Image," PC Magazine, March 1989.

Hypertext

10. Conklin, Jeff, "Hypertext an Introduction and Survey," Computer, September, 1987.
11. Fiderio, Janet, "A Grand Vision," BYTE, "Special Issue--Hypertext--In-depth Review," October 1988.

12. Guide Hypermedia Information System, Version 3.0, OWL International, Inc., Bellevue, Washington, 1990.
13. Guide Hypermedia Information System, Version 2.0, OWL International, Inc., Bellevue, Washington, 1989.
14. KMS Action Language, Knowledge Systems, Inc., Murrysville, Pennsylvania, 1988.
15. KMS Reference Manual, Knowledge Systems, Inc., Murrysville, Pennsylvania, 1988.
16. Macintosh HyperCard User's Guide, Apple Computer, Inc., Cupertino, California, 1989.

Expert Systems and Shells

17. Buchanan, B. and Shortliffe, E., Rule-Based Expert Systems: The MYCIN Experiments, Addison-Wesley, Reading, Maine, 1984.
18. Friederich, Sylvia and Gargano, Michael, Expert System Design and Development Using VP-Expert, John Wiley & Sons, New York, New York, 1989.
19. Gevarter, William B., "The Nature and Evaluation of Commercial Expert System Building Tools: Revision 1," NASA TM 88331, 1987.
20. Montgomery, Peter A., "A Prototype Expert System for Performance Evaluation of Liquid-Fueled Rockets," The University of Tennessee, Knoxville, Tennessee, May, 1990.
21. Moose, Anne and Schussler, Terry, VP-Expert, Addendum, Release 2.0, Paperback Software International, Berkeley, California, 1988.
22. Moose, Anne and Shafer, Dan, VP-Expert, Paperback Software International, Berkeley, California, 1987.
23. Parsaye, Karman and Chignell, Mark, Expert Systems for Experts, John Wiley & Sons, Inc., New York, New York, 1988.
24. Pedersen, Ken, Expert Systems Programming, John Wiley & Sons, New York, New York, 1989.

25. EXSYS, Version 3, "Expert System Development Software Demo Manual," Exsys, Inc., Albuquerque, New Mexico, 1986.
26. LEVEL5, Information Builders, Inc., New York, New York, 1989.

Example Aerospace Application

27. Allen, H. Julian and Spiegel, et al, Joseph M., High Speed Problems of Aircraft and Experimental Methods, "Wind Tunnel Measurement," Princeton University Press, 1961.
28. Beale, Donald D., "Model Selection and Design Practices Applicable to the Development of Specific Aircraft," National Advisory Committee For Aeronautics, (AGARD R-21), Washington, DC, 1956.
29. Griffin, Stanley A., "Model Systems and Their Implications in the Operation of Pressurized Wind Tunnels," AGARD-CP-174, Neuilly Sur Seine, France.
30. Griffin, Stanley, A., et al, Design of Advanced Technology Maneuvering Aircraft Models for the National Transonic Facility, General Dynamics Convair Division, San Diego, California.
31. Kraft, E. M., "An Overview of Approaches and Issues for Wall Interference Assessment/Correction," NASA Conference Publication 2319, 1983.
32. Pope, Alan and Harper, John J., Low-Speed Wind Tunnel Testing, John Wiley & Sons, New York, New York, 1966.
33. Pope, Alan and Goin, Kenneth L, High-Speed Wind Tunnel Testing, John Wiley & Sons, New York, New York, 1965.
34. Shi, George and Vakili, Ahmad, Private Conversations with the Author, November, 1990.
35. Ulbrich, Norbert, Model Sizing, AE513 Term Paper, University of Tennessee Space Institute, Tullahoma, Tennessee, 1990.
36. NASA Ames Research Center, "Users Guide for Aerodynamics Division Facilities," Moffett Field, California, February, 1987.

APPENDICES

APPENDIX 1

HYPertext

General Description and Components

To help the reader understand the implementation methodologies constructed for the intelligent hypertext manual, a brief developmental review and explanation of hypertext follows. The standard implementation methods appear in Chapters 3 through 5.

Although the idea of hypertext originated in 1945 [1], commercial realization of the concept did not occur until the 1980s¹. Current hypertext software enables access to information via links which reference specific addresses in a data base. This association of information using links removes the limits of obtaining data in a sequential or linear method.

When a user reads a hypertext data base he or she goes through it much as a reader would refer to specific yet separated pages in a book using the table of contents and index. Hypertext programs use sensitive areas, "hot spots," on the computer screen which are activated via a mouse. The hot spots contain links to the data base which when "clicked on" result in a display of the "linked" information. A key advantage of hypertext is that each piece of information in the data base can be accessed through multiple links, i.e. from different parts of the data base itself.

¹Vannevar Bush, science advisor to President Harry S. Truman, envisioned a machine that could access diverse blocks of information in a non-sequential matter similar to the process of the human brain.

One of the chief disadvantages of hypertext comes from the user getting lost in the data base, especially when the reference documents are lengthy. In addition to helping the user quickly retrieve information from the data base, the intelligent hypertext manual will help the user find his location, or to return to familiar territory.

Researchers have tried to establish mathematical models of hypertext data bases to understand their relationships [6]. The hypertext document is viewed as a sextuple $\langle N, C, W, B, M_1, M_d \rangle$ where:

N is a Petri net,

C is a set of document contents,

W is a set of windows,

B is a set of buttons (links),

M_1 is a logical projection for the document, and

M_d is a display projection for the document.

This model entails all of the possibilities that can be used, mathematically speaking, to build a data base. Even so, hard conclusions about implementing hypertext based on this sextuple are hard to reach, if not unreachable. However, this model does suggest some sort of upper limit on the bounds of the hypertext construction process. Albeit an unmanageable upper limit.

Data Bases and Hypertext Links

When reading printed books the reader reads sequentially or by skipping from page to page. Often, the user skips pages in a haphazard manner. Other times, the reader uses reference pages, such as the table of contents or index, to skip pages and find the information of interest. In all cases the reader must physically provide the linking mechanisms. A linking action taken by a reader might be to look at the table of contents and then to turn to the appropriate page.

Similarly, hypertext enables documents to be structured in three forms: hierarchic, cross-referenced and a combination of the two. Active links between parts of the hypertext data base provide the key to rapid document retrieval. The data base contents can include text, graphics, and even audio and video information. Any bit of information from a single letter to a point in a drawing, could conceivably become a link.

Hierarchies. Printed documents naturally follow hierarchic structures with a top-down arrangement (Figure 18, Appendix 3). Hierarchic structures for hypertext data bases provide a natural means for relating groups within a document. Outlines illustrate another example of hierarchic structures. Each box in the figure represents distinct parts of the document (nodes). If it is a hypertext

document the branches leading to the nodes form the links. The computer displays a node when the link to it is activated.

In a strict hierarchic structure each node possesses, at most, three types of links (Figure 18, Appendix 3). These links lead up to a higher level (step up), lead down to a lower level (step down) or lead to sections on the same level (jump). The top-most section of the hierarchic data base, usually considered the user's default entry point, will have only one link that leads down into the data base (e.g. from a table of contents node).

Cross-Referenced. With hypertext a cross-referenced data base consists of information stored on equal levels with multiple links between the nodes (Figure 19, Appendix 3). The nodes of a this type of data base become the display as do the hierarchic nodes. Cross-referenced data bases accommodate the greatest amount of flexibility to the hypertext builder. However, they present the most likely data base in which a user gets lost.

Cross-referenced links, tying cross-referenced nodes together (Figure 19, Appendix 3), are very simple links. With cross references each node can receive virtually an unlimited number of links. For instance, every node in the data base may have a link to the table of contents, index, references, nomenclature and glossary.

Combination. The combination or mixed-structure data base, as the name implies, uses both the cross-referenced and hierarchic structures. Two types of combination data bases are possible. One uses a hierarchic data base with cross references between the hierarchic nodes (Figure 20, Appendix 3). The other employs a cross-referenced data base with hierarchies added at some or all of the nodes to take the user to different levels (Figure 21, Appendix 3).

A hierarchic structure with cross-referenced links allows the mixed structure data base its great versatility and flexibility. For example, a hierarchic data base could contain cross references to the top-most (entry default) level from all of its nodes (Figure 20, Appendix 3). These types of links were used in both the Intelligent Computer Codes Reference Manual [2] and the Project Engineer's Intelligent Assistant [5].

User Interface and Operating System

Each hypertext program utilizes a user interface which must be comfortable and familiar to the user. Hypertext programs employ the capabilities of a mouse for pointing and clicking to invoke actions. The keyboard serves as a device to input data and supplement command entry.

All hypertext programs reviewed operate under control of a graphical user interface (GUI) which offers the best interface for today's computers and applications. Graphical

user interfaces provide intuitive screens, prompts and object oriented access to the operating system and applications. Windows, X Windows, and the Macintosh Desktop run on the PC/AT compatible, Sun and Macintosh computers respectively.

Common Hypertext Characteristics

In the most basic sense, all hypertext programs possess the following common characteristics:

1. Ability to store separate but related bits of information.
2. Links between these bits of information.
3. User interface.

In addition, to implement a hypertext document the program must also provide a linking facility and text editor to the author. To classify as a hypertext program the linking facility and editor are not necessary if the end-user does not intend to modify the document (data base). Other features commonly found in commercially available hypertext programs include help facilities, graphics editors, scripting languages, external links to other programs, graphical menus and browsers.

Selected Hypertext Software Survey

A survey of representative hypertext programs serves to provide an understanding of what methodologies will lead to

efficient construction of the data base, regardless of the software applied (Table 10). Complete surveys of hypertext packages are available [10] [11]. Typical hypertext displays from the prototype aerospace implementation, model sizing, are shown in Chapter 3.

Table 10. Hypertext Software Surveyed.

Software	Ver.	Computer	Operating System
Guide	2 & 3	PC, Macintosh	GUI [13] [12];
HyperCard	1.2.5	Macintosh	Macintosh [16];
KMS	6b	Sun 386i	Unix [15].

Guide

The authors of Guide constructed a hypertext package that originally aimed to keep the user from feeling lost in a hypertext document [6]. Thus, two features of Guide, replacement buttons and note buttons give the user access to information without leaving familiar surroundings.

Replacement buttons substitute an existing piece of text/graphics with linked information. In essence, the new information is inserted between the point immediately ahead of and following the replacement button. The note button

opens a small window on the screen that contains related information. This new window stays open only so long as the user holds down the mouse button. These two buttons accomplish the original goal of keeping the user in familiar territory. Yet, they are unique and thus not included in the Standard Hypertext Implementation Procedure (Chapter 3).

Guide incorporates a reference button which provides links (cross-referenced) to additional information in two ways. First, the reference button links the document to another location in the same node (file). Second, the reference button links the current file to a new file which opens in a new display window.

Guide 3.0 contains a powerful scripting language, LOGiiX, for automating actions and commands in the hypertext document. With LOGiiX, or external command buttons, outside programs can be executed. LOGiiX is not available for the Macintosh version of Guide.

Guide provides the developer with powerful text editing and formatting capabilities in a "what-you-see-is-what-you-get" (WYSIWYG) environment. On the PC, graphics editing is limited to resizing and cutting parts of pictures. For the Macintosh, Guide comes with a desktop accessory, Scribbler, which gives the user good graphics editing tools.

HyperCard

Apple supplies HyperCard with all Macintosh computers at no extra charge. HyperCard version 1.2.5, like Guide and KMS, makes possible links between any two parts of the data base. It includes a scripting language, HyperTalk. This scripting language, provides access to other programs and files. Therefore, HyperCard can act as a control program for an intelligent hypertext manual.

HyperCard has a default start point, the "Home Card." Only defined buttons can act as links--as opposed to any single character or graphic element in most hypertext systems. HyperCard stores its data bases in stacks of sequential cards. However, imaginative use of linking can form hierarchic and cross-referenced documents.

A full featured bit-mapped paint program is included with HyperCard which gives it good graphics editing capability. HyperCard exploits the Macintosh user's interface which provides the author with what-you-see-is-what-you-get text editing and formatting capabilities. However, each field and button must contain the same format within its confines.

HyperCard can also access the small computer standard interface (SCSI) ports on the Macintosh which allows full featured multi-media control.

A serious drawback of HyperCard is its screen display size limitation. The program was originally designed for

the nine-inch Macintosh screen, thus cards can be no larger than the original Macintosh screen. Only one window at a time can be displayed.

Knowledge Management Systems (KMS)

KMS is a UNIX-based hypertext program that also supports text and graphics in a what-you-see-is-what-you-get environment. Version 6b is reviewed here. Version 8 is the latest. The best description of KMS (Knowledge Management System) comes from the "Introduction to KMS" in the Users' Manual [15].

"KMS is a distributed hypertext system for supporting collaborative work. With KMS you can easily create and manage knowledge at all levels of your organization--from your personal notes and documents, to your work group to your entire organization. KMS implements hypertext through frames of information. Each frame is uniquely named by the program. All information must be structured to fit into frames. KMS displays up to two frames simultaneously."

Any item of a frame may be linked to another frame in any arrangement. Items have attributes which include formatting, link locations and an "action" attribute that is invoked when the user clicks on it. Actions are very powerful scripts written in the KMS Action Language [14].

With KMS users can edit frames directly. It offers good text editing and formatting capabilities. KMS limits graphics editing to drawing and resizing simple lines, curves, boxes and ellipses. Enclosed areas may be shaded and/or filled.

One of its strengths is its ability to access and display frames in less than one second, on average. Data bases may be distributed across file servers on a network.

Analysis and Comparison

KMS is this author's hypertext program of choice. However, a developer can successfully build an intelligent hypertext manual using any of the three hypertext programs reviewed here. Each has its weaknesses and strengths.

In HyperCard, individual words and graphic elements cannot be linked to the data base. Only separate buttons can be linked to the data base. This fact proved cumbersome. Any time a link was needed in the body text, a transparent button had to be built on top of the word that represented the link. If the text containing the linked word was moved or edited the linked word would no longer lie under the button. HyperCard offers the best capabilities for graphics integration.

The KMS Action Language proved to be very powerful. Similar capabilities were found in Guide on the PC and HyperCard on the Macintosh. The Guide LOGiiX language was

the most difficult to use. Using its Action Language, KMS offers the developer a powerful automatic indexing capability for rapid navigation through the hypertext data base. The index feature gives KMS manuals a definite search/navigation advantage over Guide and HyperCard.

Guide provides three types of links to the data base as opposed to the single type of KMS and HyperCard links. Guide has expansion buttons, note buttons and references which provide the greatest flexibility. HyperCard emulates pop-up notes with the use of scripts. Pop-up notes offer users quick access to definitions, acronym meanings and short reminders. These notes eliminate the need to navigate away from the current frame when only small bits of information are desired. This feature was not used for the Standard Hypertext Implementation Procedure.

The KMS developers think the idea of a single type of data storage unit, frames, is the best storage method, although this may not be the case. KMS offers only one type of link which can be shared over directories and networks.

KMS offers the most powerful features for formatting and displaying information. The data base access performance of KMS on the Sun 386i exceeds that of Guide running on a 386 PC. HyperCard was fast at data retrieval but slow at printing.

APPENDIX 2
EXPERT SYSTEMS

General Description and Components

Expert systems grew out of early artificial intelligence (AI) work. Although expert systems do not embody all the characteristics of artificial intelligence, they do contain knowledge and thus "intelligence." An expert system may be loosely defined as a computer program that embodies knowledge and helps humans perform tasks usually reserved for accomplished experts.

Two components make up the core of an expert system--an inference mechanism and a knowledge base. Included in these is a user interface for both the end user and developer. The most common elements that embody full-featured expert system shells are as follows: knowledge base, inference engine, editor, explanation facility, and user interface. Sessions with expert systems are called consultations.

Knowledge Bases

"Knowledge is information about the world that allows an expert to make decisions [23]." Since knowledge is used for reasoning by the expert system it is somewhat different than data contained in "number crunching" computer programs. The knowledge must be stored in structures that enable easy comparisons. Typical structures include facts (e.g. the world is round) and rules (e.g. "If" the world is round "Then" we can fly around it). The fact is a simple statement about observations made by humans. Rules contain

"If/Then" statements which prove or disprove additional facts about a subject.

Knowledge bases can be built using "If/Then" rules and facts. Frames, which hold portions of information items that are related, are another method of storing knowledge. Other methods of implementing knowledge include semantic networks and predicate logic. This section only addresses expert systems which store knowledge using rules and facts.

Inference Engines

Inference mechanisms, called inference engines, use symbolic based computer programs to make relational comparisons. Inference engines use knowledge bases and input from an external source, e.g. the user, to reach conclusions. The engine tries to establish whether or not certain statements are true or false. These statements are referred to as goals.

Inference is accomplished by one of two methods or a combination of the two and is referred to as chaining. Forward chaining starts with facts and searches all the information in the knowledge base to see if some hypothesis (goal) can be deduced to be true. Backward chaining, however, starts with a hypothesis (goal) and sees if the facts in the knowledge base support it as being true.

"Rule firing" applies to which rules in an expert system produce an action. In backward chaining only rules

that can assign a value to a goal (variable) are considered. If premises cannot be immediately proved or disproved, the inference engine establishes a sub-goal and begins operation trying to find whether this new goal is true or not. The procedure continues, firing rules with true premises, until all sub-goals and goals are found. In forward chaining all forward chained rules are considered and if the rule premise is found to be true the rule fires.

When needed, the engine will ask the user for input if it cannot find the information needed to satisfy rules. Inference engines can combine knowledge and facts to share new facts about related objects. This method of sharing information is called inheritance.

Editor

The editor provides the knowledge engineer with a tool for implementing the expert system. An editor for rule-based knowledge bases enables the developer to input the rules, facts and goals for a particular domain. As the developer builds rules for the expert system he or she is also building the knowledge base. In addition, some editors furnish the developer with ways of formatting displays.

Explanation Facility

Explanation facilities are used to answer two questions that a user may ask the expert system, and to give

additional information about a conclusion. The two questions that the explanation facility answers are, "Why does the expert system need a specific input from the user?" and "How did the expert system reach its conclusion?" If the user is prompted for input during a consultation and does not know what response is required, then the user may ask, "Why"? If users want to know how a conclusion was reached, they may ask, "How"? The explanation facility will list the rules and facts used to reach a conclusion. Once a conclusion is reached, the explanation facility may provide the user more information about the conclusion.

User Interface and Operating System

The user interface for expert systems is commonly known as the "consultation screen." This display gives information to the user about the expert system in use and prompts the user for input when needed. The specific "look and feel" of consultation screens are as different as the varied expert system shells now available. Some are very simple textual displays while others take full advantage of graphical user interfaces.

The multitude of available expert system shells assures the intelligent hypertext manual developer that the system can be implemented on just about any operating system specified by the end user.

A survey of representative shells illustrates their characteristics in an effort to understand what methodologies will lead to efficient construction of the system regardless of the software applied. More complete surveys of expert system packages are available [19] [24]. The three software packages surveyed are listed in Table 11. Figures 11 and 12, Appendix 3, show rules and a typical consultation display from LEVEL5 for the prototype aerospace implementation, wind tunnel model sizing.

Table 11. Expert System Shells Surveyed.

Software	Ver.	Computer	Operating System
EXSYS	2.0	PC compatible	MS-DOS [25]
ProEXSYS	3.0	"	"
LEVEL5		Macintosh	Apple [26]
LEVEL5 Object	1.0	PC compatible	MS-DOS
VP-Expert	2.1	PC compatible	MS-DOS [22]

EXSYS and ProEXSYS

The prototype Intelligent Computer Code Reference Manual [2], as mentioned in the Introduction, utilized EXSYS for its expert system shell. The users' manual introduction provides a good description of this program [25]:

"EXSYS is a generalized expert system development package. . . Expert systems developed with EXSYS will ask the user questions relevant to a subject. The user answers by selecting one or more answers from a list or by entering data. The computer will continue to ask questions until it has reached a conclusion. The conclusion may be the selection of a single solution or a list of possible solutions arranged in order of likelihood. The computer can explain, in English, how it arrived at its conclusion and why."

EXSYS employs If-Then-Else type rules. The inference engine determines what information is needed and the best method to get the information. The engine attains information from rules before asking the user. EXSYS utilizes a backward chaining inference engine.

EXSYS provides the developer with an editing program for developing and maintaining the knowledge base. Its formatting features are quite limited. The latest version does implement limited hypertext capabilities.

VP-Expert

VP-Expert is a low-cost expert system shell that contains forward and backward chaining, explanation and help facilities, an editor, and rule tree graphic display.

It was used for the Project Engineer's Intelligent Assistant Phase I project [5] and served as the primary user interface for this system. The system utilized a VP-Expert command to call outside programs, in this case Guide, which served as the hypertext program. VP-Expert did not provide scripts for the Phase I project but simply "launched" applications. The launched applications were DOS batch files which then called Guide.

VP-Expert, in its latest version, offers hypertext capabilities within the expert system shell. VP-Expert, version 2.1, was used for the thesis, A Prototype Expert System for Performance Evaluation of Liquid-Fueled Rockets [20]. That work employed the hypertext capabilities of VP-Expert but the implementation did not utilize intelligence in referencing the hypertext data base.

LEVEL5

LEVEL5 runs on the Macintosh, PC and includes a mainframes version. It is being used as the shell for the Project Engineer's Intelligent Assistant [8]. LEVEL5, like EXSYS and VP-Expert, uses backward chaining to reach conclusions. Its knowledge base also uses If-Then rules.

Of the three shells surveyed only LEVEL5 operates under a graphical user interface. It was the only shell that required the knowledge base to be compiled for execution. LEVEL5 provides a good explanation and help facility similar to the other shells.

Analysis and Comparison

EXSYS provides the best development structure.

For the user, an improvement to EXSYS would be a rapid start up mode. A half-dozen keystrokes and screens must be worked through before meaningful consultations begin. VP-Expert provides statements to quickly get the consultation up and running.

A new version of LEVEL5, LEVEL5 Object, was recently introduced for the PC. It was tested on a 286 AT computer. On this machine LEVEL5 Object was excruciatingly slow. For low-budget implementations, LEVEL5 Object would not be acceptable since it is the highest priced shell reviewed and should operate on high-end 386-based computers for the best execution speed.

For the intelligent hypertext manual prototype on the Sun, EXSYS ran in a DOS window. On the PC, VP-Expert operated outside of Windows. Thus, these two shells considerably slowed the execution of the intelligent hypertext manual since the computer "quit" the graphical user interface to run the expert system.

APPENDIX 3

FIGURES

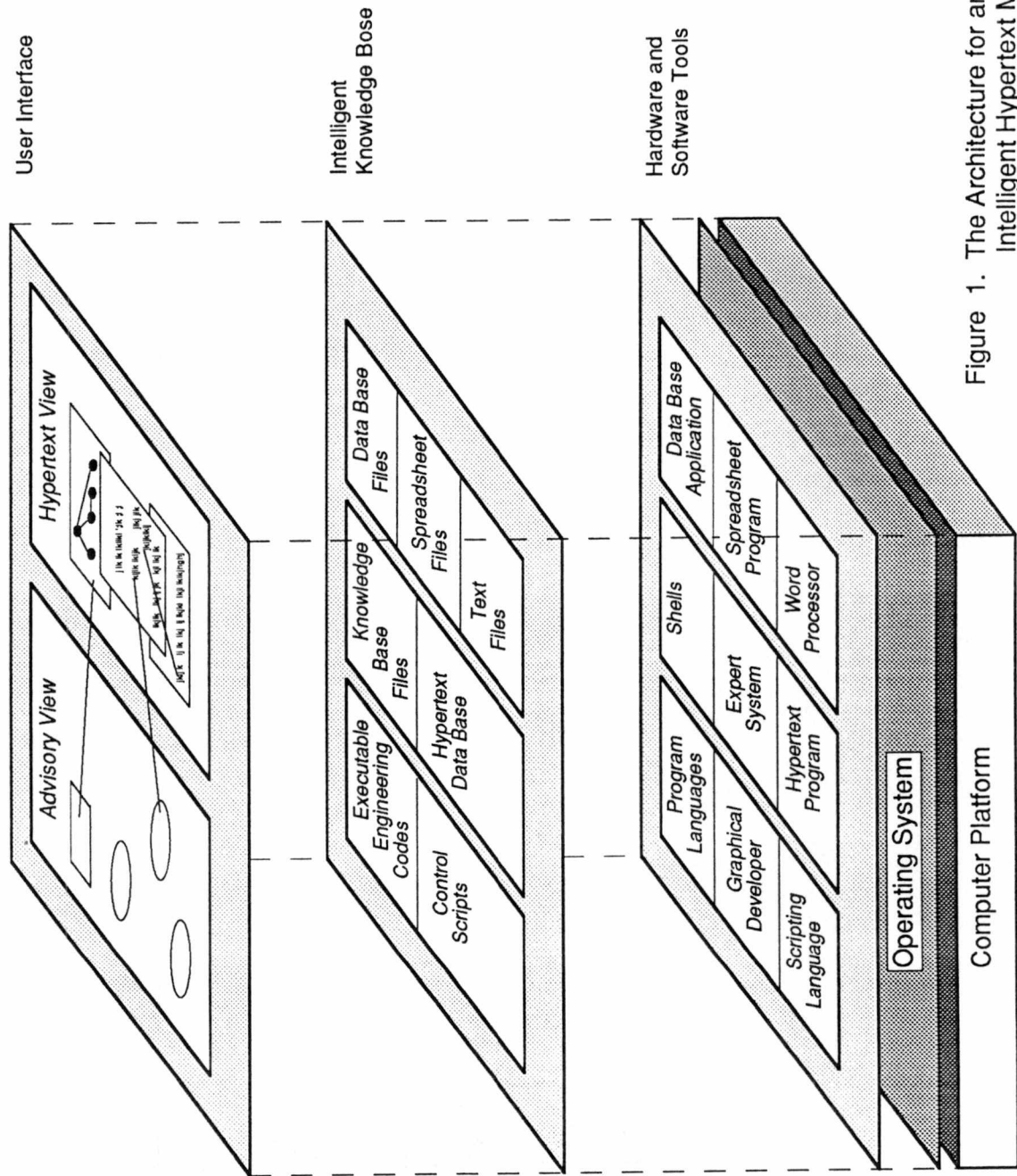


Figure 1. The Architecture for an Intelligent Hypertext Manual.

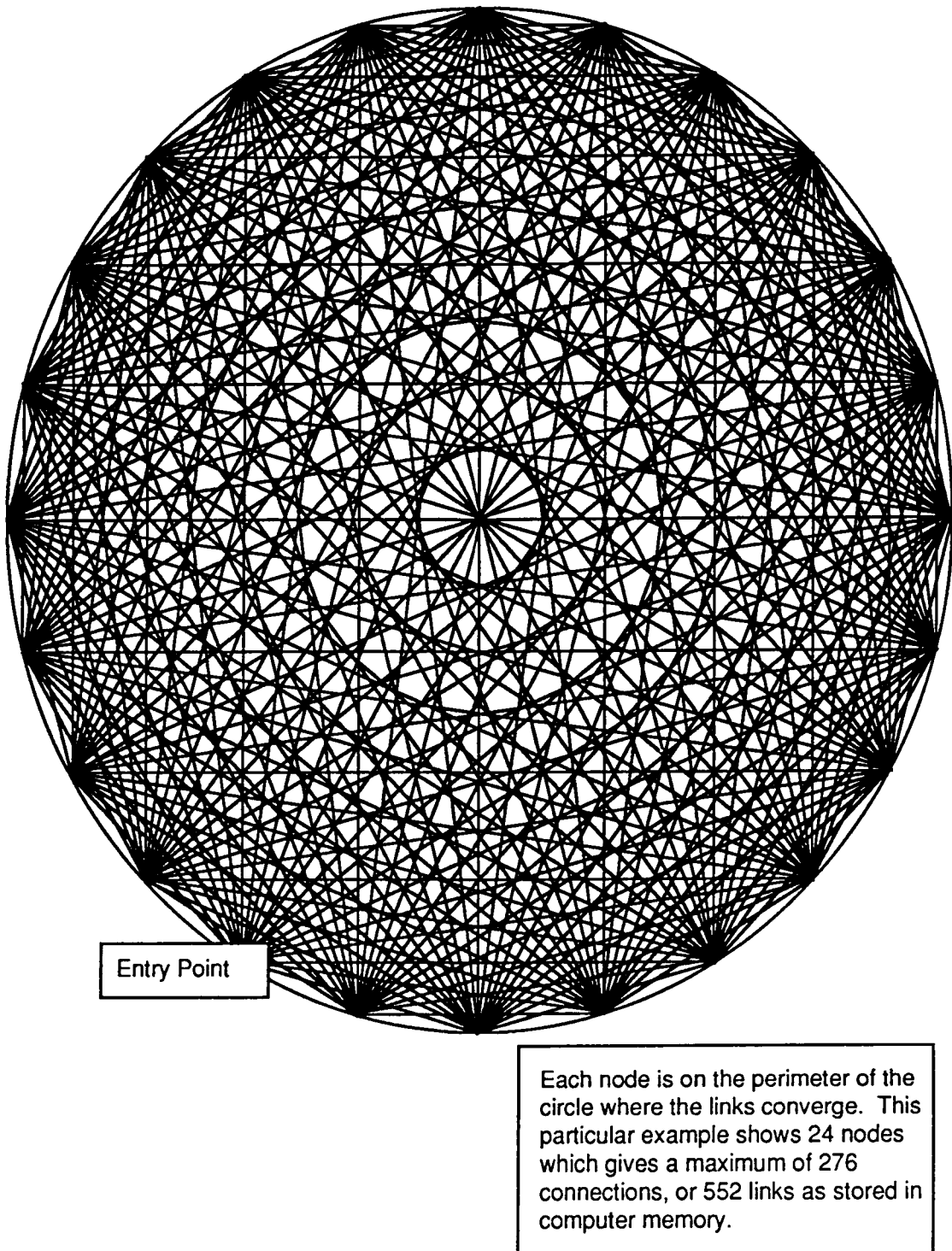
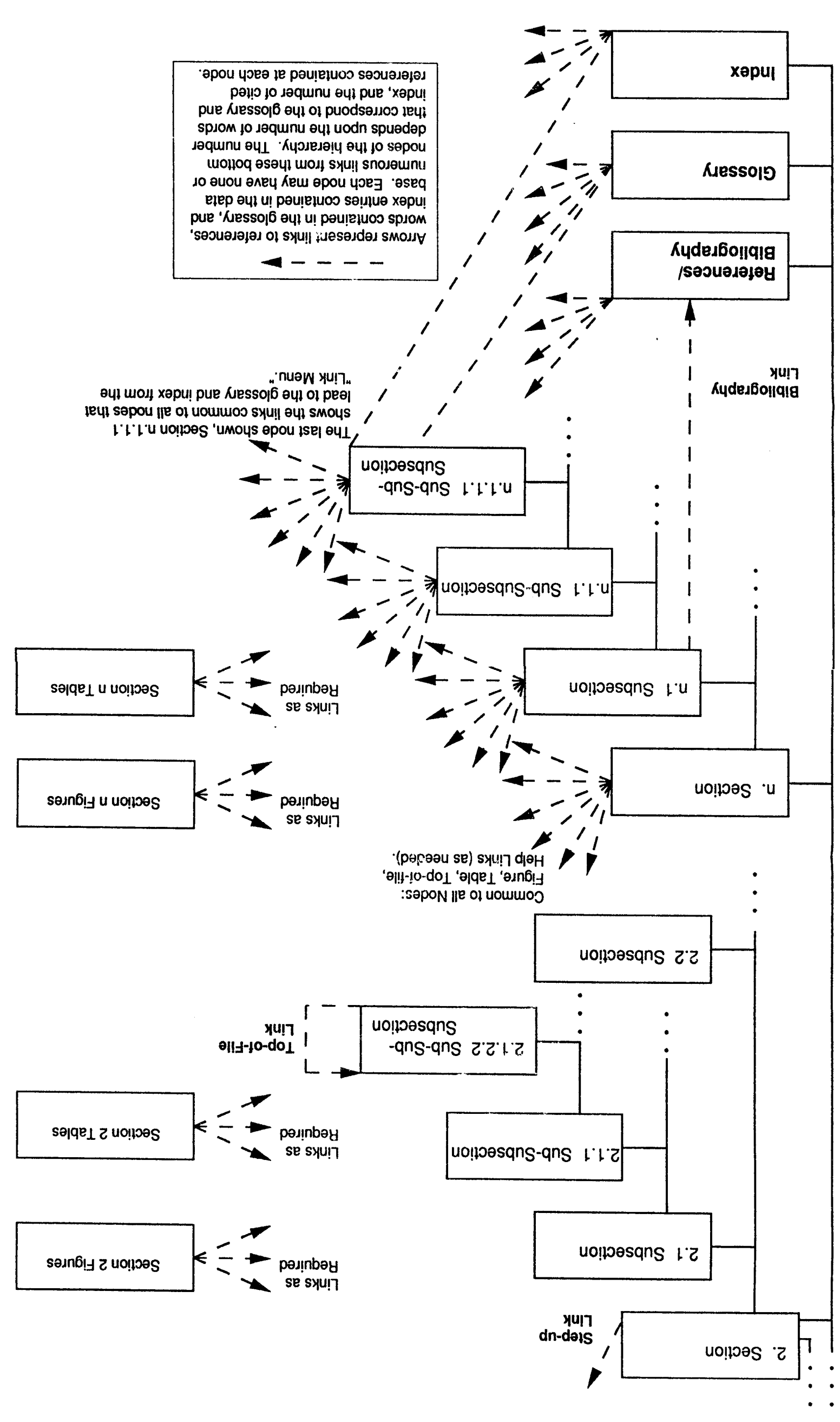


Figure 2: Node-to-Node Links.

Figure 2. SHIP structure and links (Page 2 of 2).



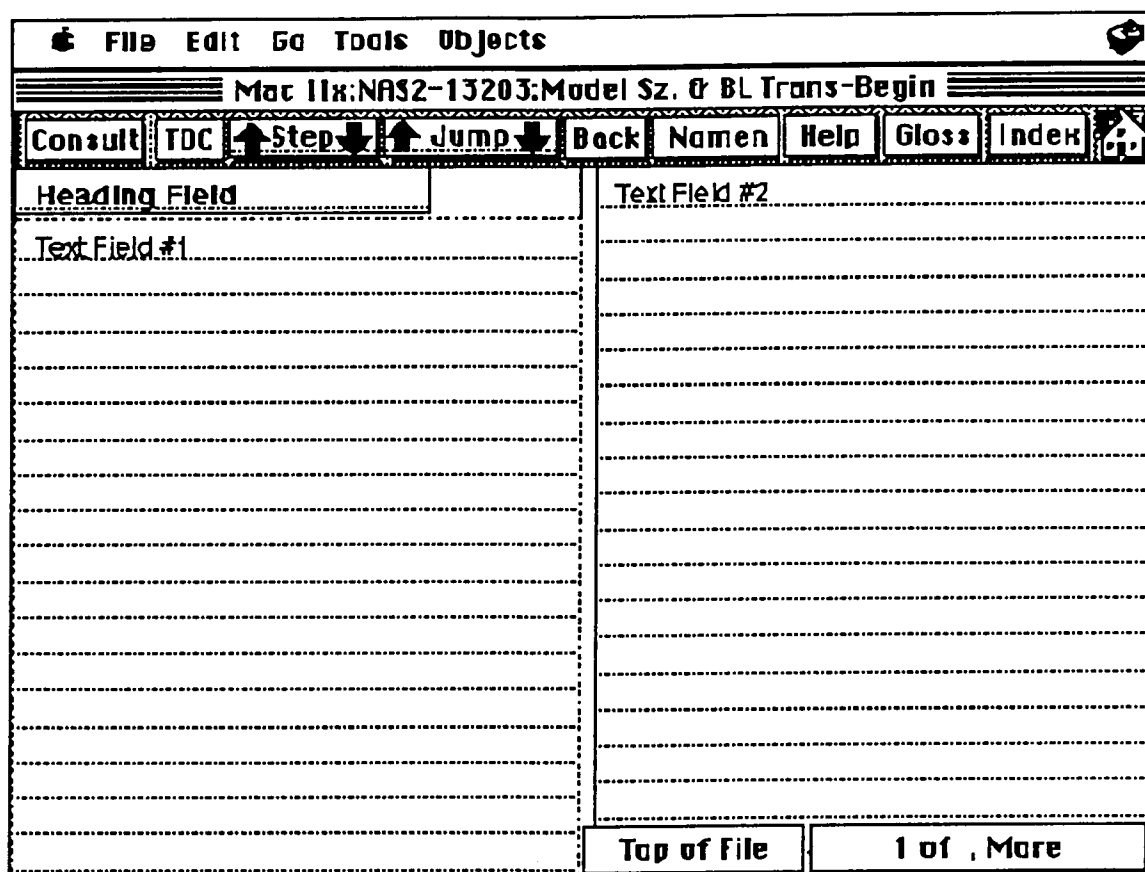


Figure 4. Hypertext Data Base Setup File in HyperCard.



Figure 5. Helvetica and Times Fonts.

[Consult](#) [Back](#) [Step»](#) [Nomen](#) [Help](#) [Gloss](#) [Index](#)

Model Size Selection in Wind Tunnel Testing

Based on Term Paper for AE513 by Norbert Ulbrich, Spring 1990

0 Table of Contents

[00](#) List of Figures

[01](#) Introduction

[02](#) Model Size Selection at Different Mach Numbers

[0201](#) Subsonic Wind Tunnel Testing

[0202](#) Transonic Wind Tunnel Testing

[0203](#) Supersonic Wind Tunnel Testing

[020301](#) Model Size Selection for A Running Tunnel

[020302](#) Model Size and Tunnel Starting

[020303](#) Model Size Selection in Free Jet Supersonic
Test Sections

[0204](#) Hypersonic Wind Tunnels--Blockage Tests

[03](#) Conclusion

[BIB01](#) Bibliography/References

Figure 6. Typical SHIP Table of Contents Display for the Model Sizing Documentation.

[Consult](#) [TOC](#) [Back](#) [«Step](#) [Step»](#) [Help](#) [Nomen](#) [Gloss](#) [Index](#)

[2](#) Model Size Selection at Different Mach Numbers

[2.3](#) Supersonic Wind Tunnel Testing

2.3.2 Model Size and Tunnel Starting

In a supersonic wind tunnel it is necessary to consider effects of a model in the test section on the starting of the tunnel. The maximum model size for tunnel starting has to be investigated. Before we discuss the influence of model size on tunnel starting it is important to derive some basic relationships.

When a supersonic tunnel is started, shocks will be created by the model. In general, a supersonic tunnel consists of two nozzles designed for Mach number 1.0 [Figure 2](#). Isentropic relationships can be applied from station 1 - 1' and 2' - 2; we know for the area ratios in this case:

$$\frac{A}{A_1^*} = \frac{1}{M_1} \left(\frac{1 + \frac{\gamma-1}{2} M_1^2}{\frac{\gamma+1}{2}} \right)^{\frac{\gamma+1}{2(\gamma-1)}} \quad (4a)$$

$$\frac{A}{A_2^*} = \frac{1}{M_2} \left(\frac{1 + \frac{\gamma-1}{2} M_2^2}{\frac{\gamma+1}{2}} \right)^{\frac{\gamma+1}{2(\gamma-1)}} \quad (4b)$$

Across the normal shock we have to apply the shock relationships:

Figure 7. Typical SHIP Text Display for the Model Sizing Documentation.

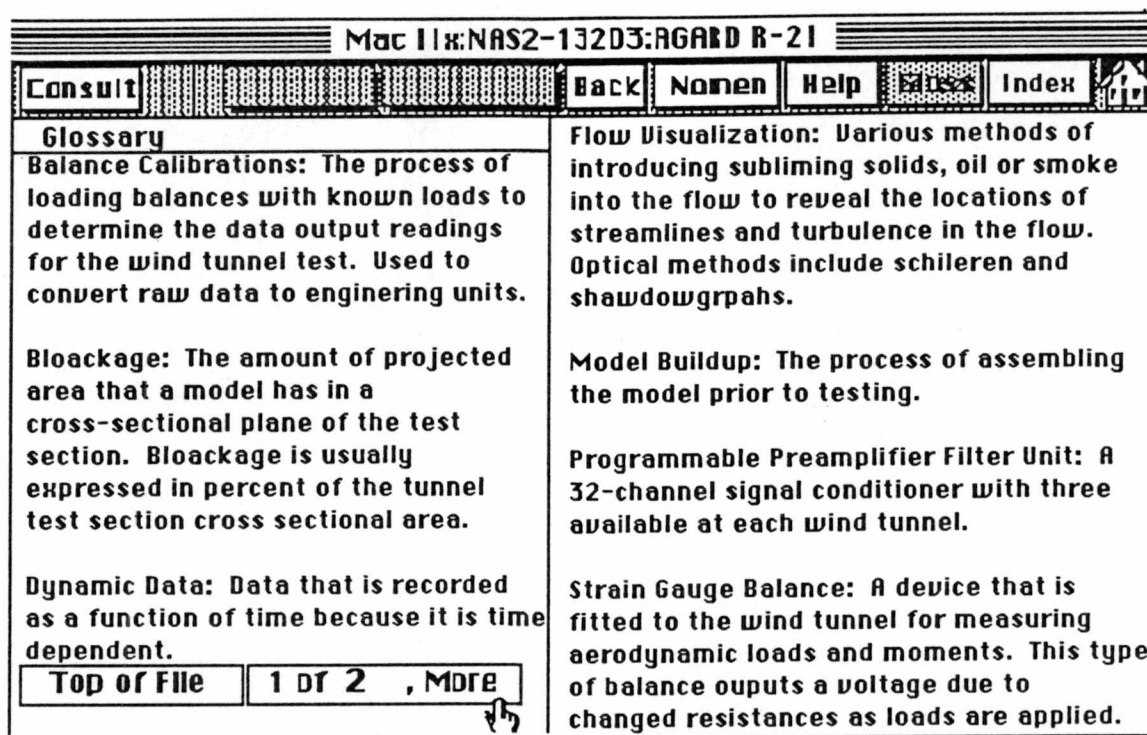


Figure 8. Typical SHIP Glossary Screen.

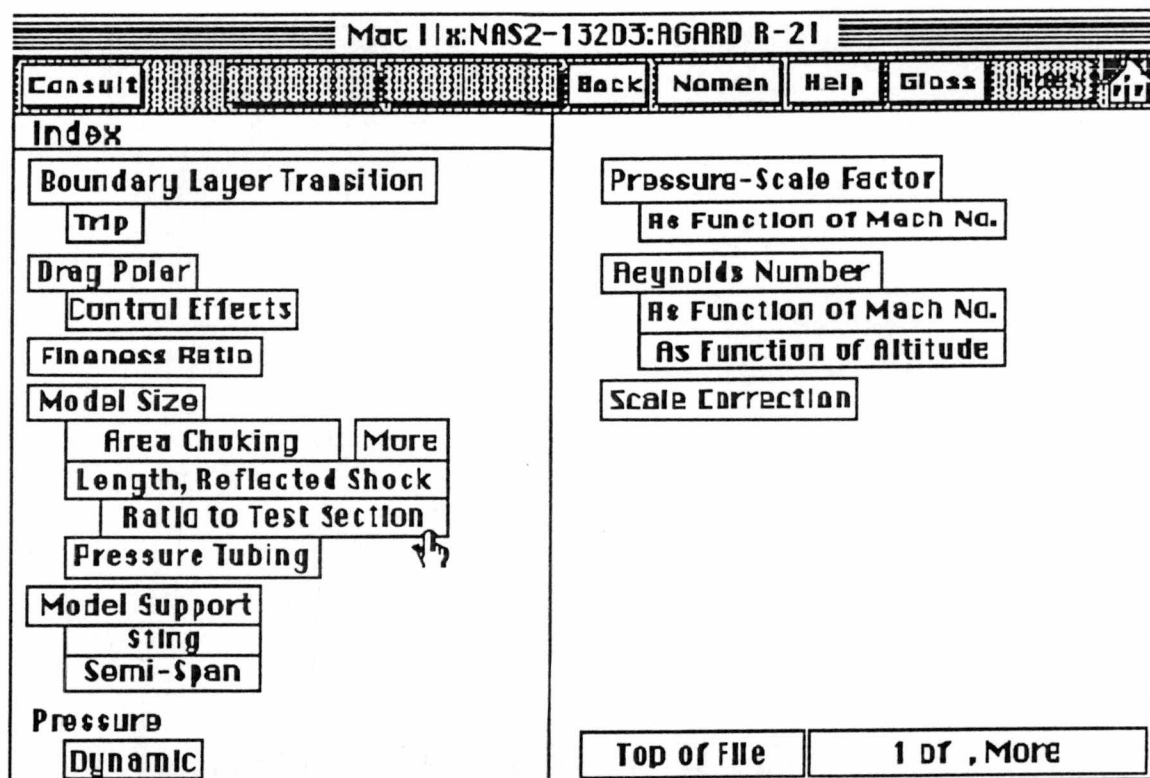


Figure 9. Typical SHIP Index Screen.

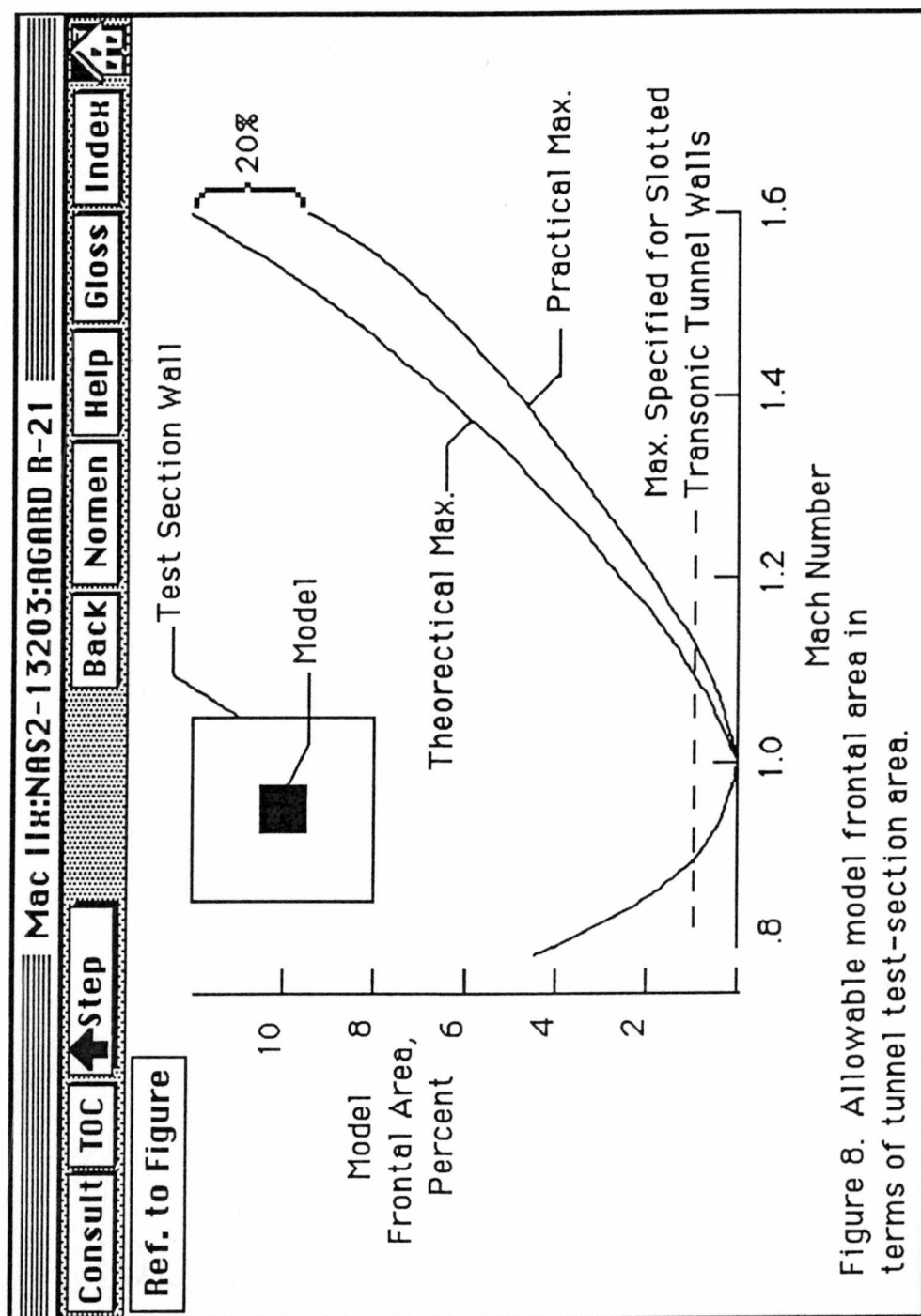


Figure 8. Allowable model frontal area in terms of tunnel test-section area.

Figure 10. Typical SHIP Graphic Screen.

TITLE Model Size Documentation Expert System !DISPLAY

! NOTE For HyperCard to work properly all file names must begin with a letter not a
! number, e.g. O402 (OH-four-ZERO-two).

ATTRIBUTE Card name
AND Tunnel
AND Mach No. Regime
AND Mach No.
AND Test Type
AND Model Type

FILE DocNavInfo
MULTI Card Name
AND Mach No. Regime

GOALS

1. Card Name
1.1 Mach No.

Rules that return the GOAL: Card Name

RULE Table of Contents
IF Tunnel \ Not Yet Specified
THEN Card Name \ 99

RULE Subsonic Tests
IF Mach No. Regime \ Subsonic
THEN Card Name \ O4
AND FILE Subsonic Tests Info

RULE Transonic Tests
IF Mach No. Regime \ Transonic
THEN Card Name \ FIG08
AND FILE Figure 8

RULE Supersonic or Transonic Tests
IF Mach No. Regime \ Supersonic
OR Mach No. Regime \ Transonic
THEN Card Name \ O402
AND Card Name \ 0403
AND ...

END

End Document Location Information

Figure 11. Sample Rules from the Model Sizing Document Location Expert System.

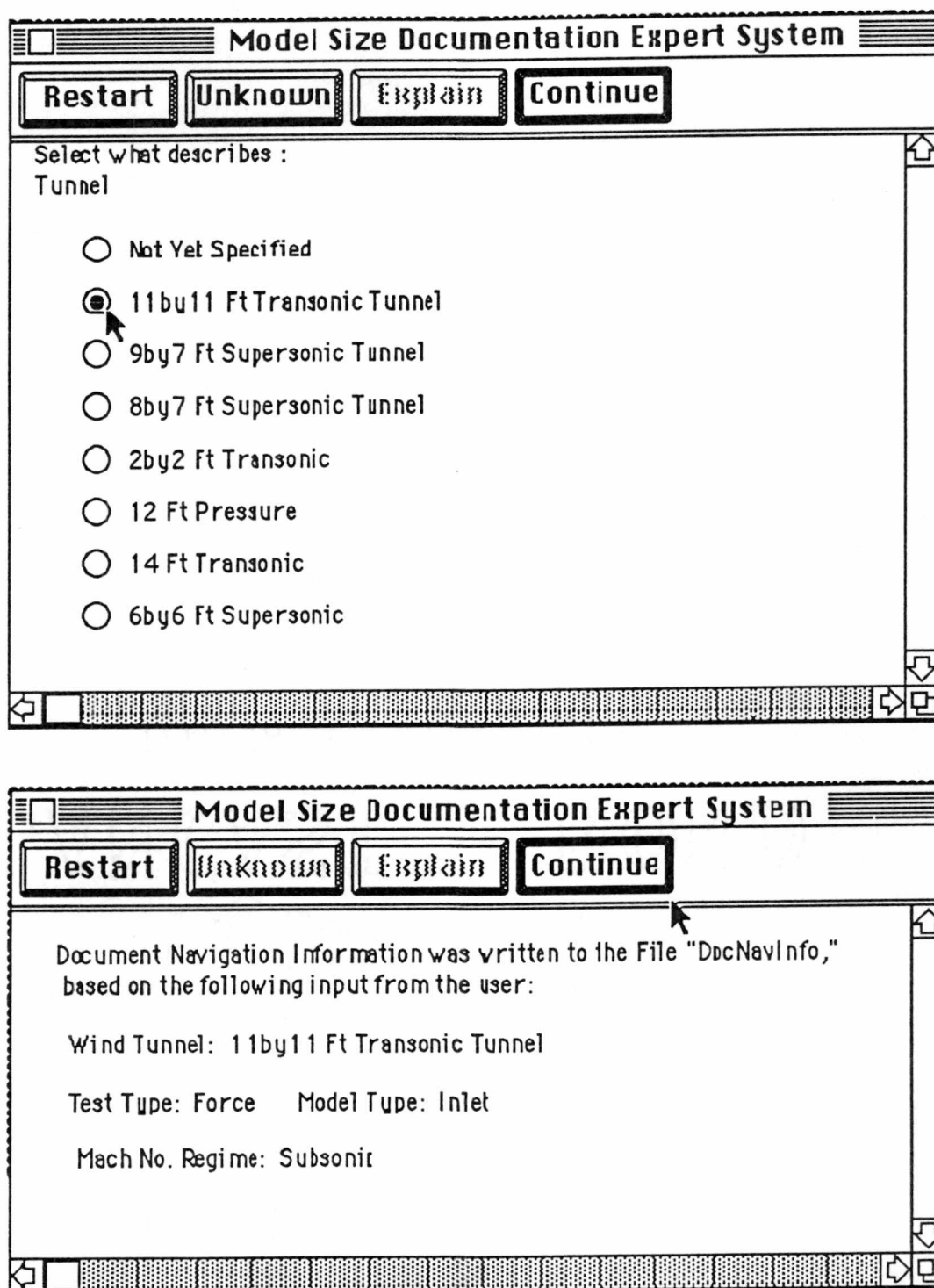


Figure 12. Sample Consultation Displays from the Model Sizing Document Location Expert System.

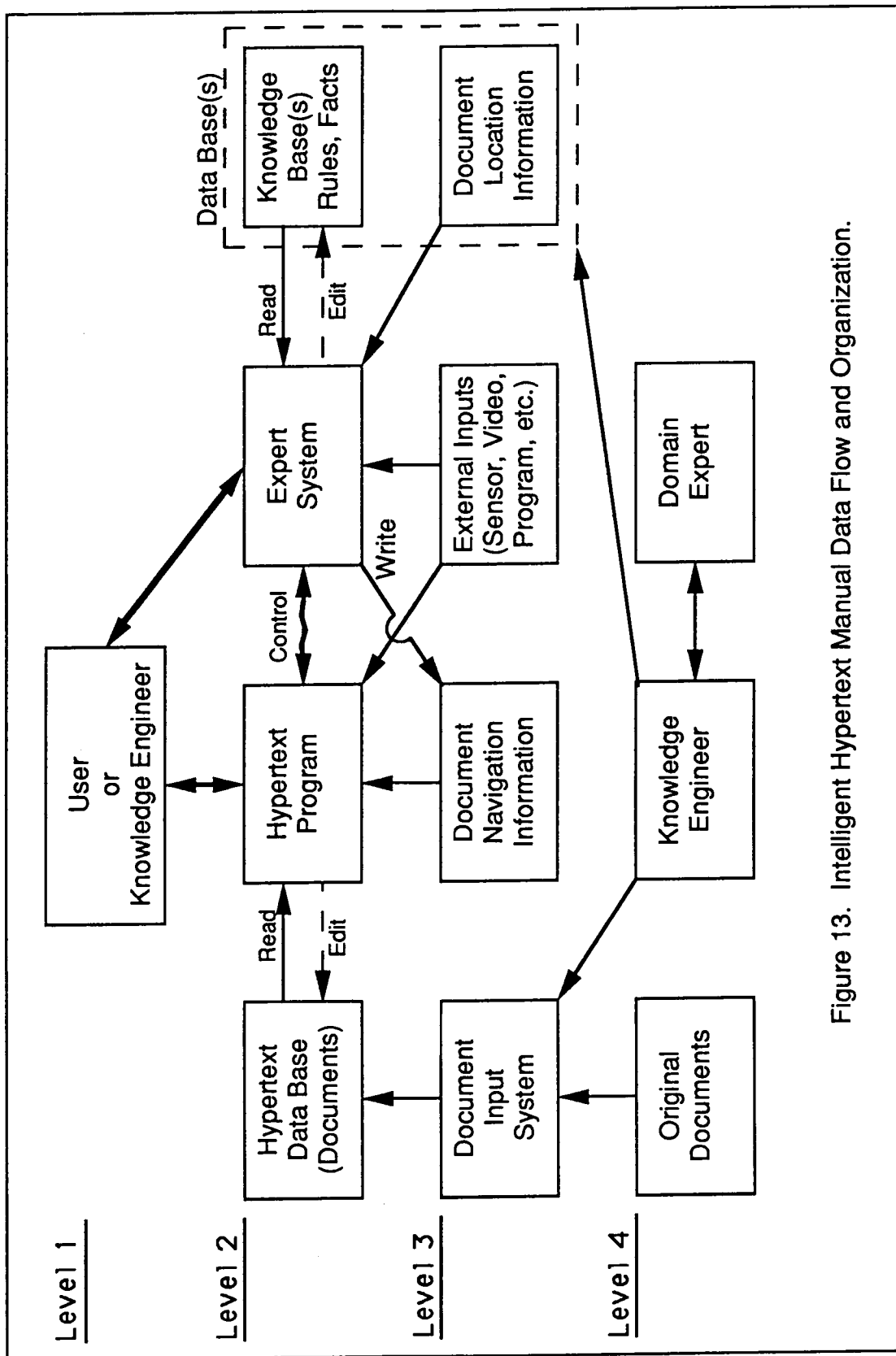
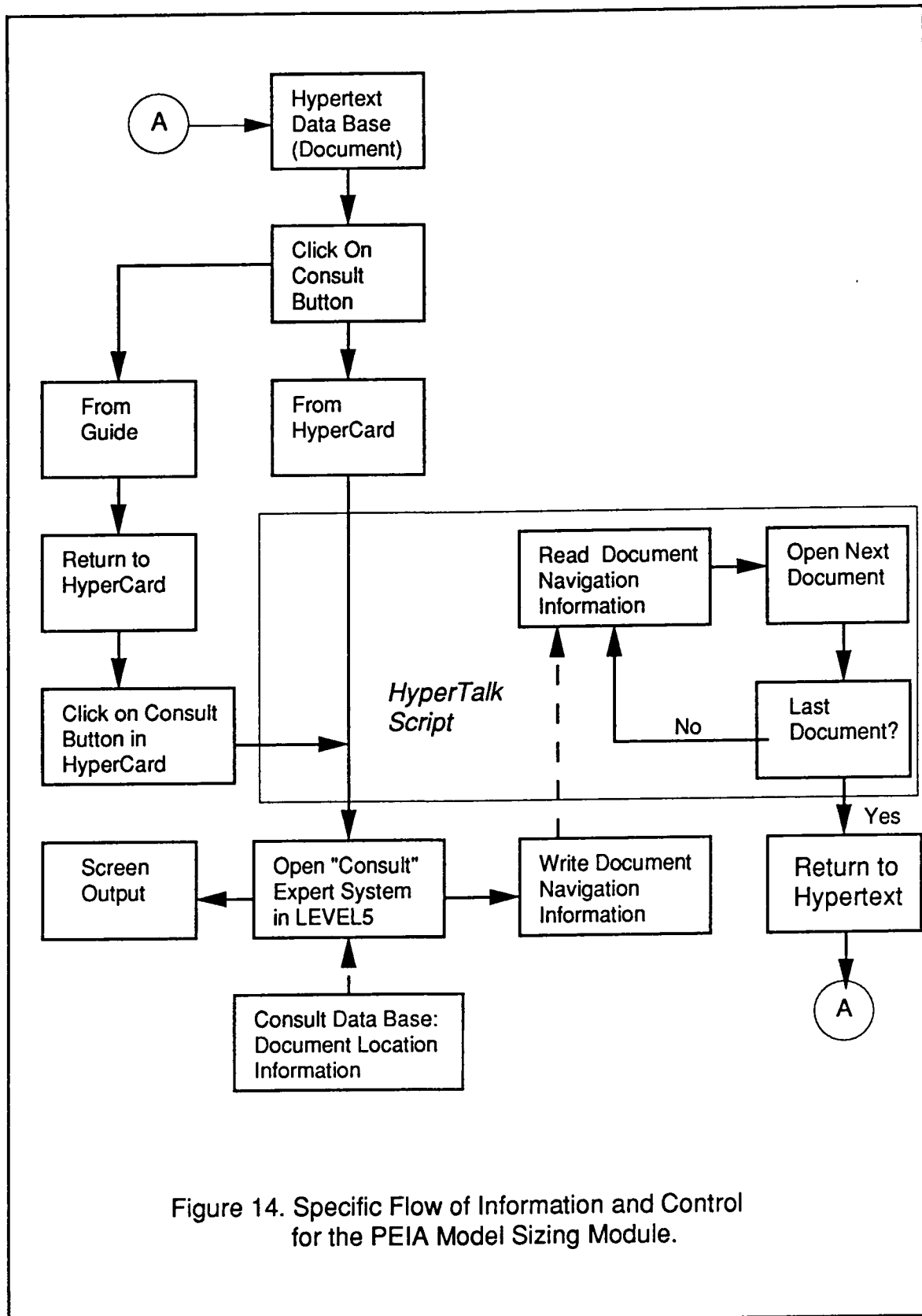
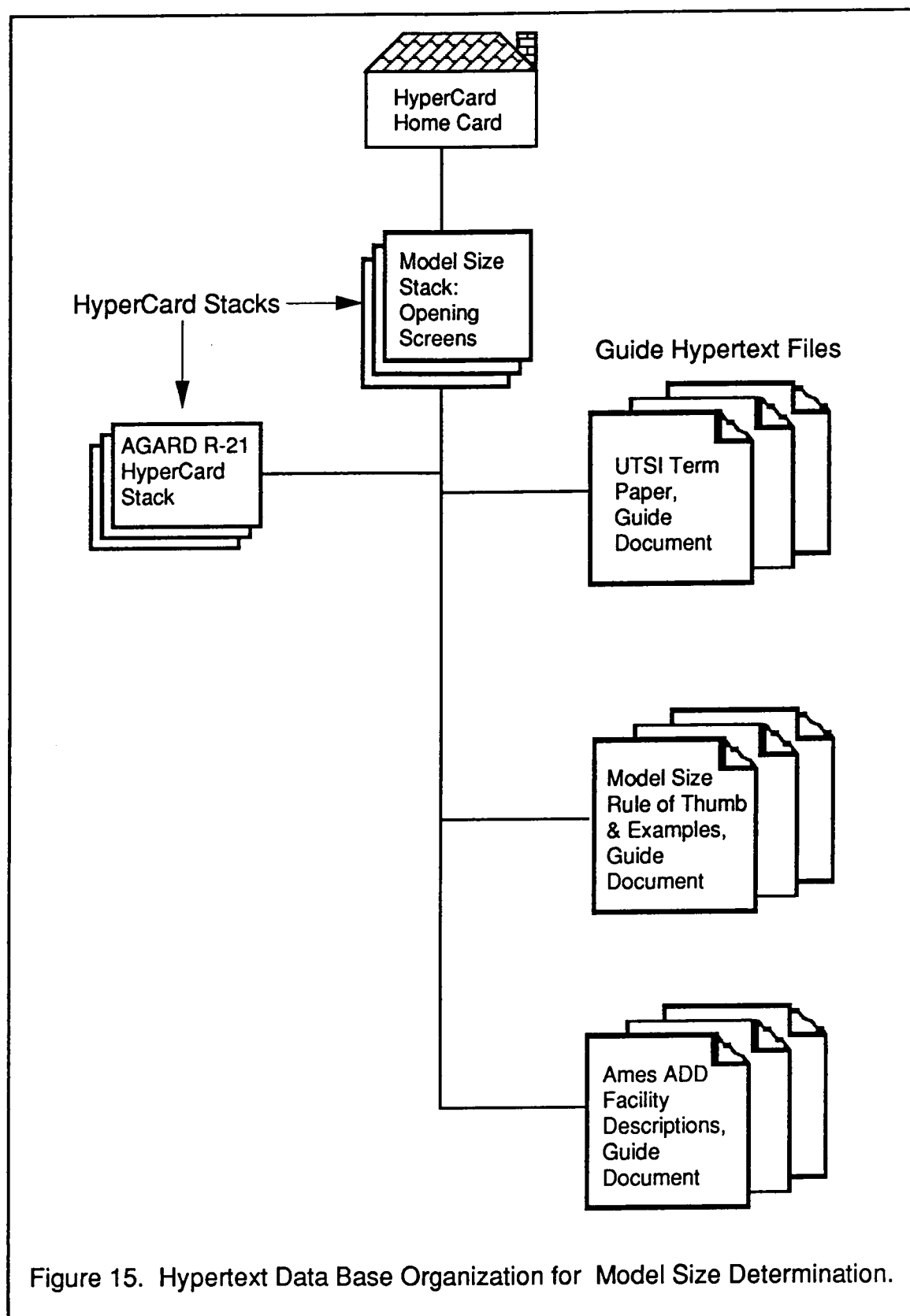


Figure 13. Intelligent Hypertext Manual Data Flow and Organization.





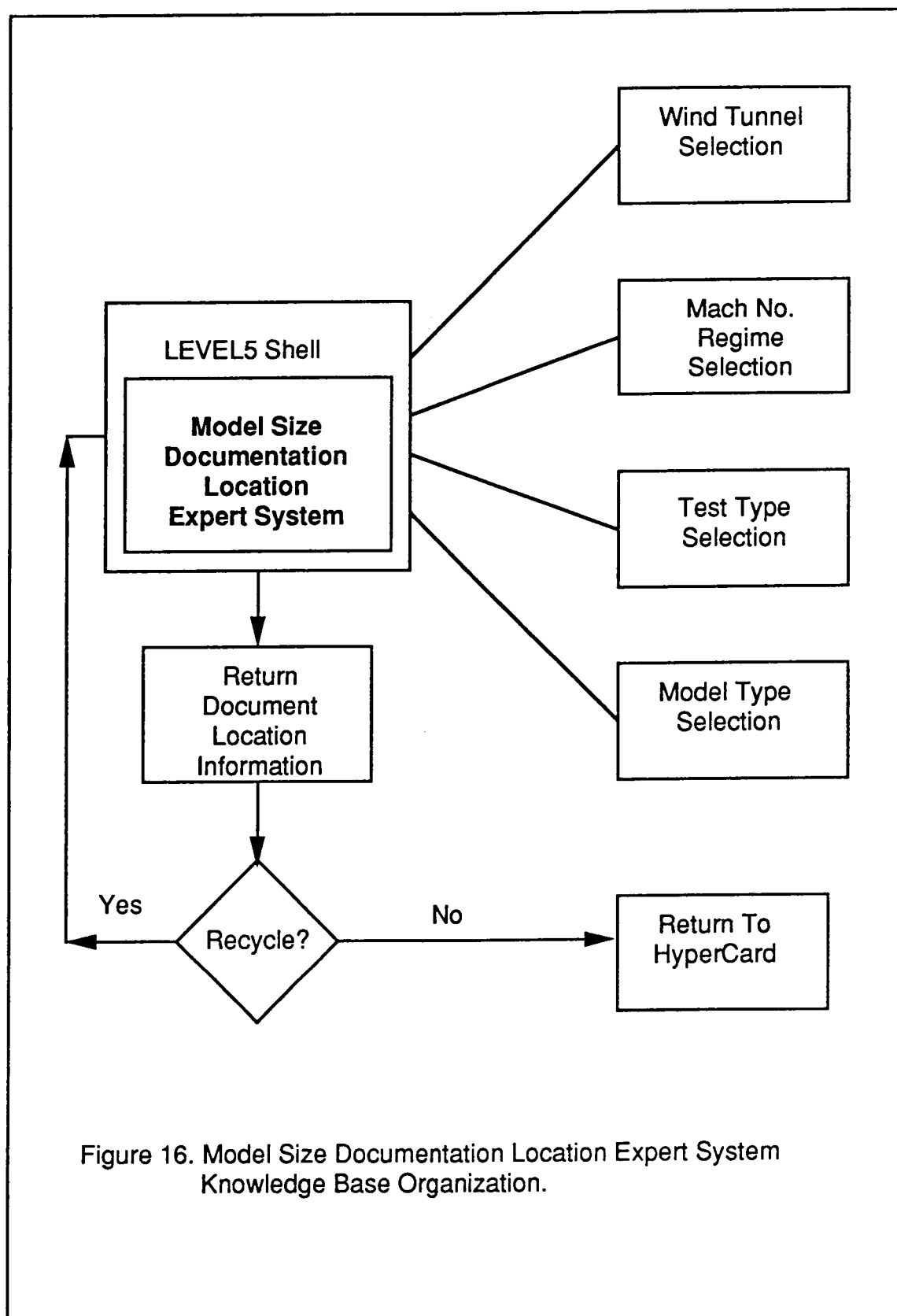


Figure 16. Model Size Documentation Location Expert System Knowledge Base Organization.

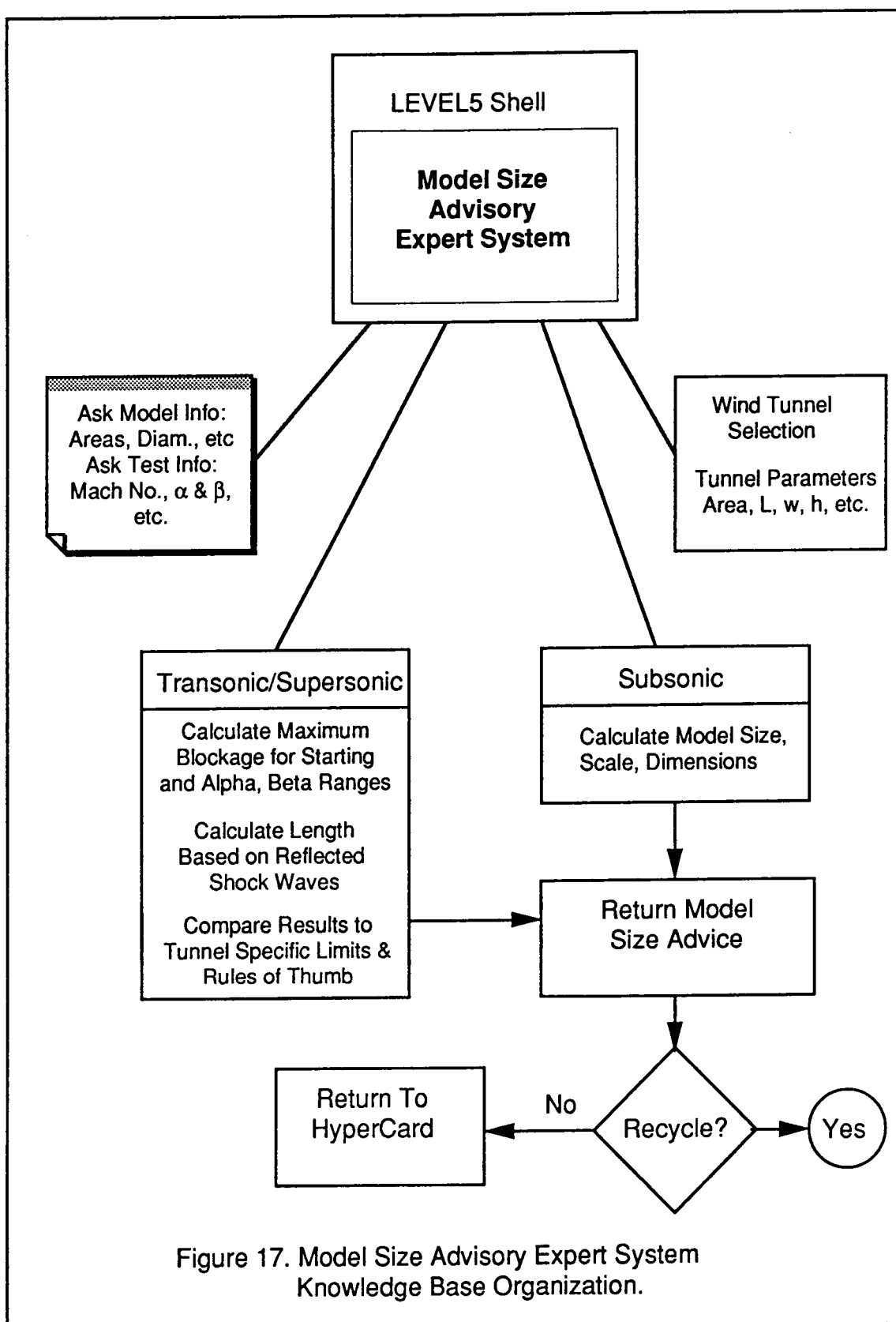


Figure 17. Model Size Advisory Expert System Knowledge Base Organization.

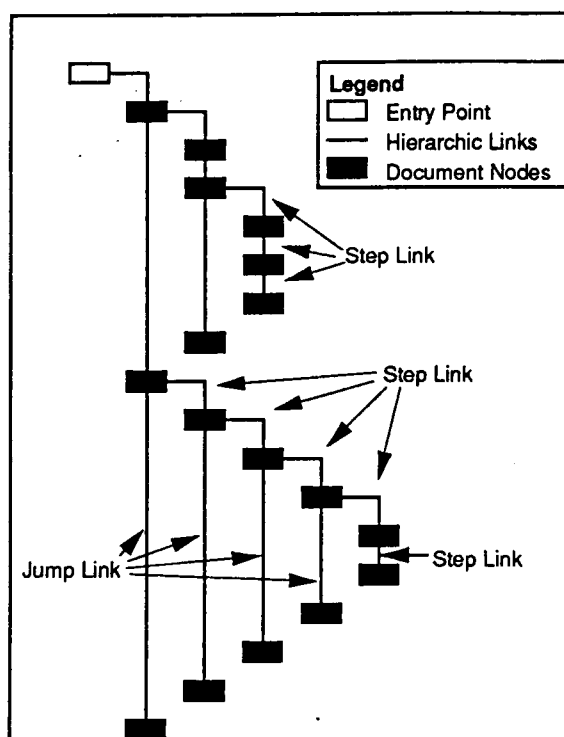


Figure 18. Typical Hierarchic Structure.

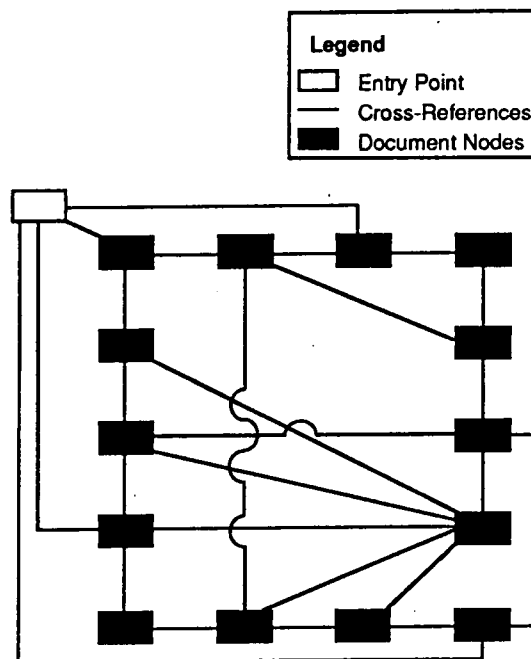


Figure 19. Cross-Referenced Data Base Structure.

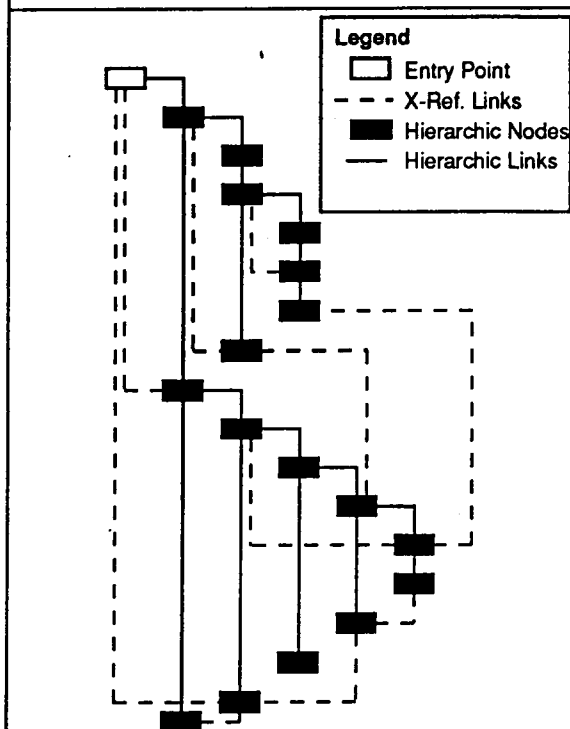


Figure 20. Combination Data Base, Hierarchic with Selected Cross References.

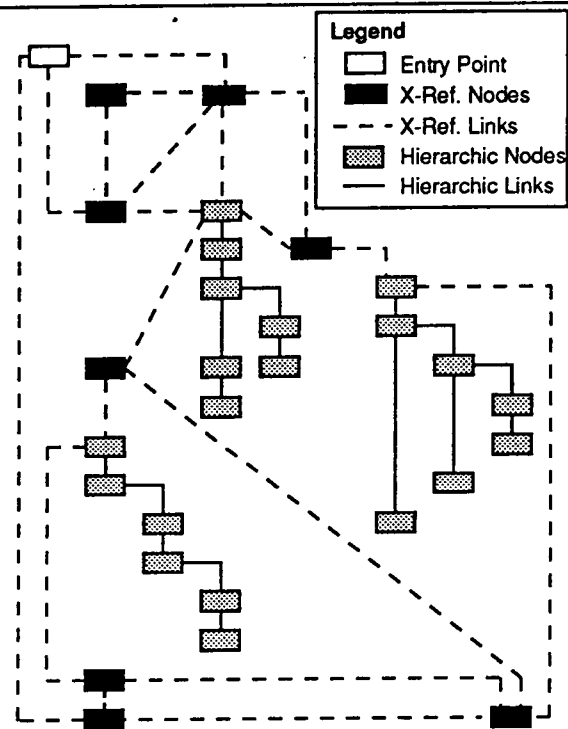


Figure 21. Combination Data Base, Cross-Referenced with Hierarchies Existing at Selected Nodes.

APPENDIX 4

TRADEMARKS

The following, legally named, computer programs and computer hardware were mentioned during the course of this thesis.

1-2-3 is a trademark of Lotus Corporation.

Apple, HyperCard, HyperTalk, and Macintosh are registered trademarks of Apple Computer, Inc.

dBase IV is a trademark of Ashton-Tate, Inc.

EXSYS and Pro-EXSYS are trademarks of EXSYS, Inc.

Guide, Scribbler and LOGiiX are trademarks of OWL International, Inc. (Office Workstation Laboratories).

IBM, IBM PC and AT are trademarks of International Business Machines.

KMS and KMS Action Language are trademarks of Knowledge Management Systems.

LEVEL5 is a registered trademark of Information Builders, Inc.

Microsoft is a registered trademark of Microsoft, Corp.

Windows and MS are trademarks of Microsoft, Corp.

Small Talk is a trademark of ParcPlace Systems.

Sun and Sun View are trademarks of Sun Micro Systems.

VP-Expert is a registered trademark of Paperback Software, Inc.

VITA

William Andes Hoyt Jr. was born in Aiken, South Carolina, on September 27, 1955. He began his elementary education in that city. He completed elementary school in Beaumont, Texas, and was graduated from Forest Park High School in June 1973. The following September he entered The University of Texas at Arlington and received a Bachelor of Science Degree in Aerospace Engineering in 1978.

He spent four years working with McDonnell Douglas in Houston, Texas, where he supported the Space Transportation System (STS). He was there for STS-1, the first flight of the Space Shuttle, and the second and third shuttle flights.

From there, Mr. Hoyt went to Rogersville, Tennessee, where he served as an editor for the Rogersville Review, a weekly newspaper. In 1985 he became publisher of the newspaper and served in that capacity until 1988 when he entered The Graduate School of The University of Tennessee, Knoxville. He attended classes and conducted research at The University of Tennessee Space Institute and received the Master of Science Degree in Aerospace Engineering in December 1990.

The author is a member of Sigma Gamma Tau Aerospace Engineering Honor Society and the American Institute of Aeronautics and Astronautics.

Mr. Hoyt works for Engineering Research & Consulting, Inc., located in Tullahoma, Tennessee.