(1)

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/MA/GCS/88D-1

AN OBJECT-ORIENTED APPROACH TO THE DEVELOPMENT OF COMPUTER-ASSISTED INSTRUCTIONAL MATERIAL USING HYPERTEXT

THESIS

Michael L. Talbert
First Lieutenant, USAF

AFIT/MA/GCS/88D-1

AFIT/MA/GCS/88D-1

AN OBJECT-ORIENTED APPROACH TO THE DEVELOPMENT OF COMPUTER-ASSISTED INSTRUCTIONAL MATERIAL USING HYPERTEXT

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Michael L. Talbert, B.S.
First Lieutenant, USAF

December 1988

Acknowledgements

I am grateful to many whose support provided me the priviledge of writing and receiving credit for this thesis. For hardware support I thank my officemate Capt. Jim Skinner, who saw to it that I had a computer and a desk to put it on. For office space and the software which enabled completion of this work, I thank my thesis advisor, Capt. David Umphress. I thank Dave also for the moral support he provided by agreeing to advise this thesis, and by providing timely, thoughtful feedback on the many drafts.

For technical support and expert advice, I thank my thesis committee members Professor Daniel Reynolds and LtCol John R. "Skip" Valusek. Extra thanks to Dan for his ceaseless enthusiasm, his interest in hypertext, and his commitment to "educating" engineers. Likewise additional thanks to LtCol Valusek for his interest in computer-based training aids and his unwillingness to compromise his high standards for the thesis work he reviews.

For additional support I thank Capt. Will Bralick, Maj. Jim Howatt, and the members of GCS-89D who contributed to the evaluation of my computer-based tutorial. I thank Dr. Charles Fenno for his willingness to discuss the educational implications and possibilities of using hypertext as I have done. Additionally, for logistics support I thank Kristine Zobrosky from the AFIT Engineering Library, who provided

invaluable support by acquiring numerous textbooks and journal articles which were essential for my research.

For moral support I am indebted to many as well. I thank my wife, [redacted] for her love and devotion, and for her determination to keep as busy as I have been throughout not only this thesis effort, but all of AFIT as well. I thank also my parents who supported me by their confidence in my abilities and my Christian friends who have served both as a source of encouragement and as a reminder that it's not how much free time you're given, but what you do with that time that counts. And above all, I thank God whose plans and ways are so beyond human understanding, for seeing fit to manifest His promises to me through the members of this support group.

Michael Lane Talbert

Table of Contents

List of Figures

List of Tables

Abstract

This paper combines the concepts of learning theory, knowledge engineering, software engineering, and hypertext. It presents a methodology for creating formally structured hypertext-based documents for use in transfer learning-oriented computer-based tutorials. The methodology, which parallels the proven Object-Oriented software design paradigm, facilitates the decomposition of a knowledge base into a hierarchical structure of text passages which form a tutorial. Application of the methodology results in encapsulated text objects which demonstrate many of the desirable software characteristics (e.g. modularity, cohesion). Evaluation of the methodology showed that it produced a computer-based tutorial which facilitates a learner's relationship-oriented assimilation of the concepts presented in the tutorial.

1. Introduction

1.1 Theme and Goals

Hypertext, with its capacity for non-linear representation of text and graphical images, is a useful software environment for the development and delivery of Computer-Aided Instruction (CAI). The application of a formal structure to hypertext in the light of instructional delivery techniques advocated by modern learning theory, can result in a computer-based product which facilitates improved understanding of the concepts presented. This thesis proposes an object-oriented methodology by which such a formal structure can be applied to the development of computer-based tutorials.

The goals of this research are:

1. Formalize and propose a methodology by which an unstructured or informally structured knowledge base can be mapped into a hierarchically structured, relationship-driven hypertext-based tutorial.

2. Using a prototype tutorial structured via the proposed methodology, obtain and analyze both objective and subjective measurements of the structure's learning effect on the users of the tutorial.

3. With the formal, built-in structure of a hypertext document, overcome some of the recognized pitfalls and frustrations associated with unchecked, ill-structured, or poorly implemented hypertext.

1.2 Background

Within the last few decades, training aids for instruction in both the military and civilian sectors have shown marked improvement from earlier training manuals and programmed text training books. Improvements can be ascribed primarily to the following two achievements: 1) advances in instructional methods stemming from research in cognitive sciences and pedagogical techniques, and 2) the introduction and more widespread availability of Computer-Aided Instruction (CAI) and Computer-Based Training (CBT) products and authoring systems. Computer-based products, ranging from simple "electronic page turners" to multimedia presentations, have been employed in various military, industrial, and academic training programs, and have met with initial success (see e.g., Dickinson, 1985; Enger, et al., 1985; MacNiven, 1987; Roman, 1985; Verano, 1985).

Although graphics and pointing devices (e.g. mouse, joy stick) have been used to enhance user interaction, much of the text of some CAI products is still little more than a training manual entered verbatim into the computer. The result is an essentially linear and otherwise unstructured presentation of the material. This organization of material generally promotes rote learning, not understanding of the underlying concepts and the relationships between them. In contrast, modern instructional methods advocate a non-linear

presentation of material, one that more closely models the way humans mentally represent information, and promotes understanding (Mayes, et al., 1988; Novak, 1977; Novak and , 1986; O'Shea and Self, 1983).

Hypertext and hypermedia environments are relatively new computer concepts which facilitate a non-linear presentation of material. Such environments are based on hardware-supported links between text and graphics (Conklin, 1987a:4). Hypertext environments are becoming readily available at relatively low cost, and can be used for the development of non-linear CAI materials, especially in the military training environment. Consequently, CAI products developed in this manner may extend or eventually replace the current linear computer-based products currently in use in DoD training programs.

1.3 Statement of the Problem

Only recently have the development and utilization of hypertext-based CAI materials for military training begun to be seriously considered or attempted (Linn, 1988; Psotka, 1987; Burns, 1988). The primary causes are: 1) commercially available hypertext technology has only recently become available and feasible (Barney, 1987a:26; Yankelovich, et al., 1985, 1987, 1988; Conklin, 1987a), and 2) as the absence of references in the literature indicates,

techniques for developing hypertext-based CAI materials are not widely known or practiced.

1.4 Scope of the Research

This thesis addresses the goals expressed earlier in this chapter. The primary end product of the research is a methodology for mapping the key concepts of a base of information into a representation which can be implemented using hypertext. Application of the research is limited to a reception learning mode of knowledge transfer, in which the student follows hypertext links to browse through the tutorial text. The tutorial product is suited for implementation by a commercial hypertext environment, which can be used on microcomputers such as the Zenith Z-248.

1.5 Assumptions

For this thesis, a commercially available hypertext or hypertext-like software environment was used in lieu of the creation of an original environment by the researcher. This substitution is justified by the realization that individual hypertext environments have been years in development and testing, and time does not permit any new creation. Most significantly, the hypertext materials developed as a result of this research must be portable to microcomputers running MS-DOS, such as the Zenith Z-248, which is already widely in use in the military. Further, large-scale hypermedia

environments such as Brown University's Intermedia are neither available to military CAI users and developers, nor are they currently portable to the Z-248 computers.

In addition, the prototypical hypertext product developed deviates from Conklin's ideal list of hypertext characteristics (Conklin, 1987a:6-7 and see Chapter 2). Primarily, with the prototype developed here, the user is not able to dynamically create his own links, but uses a pre-designed "learn-by-browsing" tool (Mayes, et al., 1988). This is not viewed as a shortcoming of the methodology. This is stated in view of the fact that the product is designed as a "transfer learning" or "reception learning" tool whose purpose is teaching the students from a specific body of information. However, flexibility, choice, and motivational factors are incorporated into the tutorial design (Landow, 1987; Linn, 1988).

1.6 Materials and Support Requirements

Successful completion of the goals of this thesis required the availability of a commercially available PC-based hypertext environment. Knowledge Garden's "KnowledgePro" and Owl's "Guide," both of which run on MS DOS compatible machines, were considered as candidate systems. Additionally, Apple's "HyperCard" program would support prototype development. The Knowledgepro environment was used in this thesis, primarily because MS-DOS compatible

microcomputers were readily available at AFIT for use by the researcher. Also, future research involving KnowledgePro is planned at AFIT, beyond this single work.

Technical assistance concerning the information content of the training material used in this thesis was provided by instructors in the Graduate Computer Systems curriculum in whose classes the product was evaluated. AFIT Mathematics and Operations Research Sciences faculty provided assistance in the design of the experiment used to evaluate the methodology. In addition, members of entry level computer science classes evaluated the usefulness of the prototypical hypertext-based product developed as a part of this thesis.

1.7 Overview

The overall thesis effort consists of four key phases. Chapter 2 surveys recent literature in the fields of Computer-Aided Instruction, knowledge representation, and hypertext. Chapter 3 presents a formal methodology which facilitates the decomposition of a knowledge base into a set of text passages which form the "textbase" of a computer-based tutorial. This methodology is offered as an structured method of hypertext document creation. Chapter 4 presents an example knowledge base decomposition using the proposed methodology. Chapter 5 discusses an evaluation of the methodology by way of experiment, examining the results of the tutorial as used in an educational environment.

2. Literature Survey

2.1 Introduction

The last decade has seen Computer-Aided Instruction (CAI) tighten its already firm hold in the schools and take a solid place in industry and the military as an in-house tutorial delivery system (see O'Shea and Self, 1983; Roman, 1985; MacNiven, 1987). Research on both the cognitive and computer sides of CAI has extended the basic CAI tutoring "drill and practice" function in primarily two ways. First, it provides a flexible, non-linear progression through CAI tutorials (O'Shea and Self, 1983:73-78). Second, it places the computer in the role of an Intelligent Tutoring System (ITS) (Ohlsson, 1987; Yazdani, 1987). The latter is a more recent enhancement and is a product of the application of Artificial Intelligence (AI) techniques. The former is most recently facilitated by "hypertext" environments, which allow virtually unlimited linking and branching between sections of text and graphics (Conklin, 1987a).

The focus of this thesis is on the design and development of microcomputer-based, "non-AI" CAI tutorials which use hypertext. This literature review presents a survey of the state of CAI, knowledge representation techniques, and hypertext, with the following questions in mind: What similarities hold between CAI desirables and

hypertext capabilities that will enable the two to come together, and how can the CAI textledge base be structured to facilitate the merger?

2.2 Computer Aided Instruction (CAI)

CAI has progressed from linear programmed "learning machines" to non-linear branching programmed instruction, to "adaptive teaching programs" which "decide" frame-to-frame progression in accordance with the student's performance history. Now, hypertext-based systems and intelligent tutoring systems are providing even more intelligent, more non-linear branching capabilities for CAI programs and systems (Self, 1983:74).

In industry, corporate training program managers have advocated replacing or supplementing lecture-based training with on-line or microcomputer-based training. They cite advantages such as the flexibility of individual instruction, availability of ready-made software packages, automatic progress checking and record keeping, and lower long run costs compared to lecture-based training (Roman, 1985:82-89).

In the schools, the focus of CAI upgrade efforts are not entirely on the physical capabilities of hardware and software, but on the support CAI presentation methods lend

to the actual learning process. The editors of Artificial Intelligence and Education, express this thought:

> Authentic knowledge means what is to be learned should not merely be 'added to the knowledge base,' but rather assimilated into the person's pre-existing system of knowledges, and even more, should be freely expressed from internal motives when appropriate. The emphasis leads to a focus on the activity of the learner because it must be HIS internal action that integrates new knowledge to old and expresses that integration creatively [Lawler and Yazdani, 1987:x].

The schools have found microcomputers to be a useful instructional delivery system which provides the interaction the learner needs to assimilate new material. Many educational journals (e.g. Academic Computing, Computers and Education, Educational Technology Systems, and Journal of Computer Assisted Learning), textbooks, and conference proceedings are dedicated solely to school utilization of microcomputer-based CAI. Currently, much of the published works approach school use of CAI in terms of evaluation, special uses, and design (see e.g. Hativa, 1986; Collier et al., 1987; McCaughey, 1986-87; Raymond, 1987; Boulet, 1987; Plambondon and Déschênes, 1986; Rambally and Rambally, 1987; Shaw, et al.,, 1985).

While the USAF has successfully employed computer-based training in many environments, a lack of centralization and organization in the design and development of CAI materials has prevented cross-service and intra-service uniformity among CAI environments and products (MacNiven, 1987). The

Army Research Institute's (ARI) Joseph Psotka has reviewed current Intelligent Tutoring Systems in use in the U.S. Army. He has found the combination of natural language processing technologies, AI database relationships, and hypertext interface capabilities demonstrates a very powerful method of instructional presentation in the military training environment. He concedes, however, that "too little work has been aimed at developing new representations for information and relationships" (Psotka, 1987:6), and has come to this conclusion:

> Of most obvious importance is the development of structures that encode and make explicit causal connections, and decompose systems into simpler causal structures where functional relationships are more apparent [Psotka, 1987:6].

2.3 Knowledge Representation

One method of representing these structures is the decomposition of a knowledge base through the construction of a "concept map" (Novak and Gowin, 1986), "associative network" (Charniak and McDermott, 1984), or "semantic network" (Psotka, 1987). These terms may be seen as concept-oriented equivalents of software-oriented "functional decomposition" (Pressman, 1987) and "object-oriented design" (EVB, 1985), and the relational data base concept of "entity-relationship" models.

Concept Maps. Concept maps have proven to be a versatile tool as a means of representing the non-linear structure of

related concepts or components. They have been used successfully as a knowledge elicitation tool in environments ranging from elementary education (Novak and Gowin, 1986) to operations research (McFarren, 1987), to artificial intelligence (Hayward, et al., 1987). McFarren lists three benefits concept maps provide as knowledge representation tools:

> 1) identification of the small number of key ideas within a subject, 2) a visual road map showing the conceptual journey, and 3) a schematic summary of the cognitive domain of interest [McFarren, 1987:46].

In the construction of a concept map, a body of knowledge or other related information components is represented as a network whose links are relationships between the concepts or components (Novak and Gowin, 1986). An example concept map used as an operations research knowledge representation tool is provided in Figure 2-1. Though concept maps are primarily used as a knowledge acquisition tool, (McFarren, 1987:51-52), concept-relationship structures are also used to define the structure of relational databases (see Figure 2-2) and rule bases of artificial intelligence (AI) and expert systems (ES) knowledge bases (Charniak and McDermott, 1984:22-29; Winston, 1984:253-259; Hayward, et al., 1987).

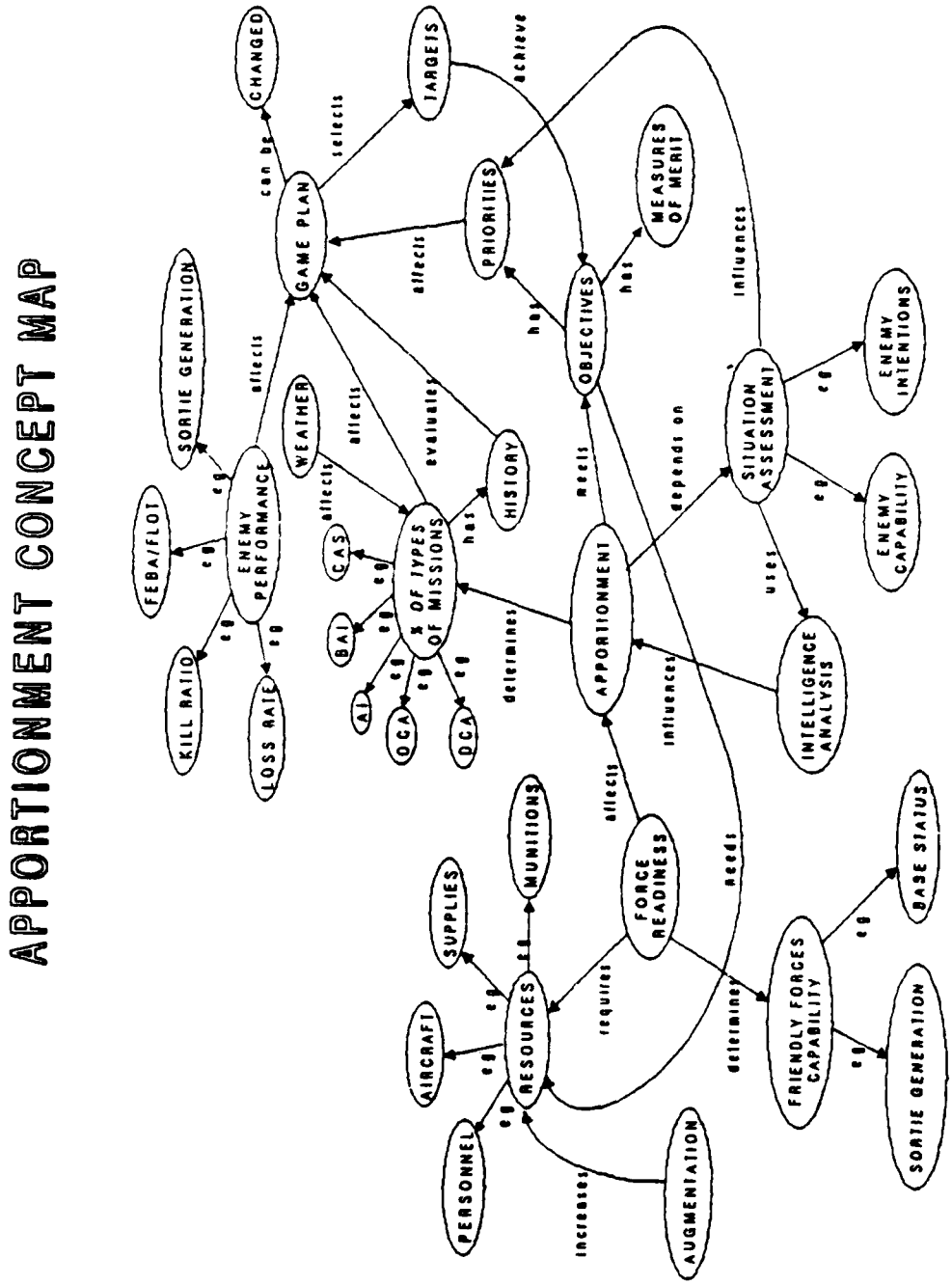

Figure 2-1. Concept Diagram as Knowledge Representation Tool (Wilcox, 1988)

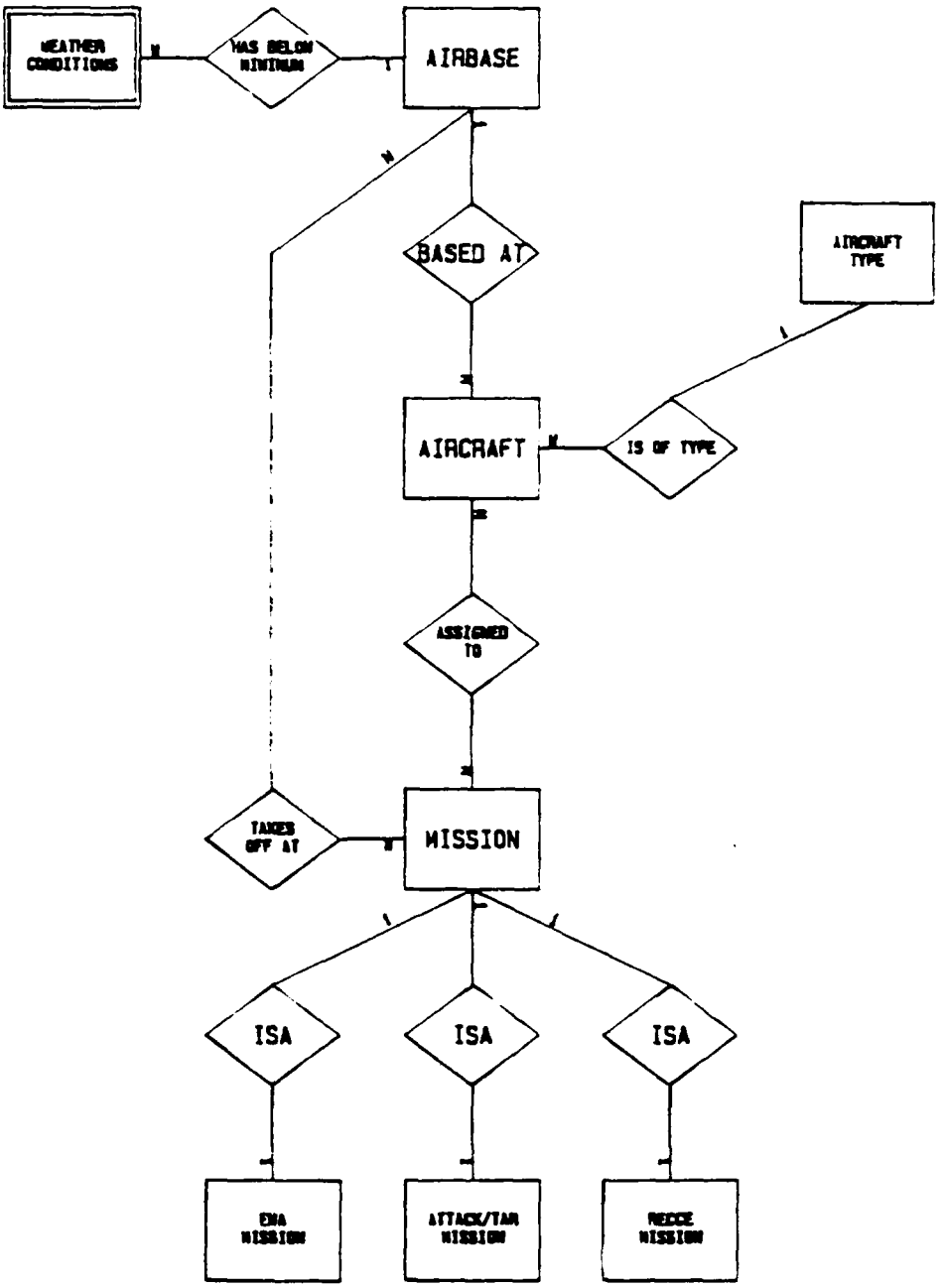

Figure 2-2. Concept Diagram as Relational Database Structure (Walker, 1988)

2.4 Hypertext

The recent introduction of microcomputer-based hypertext products such as Apple's "HyperCard" (Williams, 1987) and Owl's "Guide" (Hershey, 1987), and large-scale hypertext system research projects in the academic community (Beeman, et al., 1987; Smith, et al., 1987; Yankelovich, et al., 1987, 1988), have sparked widespread public and academic interest in hypertext systems.

2.4.1 Brief History of Hypertext

In 1945 Vannevar Bush introduced the basic ideas of modern hypertext systems with his "Memex,...a tool that provides access to a large collection of microfilm and mechanisms to make links between any two pieces of information in the system" (Smith, 1988:33). However, the term "hypertext" is credited to Theodor Nelson, who in 1965 defined it as:

> a combination of natural language text with the computer's capacity for interactive branching, or dynamic display...of a non-linear text...which cannot be printed conveniently on a conventional page [Conklin, 1987b:17].

By the late 1960's, Nelson, with the help of computer scientists Douglas Englebart and Andries van Dam, had begun to prototype the first hypertext system at Brown University. Surprisingly, the original utilization of hypertext was as a linear manuscripting tool (a souped-up text editor), primarily for the sophisticated editing features hypertext

offered. But by the mid 1970's, seminal experiments using hypertext as an educational tool were proposed and implemented at Brown University (Barney, 1987a:26).

2.4.2 Some Hypertext-Related Definitions

The definition of key terms is in order before a review of current research and applications of hypertext. A definition of "hypertext" has already been provided; however, a frequently used alternate term, "threaded text," which is perhaps a more intuitive expression of the idea, evokes the image of text physically linked by some fibrous matter (Thompson and Thompson, 1987a:25). Other popular terms are "non-linear text" (Shasha, 1987:163), and "non-lineal text" (Beeman, et al., 1987:67), both of which address the function of hypertext more than the definition, and imply that text need not be scanned in a strict sequential manner as in ordinary text books. "Hypermedia," then, is simply the extension of non-linearity to media beyond text and graphics, to include audiovisual media (e.g. film, spreadsheets, sound recordings, etc.) (Smith, 1988:33). Jeff Conklin gives a user-oriented description:

> The concept of hypertext is quite simple: windows on the screen are associated with objects in a data base, and links are provided between these objects, both graphically (i.e. as labelled icons) and in the data base (i.e. as pointers) [Conklin, 1987a:4].

It is also Conklin's works which address, generally from a technical standpoint, several other hypertext-related terms.

A "node" may be thought of as any discrete section of text or graphics, which is linked to another. Nodes may be single characters, words, paragraphs, pages, icons, etc., and serve as portals through which the user may enter another section of the hypertext document, arriving at a destination node. The new section may then present other nodes, which open to text sections with still other nodes. (Conklin, 1987a:48-53).

A "link" is the hardware supported connection between two nodes. One may move about from node to node via links. Links may be one-way or two-way, depending on the sophistication of the particular system hardware. According to Conklin, machine supported links are "the essential feature of hypertext systems....It is this linking capability which allows a non-linear organization of text" (Conklin, 1987b:18).

According to Conklin, beneath the user's view of a hypertext document, the underlying database of linked text and graphics is referred to as the "hyperdocument," or alternatively the "hypergraph" (Conklin, 1987a:6) (see Figure 2-3). Document systems so configured "consist of fragments of text embedded in a directed graph with labelled edges" (Shasha, 1987: 163).

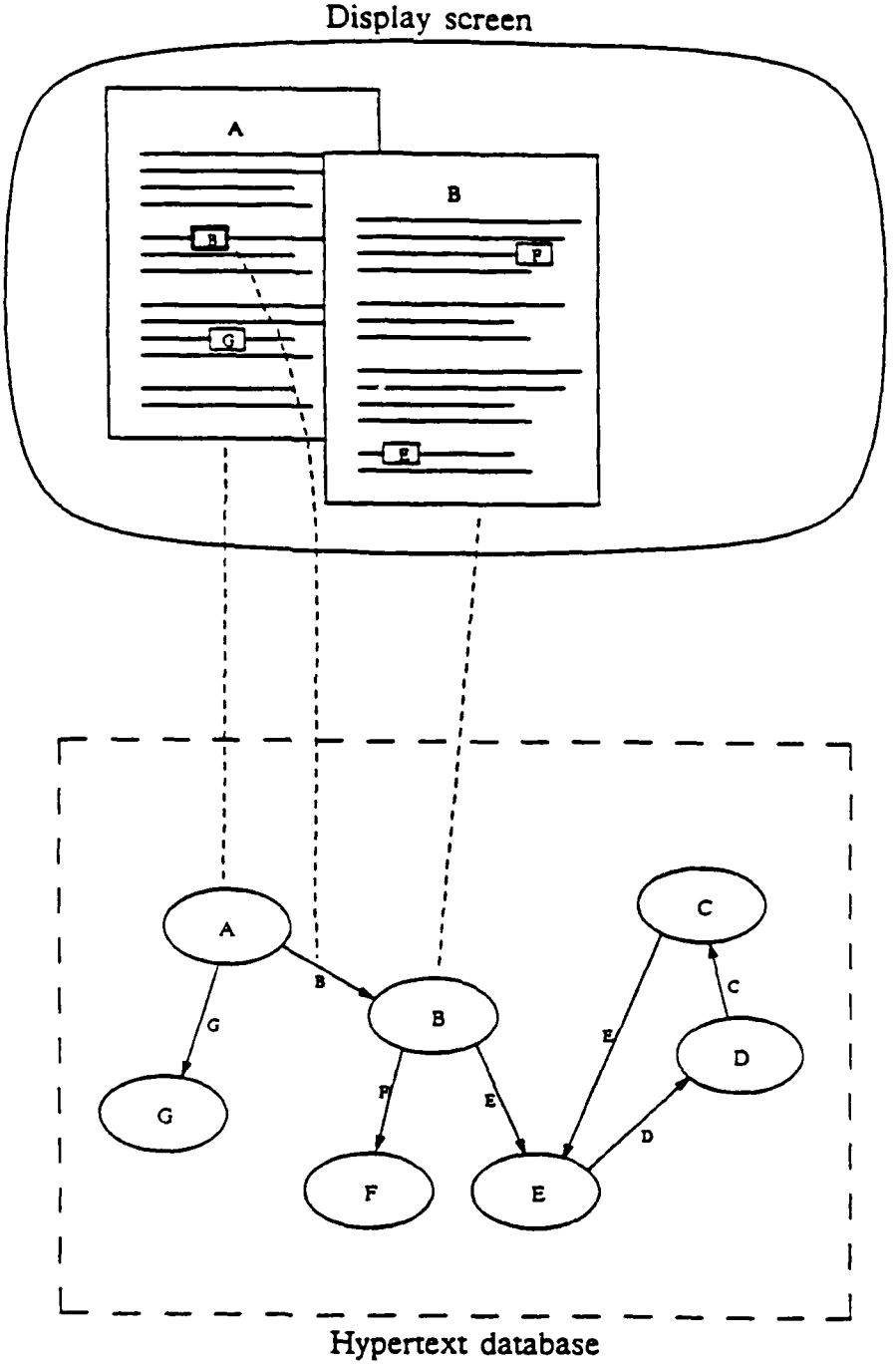

Figure 2-3. Typical Hypertext Document Structure
(Conklin, 1987a:5)

2.4.3 More Details Concerning Hypertext

Conklin's survey of hypertext (Conklin, 1987a), one of the more technical recent works, includes a discussion of features which characterize hypertext systems. Conklin (1987a:39-53) states that an ideal hypertext system should include:

- A database which is a network of linked text and graphics nodes;
- A fully supported window system in which named windows correspond to text or graphics nodes in the database network, and in which windows may be represented by icons on the screen and opened and closed with a pointing device on the icon;
- A set of commands or tools by which a user may create new links between text and graphics nodes;
- A means by which the database may be visited by a) following links between established nodes, b) searching for keywords and strings, and c) by using a specialized graphical 'browser' that displays the nodes of the database, their relative position to each other.

Of Conklin's listed characteristics, it is the use of multiple on-screen windows which gets much of the attention in the literature. This capability, while neither new nor strictly tied to hypertext systems, has itself become a

separate research issue (see e.g. Tombaugh, et al., 1987; Norman, et al., 1986). The use of multiple window displays provides the user with the capability to examine, often simultaneously, separate sections of text or graphics. Not surprisingly then, windowing functions in combination with the branching abilities of hypertext systems have opened the door for non-linear reading and writing of electronic literature (Neuwirth, et al., 1987; Thompson and Thompson, 1987a; Tombaugh, et al., 1987; Yankelovich, et al., 1985).

2.4.4 Applications of Hypertext

Hypertext systems have been applied in information-intensive environments, many academic. Gary Marchionini and Ben Shneiderman submit that hypertext can impact electronic information systems from three primary directions: information retrieval, interface design, and cognitive science (Marchionini and Shneiderman, 1988: 71-72). Karen Smith discusses other current and proposed uses of hypertext, including document retrieval and file manipulation (Smith, 1988). Still other applications include text browsers (Yankelovich, et al., 1985) and writer's tools (Neuwirth, et al., 1987; Smith, et al., 1987; Beck, 1988; Mayes, et al,. 1988).

Educational Use of Hypertext. Of most relevance to this thesis is the dedication of hypertext and hypermedia systems to the non-linear presentation of educational

material. The bulk of research in this area has been conducted at Brown University in Providence, Rhode Island, by the Institute for Research in Information and Scholarship (IRIS). IRIS coordinator, Nicole Yankelovich, has led most of the Brown University research, and is one of the most published authors concerning hypermedia and CAI to date (see e.g. Yankelovich et al., 1985, 1987, 1988). Further, the seminal Brown University studies are the source of most education-oriented hypermedia journal articles (see e.g., Beeman, et al., 1987; Garrett, et al., 1986; Smith, 1988; Yankelovich et al., 1987, 1988).

The hypermedia research at Brown University began in the summer of 1985 when IRIS team members began developing a "large-scale hypertext/hypermedia system called Intermedia" (Smith, 1988:32). Intermedia was designed to be a system which "enables instructors and students to create, organize, visualize and connect multimedia information with a set of integrated editors and application programs" (Yankelovich, et al., 1987:1). The Intermedia system has since been used in an English Literature (Landow, 1987) and a Cell Biology classroom, and has met with initial success. According to the participating instructors, students in the hypermedia-based classrooms felt they came out of the class with a deeper understanding of the course material, and would choose other such courses over regular "linear" courses in the future (Beeman, et al., 1987; Yankelovich et al., 1987).

Besides the ongoing work at Brown, little other work published to date addresses hypertext-based CAI projects in the academic community. However, on behalf of the DoD, projects are planned or underway at the University of California at Berkeley (Linn, 1988), the Army Research Institute (Psotk 1987), and the Air Force Human Resources Laboratory at Brooks AFB, TX, where the "Hypertext Initiative Program" is slated for later this year (Burns, 1988). At U.C. Berkeley, the proposal has been made to introduce hypermedia into the DoD, as a means of training DoD members in computer programming languages and programming techniques (Linn, 1988). At Brooks as at the ARI, emphasis is on intelligent training systems which are based on non-linear text (Psotka, 1987; Burns, 1988).

2.4.5 Large-Scale Hypertext Systems

Intermedia. Perhaps the largest and most well-implemented hypertext system to date is Brown University's Intermedia (hypermedia) environment. Both an author's tool and a user's tool, the icon-driven, window-driven, menu-driven Intermedia system is based on tools available on the Apple MacIntosh, and runs on a network of IBM RT PC workstations, using UNIX 4.2 (Smith, 1988:34; Yankelovich, et al., 1988:82). The main components of Intermedia are five applications tools, "a text editor (InterText), a graphics editor (InterDraw), a scanned image viewer

(InterPix), a three-dimensional object viewer (InterSpect), and a timeline editor (InterVal)" (Yankelovich, et al., 1988:82).

Intelligent Design Environment and Authoring System. The ARI, in its research of Intelligent Computer-Aided Instruction systems (ICAI), is investigating the use of several hypertext-based systems. The primary one, IDEAS, is based on Randall Trigg and Xerox PARC's NoteCards system (Conklin, 1987a:13-15; Psotka, 1987:11). The basis of the system is computerized 3 X 5 index cards. According to Psotka:

> A Notecard is intended to contain a small, single, idea-sized chunk of information, either in textual or graphical form. A collection of Notecards can be arbitrarily linked together to form networks that convey the relationships among the ideas stored in the various Notecards....The links are visible within each notecard as a hotspot title or icon that is mouse-sensitive and pops up into the full text or graphic when it is buttoned [Psotka, 1987:11].

Like the MacIntosh-based Intermedia, the Lisp-based IDEAS uses a unique toolset made up of three "IDEs." The Instructional Design Environment (IDE1), is designed to "help trainers think more thoroughly about their instructional intent" (Psotka, 1987:13). The Instructional Development Environment (IDE2) provides "tools to manage the process of development, make it faster and more effective, and make it conform more closely to the specifications laid out in IDE1" (Psotka, 1987:13). The Instructional Delivery

Environment (IDE3) "provides facilities for delivering, presenting, testing, controlling, and managing the training" (Psotka, 1987:13). IDEAS uses two high resolution screens on PC workstations, similar to Intermedia.

The U.C. Berkeley System. The U.C-Berkeley study implements a hypertext system built around an advanced workstation such as the Apple MacIntosh, and takes a position near the middle of the large-scale to small-scale continuum. Their system will "combine recent advances in understanding how students learn and how instruction can be effective with recent advances in technology to provide powerful self-paced programming courses" (Linn, 1988:1). Their proposed environment emphasizes "expert strategies for the design of computer programs," and "self-paced instruction including acquisition of self-regulation skills," and "would also include a database of alternative representations for the same programming problems" (Linn, 1988: 1-2).

2.4.6 Microcomputer-Based Hypertext Systems

Apple's HyperCard. As Conklin, who has reviewed nearly all hypertext-like systems in use to date, points out, "Ironically, HyperCard is not hypertext" (Conklin, 1987a:32). In fact, HyperCard was never really intended to be hypertext, but instead "a personal toolkit that gives users the power to use, customize, and create new

information using...text, graphics, video, music, voice, and animation" (Williams, 1987:109; Conklin, 1987a:32). Even so, HyperCard has been very well received by hypertext enthusiasts (Barney, 1987b).

HyperCard appears on the screen as one of a stack of notecards, similar to the NoteCards system described earlier, with mouse-sensitive buttons, icons, pull-down menus, etc. HyperCard provides the non-linearity characteristic of hypertext systems, but, being graphics driven, does not provide text-to-text links (Conklin, 1987a:32).

Recently, however, the MacIntosh's graphics-to-graphics linking capabilities have been used in "StrathTutor," a "Learning-by-Browsing" Computer Assisted Learning (CAL) system developed in Europe. Though not a true hypertext system, StrathTutor is an intelligent CAI system, which the creators regard as an example of a "reactive learning environment." The system is predicated on providing user interaction which encourages learning through "exploring the database and developing a conceptual structure of it" (Mayes, et al., 1988:222-228).

Owl's Guide. Guide is a MacIntosh-based hypertext system which, in contrast to HyperCard, allows for both text-text and text-graphics linkages, more in line with Conklin's description of an ideal hypertext system (see

above). Boldfaced text and graphics objects can be buttons that are linked to new text and graphics. Some buttons cause the expansion of "hidden text," much like the expansion of an outline heading. Other buttons cause new windows to open, which in turn contain new buttons (Hershey, 1987).

KnowledgePro. One of the newest microcomputer-based hypertext products has it roots in AI systems. KnowledgePro is itself actually an expert system shell. However, features of KnowledgePro allow it to function as a hypertext environment. It allows for and enforces structured groupings of hypertext nodes. With KnowledgePro, text sections become icons by being highlighted in different colors (Thompson and Thompson, 1987b; Knowledge Garden, 1988).

TextPro. Textpro is a less sophisticated hypertext-generating component of the larger expert system environment, KnowledgePro. In contrast to HyperCard and Guide, TextPro (as well as KnowledgePro) is designed for use on MS-DOS compatible machines, like the Zenith Z-248 personal computers in use in the USAF (Thompson and Thompson, 1987b; Knowledge Garden, 1987).

3. Proposed Knowledge Base Decomposition

3.1 Motivation for a Textbase Structure

This chapter implements the relationship-based structure of knowledge representation with the network-based structure of hypertext. The result is applied to the "textbase," the collection of paragraphs which form the textual material, of a computer-based tutorial. In the process, there are obstacles which must be overcome. First, the traditional linear format of text too greatly constrains the written representation of information. Hypertext affords a means for overcoming this shortfall. However, and secondly, the general network structure of hyperdocuments may be unsuitable for presenting some instructional material (Smith, et al., 1987). The challenge, then, is this:

> Hypertext authors must...transform inchoate ideas into coherent structures that can be comprehended as well as traversed. Users of hypertext documents must...understand what they read (or see, or hear...) and must construct relations between new information and old, one idea and another [Smith et al., 1987:196].

And to do so,

> one must employ devices that enforce hypertext capacity to establish intellectual relations" [Landow, 1987:334].

The goal of this chapter is to draw on principles of learning theory and software engineering and apply them to structuring a hypertext-based tutorial. The result is a

methodology for designing and developing a relationship-driven tutorial textbase.

3.1.1 Learning Theory Advocates Formal Structure

From a cognitive perspective,

> The central purpose of any education is to communicate to students a grasp of the essential and central ideas of the subject matter. These are the concepts, principles, and procedures around which the propositions of the knowledge base of the discipline are organized...[Shulman and Ringstaff, 1985:12].

And the structuring of the presentation of facts, concepts and principles of a discipline is essential to learning those facts, etc. (Shulman and Ringstaff, 1985:11-12). However, in the realm of reception or transfer learning, the central ideas of a subject matter are commonly articulated via the written prose of textbooks and journal articles. What structure there is of this method of presentation is inherently linear and sequential.

There are several shortcomings of a strict sequential presentation of text. First, it does not facilitate understanding of the concepts it presents. Cognitive theorists view true learning as taking place when a new concept can be understood by its relationship to concepts already learned (Novak, 1977:64-93; Shulman and Ringstaff, 1985:11-17; Woolfolk and McCune-Nicholich, 1984:239-243). According to David Ausubel's theory, a reader assimilates new information by relating it to previously understood concepts, subsuming new details under broader concepts

(Novak, 1977:24-28; Woolfolk and McCune-Nicholich, 1984:239-243). In this way, we understand new ideas in a hierarchical structure. Older, broader concepts are at the top and finer details are toward the bottom.

Secondly, a sequential text format does not conform to the network structure presented in Chapter 2 as concept maps. A network structure closely parallels the reader's "mental model" of information (Marchionini and Shneiderman, 1988:72-74); it is necessary for establishing long-term memory of concepts (Smith et al., 1987). Here again, it is the relationships among concepts which are necessary for learning.

3.1.2 Potential Pitfalls with Hypertext

The fundamental representation of hypertext enables a writer to construct the necessary relationship-oriented links between concepts in a hyperdocument. However, the structure of a typical hyperdocument is neither a hierarchy nor a formal network, but is simply a directed graph of nodes connected by links (Conklin, 1987b). Without formal structure and the "[enforcement] of hypertext capacity to establish intellectual relations" (Landow, 1987:334), there are potential problems with carelessly constructed hyperdocuments.

Unchecked Network Promotes Disorientation. One common caveat in hypertext literature addresses the potential for

readers to become disoriented in large and complex hypertext networks. In this state, the reader may not fully appreciate the relationships between text sections. Remedies to this problem include tracing the reader's trail through the hyperdocument and displaying global or localized "neighborhood maps" of the hyperdocument. Through these mechanisms, the reader may orient himself in the document (Conklin, 1987a; Oren, 1987).

A related disorientation problem is that of document closure. In this case, the reader may be left with no sense of how much has been covered or remains unread. Since the document is non-linear, the textbook concept of an "end of the book" no longer applies. As with the general disorientation problem, graphical roadmaps of the document are often used as a remedy (Oren, 1987).

Misleading Links Frustrate Readers. Although disorientation is related to the hyperdocument structure, semantic problems internal to the document text, are equally as disquieting to the learner. A reader must navigate through the hyperdocument via links marked as nodes. These link markers commonly appear directly in the text as highlighted words, phrases, etc., and invite the reader to explore. When the nodes are carelessly labeled, the linked text may not be what the reader expected.

George Landow, of Brown University's Institute for Research In Information and Scholarship (IRIS) has observed these three rules concerning hypertext links:

> 1. Hypertext links condition the user to expect purposeful, important relationships between linked materials.
>
> 2. The emphasis upon linking materials in hypertext stimulates and encourages habits of relational thinking in the user.
>
> 3. Since hypertext systems predispose users to expect such significant relationships among files, those files that disappoint such expectations appear particularly incoherent and nonsignificant [Landow, 1987;332-333].

Landow's rules are addressed further in this chapter. As an example of the third rule, in a very large hyperdocument with pre-established links connecting a wide and varied collection of text passages, some node names may likely be homonyms. Consequently, a reader in the middle of a discourse on "butterflies" may activate a link on a node labeled "monarch," and end up in a text concerning "European Rulers" (Raskin, 1987:327). Nodes with vague or ambiguous labels are also undesirable. These nodes may link to material whose content is unrelated to that of the text from which the link is activated. These are understandably sources of reader frustration. They waste the reader's time, interrupt his thought pattern, and could reduce his desire to explore additional links (Landow, 1987).

A proposed solution to these problems, in view of the need for a hierarchical, relationship-driven textbase, is to impose structure on the hyperdocument. Structure can be enforced even as the document is being designed. For this much needed structure, we may borrow the structured approach to design from the principles of software engineering (Pressman, 1987).

3.2 Borrowing Structure from Software Engineering

A premise upon which this thesis is built is that the design of a tutorial textbase is not unlike the design of any other software system. Using a loose translation of the term, the text and graphics which comprise the textbase may themselves be considered a form of "software" (Pressman, 1987:5-12; Webster, 1973). That is, since the text segments and nodes are not hardware, they must be software.

With that equivalency in mind, what about the principles of software engineering makes them useful for developing hyperdocuments? Foremost, the application of software engineering principles results in formally structured, understandable, modular software components. In addition, these modules enforce data abstraction and information hiding (Pressman, 1987:222-230). In view of the desired structure and function of a tutorial as expressed earlier, the motivation for a software engineering approach to tutorial design becomes clear.

But the field of software engineering offers numerous design methodologies, such as Jackson System Development (JSD), Data Structured System Development (DSSD), and Object-Oriented Design (OOD) (Pressman, 1987). These methodologies employ equally as many program flow representations (e.g. Data Flow, Transaction Analysis, Transfer Analysis) (Pressman, 1987). A desirable software engineering model for use as a means of structuring concept-based learning material is one which facilitates a concept-oriented approach. The object-oriented representation facilitated by the OOD methodology satisfies this need. As a result, the textbase design methodology proposed here is patterned after the OOD software design paradigm.

3.2.1 The Object-Oriented Software Design Paradigm

OOD provides a mapping from a problem space (all the components of the problem to be solved) to a solution space (the software which models or solves the problem). Formally, the OOD methodology involves the sequence of steps outlined below (Booch, 1983:40-44).

1. State the problem succinctly in a single sentence
2. Bound the problem statement to a single paragraph
3. Identify the objects involved in the problem
4. List the attributes of the objects
5. Identify operations on the objects
6. Define relations and interfaces between objects
7. Decide on implementations of the objects and operations
8. Recursively apply the process to the operations as needed

The first step in an OOD decomposition is the identification and clarification of the problem to be solved. EVB (1985:2-2) advocates accomplishing this step by stating the problem in one complete, grammatically correct sentence. From this one sentence, the scope of the problem is defined and bounded by expanding the sentence to a single paragraph. It is typically a five to nine sentence paragraph which describes, in high level terms, the problem to be solved. It is important to write this first problem description at a constant level of abstraction to preclude ambiguity or incompleteness in the lower levels of the problem space decomposition (EVB, 1985; Pressman, 1987).

After bounding the problem, the process of decomposition begins. OOD effects decomposition by representing the problem space as data objects and operations on and by them. Objects are the "major actors, agents, and servers in the problem space" (Booch, 1987:48). Operations "serve to characterize the behavior of each object or class of objects" (Booch, 1987:49). Typically, the objects are the nouns and noun phrases in the problem

description and the operations are the verbs and verb phrases (EVB, 1985:2-6 - 2-10).

The identification of objects and operations begins at a high level of abstraction and seeks progressively lower levels of abstraction (higher levels of detail). For example, a first level object may be a "Fuel_Regulator," with associated operations such as "Monitor_Flow_Rate" and "Control_Fuel_Flow." Since each of these terms are broad enough to encompass several lower level objects and operations, further decomposition may be warranted. In this case, another problem description may be needed for each of these subproblems.

As in the first iteration, nouns and verbs are identified as components of the next lower level of decomposition. These might include objects "Rate_Meter" and "Flow_Valve" and operations "Read_Meter," "Open_Valve," and "Close_Valve." Applying this technique, a software designer may then recursively decompose the results of each iteration until sufficient detail has been extracted (Booch, 1987; EVB, 1985; Pressman, 1987).

Once the objects and operations have been identified, a process of refinement begins. Unnecessary or superfluous objects are no longer considered. Like objects are grouped or classified. Data types and operations which apply

specifically to an object are encapsulated in a stand alone module (EVB, 1985:2-10 - 2-15; Pressman, 1987:350-353).

In this encapsulated manner, there exists a bond between an object, its data types, and the operations which act on or are exported by the object. Pressman states that "objects 'know' what operations may be applied to them" (Pressman, 1987:363) by the combination of an object and its associated operations" in a single program component" (Pressman, 1987:363). This single component commonly takes the form of an Ada package which may "export" any or all of its data types and operations (Booch, 1987). A typical encapsulated object is depicted in Figure 3-1.

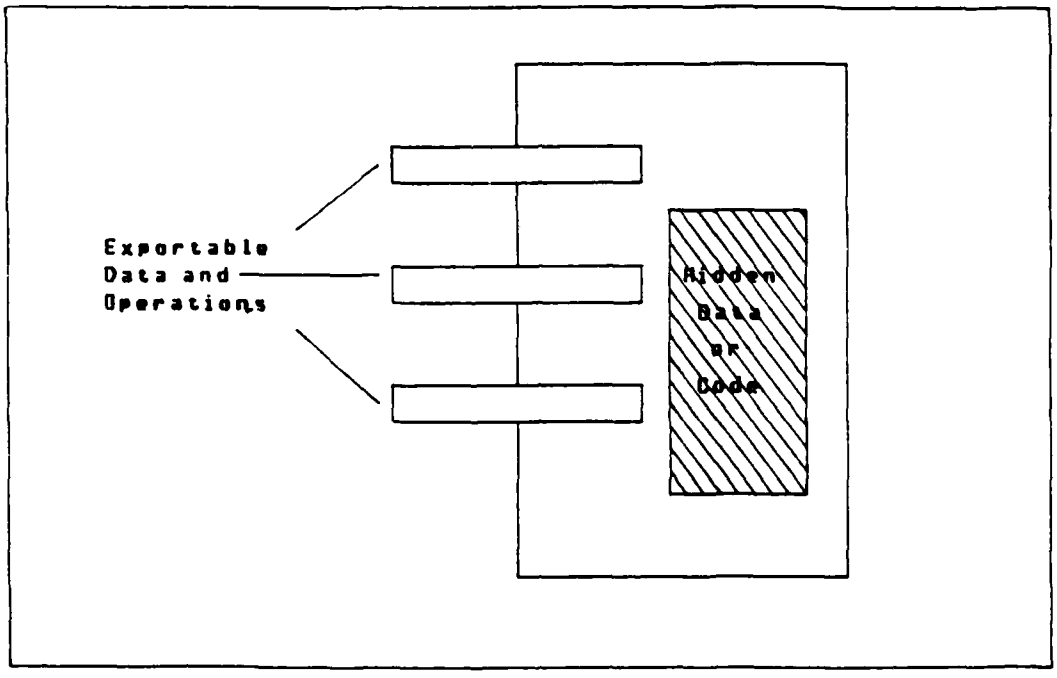

Figure 3-1. OOD Encapsulated Object Representation

After the encapsulation of data and operations into modules, communication or "visibility" between object-modules must be established (Booch, 1987:49). That is, there must be some means by which one object may call or reference another, or by which a programmer can access the operations and data types encapsulated in a module. In Ada, for example, each object package has a "specification" which defines the syntax and semantics for accessing its exportable components (Booch, 1987:55).

As a result of the application of the OOD methodology as abbreviated above, the eventual software structure is determined. It can be depicted as a network of objects, with links representing communication between them (see Figure 3-2). A similar, ordered approach to defining structure is desired between text modules in a hypertext document. This theme will guide the remainder of the chapter.

3.3 An Object-Oriented Approach to Textbase Design

This section of the thesis keys on an abstract view of a CAI tutorial textbase. The textbase will be viewed simply as a collection of discrete text segments, which in the aggregate, through a set of text-to-text links, comprise the entire tutorial.

In the process of imposing structure on the text, the manipulation of text segments will be addressed from an object-oriented point of view. The objects are text segments, the paragraphs and passages of the tutorial. The segments present discussions of the key concepts and expound on relationships between one concept and another, in keeping with Ausubel's approach. The object themes, concept-relationship phrases, will be abstracted as hypertext node names, since the node functions as an access point to the text segment expounding on the concept-relationship duo. Subsequently, these phrases become text object names.

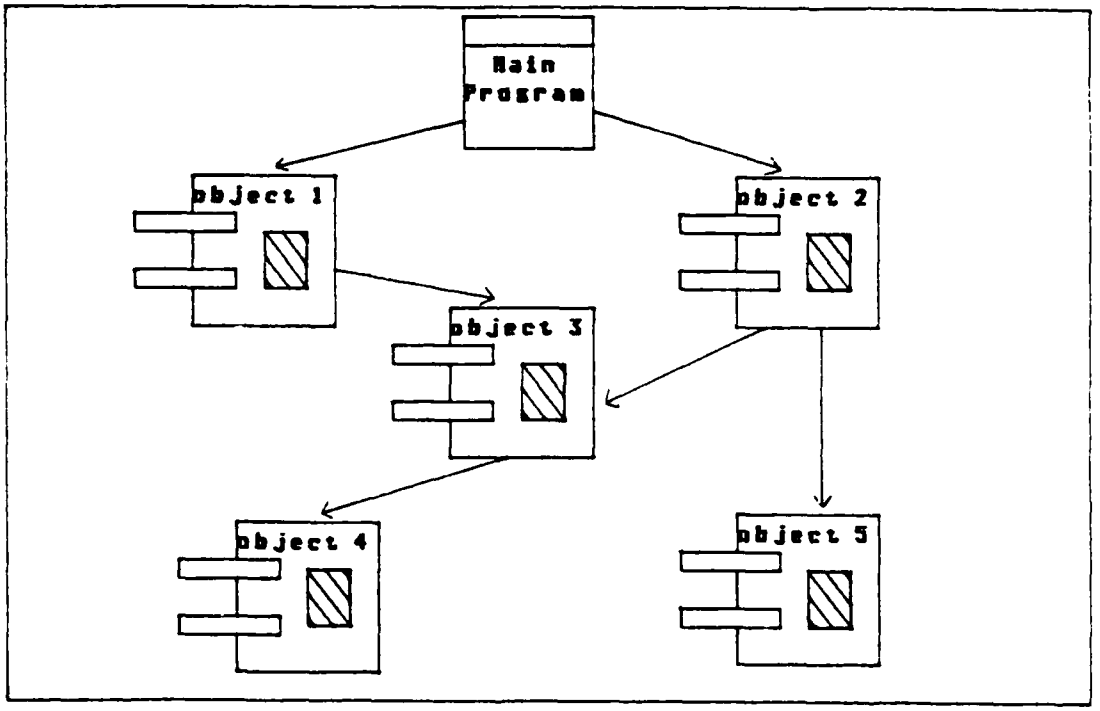

Figure 3-2. OOD Object Visibility Representation

3.3.1 Methodology for Knowledge Base Decomposition

Recall the view of a hyperdocument taken here is a set of text segments connected by hypertext links. This abstract depiction of a textbase opens the door to several lower level questions. How are the text segments created? What level of abstraction should one segment maintain? How many topics can be discussed in each one? What effect does the decomposition process have on the teaching method of the resulting CAI tutorial? These issues are addressed along with the knowledge-base-to-textbase decomposition proposed in this section.

In parallel to OOD, the present methodology provides a mapping from one space to another. Since the product here is an educational tool, the mapping is from an expert knowledge base (a knowledge space) to a tutorial text space. That is, the expert's knowledge is decomposed into text segments which makeup a CAI tutorial textbase.

As in OOD, the decomposition begins at a high level of abstraction. The starting point will be the central concept, the theme of the tutorial. From there the goal is to introduce progressively more detail, less abstract, less inclusive concepts, and more specific examples, through iterative application of the process. In this way, the textbase will take, by default, a hierarchical structure.

Continuing the similarity to OOD, there are formal steps which govern the decomposition process.

1. State the thesis of the tutorial in a single sentence.
2. Expand the sentence into a high level paragraph.
3. Identify the key concepts.
4. Identify the relationships between the key concepts and the paragraph theme.
5. Review and revise.
6. For each new concept repeat steps 1-5 as needed.
7. Encapsulate related concepts in a single object-module.
8. Implement text links in hypertext.

Many of the steps are performed in an OOD-like manner. A complete description of each step is presented below.

1. State the tutorial thesis in a single sentence.

This step is analogous to the first step in the OOD process. The intent is to clarify the theme of the tutorial. This step will establish the highest level of abstraction, the most inclusive concept(s) the tutorial will address.

2. Expand the sentence into a high level paragraph.

As in Step 1, this step nearly parallels its OOD counterpart. The paragraph should be five to nine sentences long (with exceptions noted below), and should maintain approximately the same level of abstraction as the single sentence on which it expands. The objective is to introduce applicable high level terms, principles, etc., which relate to the immediate concept or theme. By the time the paragraph is completed, the scope of the information to follow should be clear to the reader. That is, only

progressively finer detail should be encountered throughout the rest of the tutorial.

In contrast to the OOD method, there is latitude for additional paragraphs, if the tutorial is sufficiently large enough to warrant them. There is an important point to consider which will justify the acceptability of more paragraphs. Since the tutorial structure will be hierarchical, there may be several lower level concepts to explore from the discussion of a single higher level concept. After the reader traverses links to some text several levels of abstraction lower, he must return to higher levels to begin a new path through the document. Consequently, a reader may face the same screenful of text several times as he returns to explore new trails.

Thus, the use of more than one paragraph may help the reader feel a sense of progression through the document and closure of the entire document, even as he returns to the same high level. In any case, the decision for multiple paragraphs in a single display will be left to the judgement of the tutorial designer on an individual basis.

For any paragraph, however, these guidelines apply:

a. Write at a constant level of abstraction.

b. Provide the reader with the highest level, most inclusive concepts that apply at the current level of

abstraction. This allows the reader to view lower level details in relation to these broader concepts.

c. Write to call attention to key concepts which will be given lower level, more detailed explanations later in other segments of text. These concepts, the nouns and objects in the text, will become hypertext nodes.

d. Write to expound on the relationship between the paragraph theme (a concept-relationship pair) and the higher level paragraph in which it was introduced. Elaboration of the details of relationships is the raison d'être for each new text passage.

e. Write to make explicit the relationships between the paragraph theme and each lower level concept introduced in the paragraph. Relationships should take the form of object phrases, with the new concept being the object. The objective is to whet the reader's appetite for more detail, while maintaining the current level of abstraction.

3. Identify the key concepts (from Step 2c). Select only those concepts which are at a level of abstraction consistent with the entire paragraph. These should be the objects in a correctly written paragraph (as per Step 2). Some consideration must be given to the determiniation of which objects are consistent with the current level of

abstraction. It is important to exclude the selection of concepts which, though germane to the topic, are not at the level of abstraction the paragraph represents.

4. Identify relationships between new concepts and the paragraph theme. According to Rule 2e, relationships are articulated as object phrases, i.e., verb phrases which take an object. The objects are the concepts identified in Step 3. Ideally, an entire object phrase will become a hypertext node and a new text object theme. While this step should follow cleanly from well-executed Steps 2 and 3, in some cases adjustments must be made for the sake of clarity and for the final hypertext implementation.

a. For implied relationships between the paragraph theme and a new concept, consider rewriting the passage to make the relationships explicit.

b. For a series of concepts separated by commas, a relationship will be considered explicit for each concept in the series. For example, consider the sentence "The electronic mail facility delivers mail, error messages, and system notices." The explicit relation "delivers" is attached to "error messages" and "system notices", as well as "mail." The three relationships might each warrant their own detailed paragraphs at a lower level, but the relationship "delivers" must be the linking relationship to each.

Consequently, the next level paragraph for each object would key on a "delivers" relationship (see Step 2d).

5. Review and revise. Review each paragraph to ensure a constant abstraction level is maintained internally. Check for concepts which may best be introduced at a different level of abstraction. Determine which, if not all, concepts and relationships selected in Steps 3 and 4 should be expounded on in more detail. Revise paragraphs as necessary to reflect these decisions.

6. For each new concept, repeat Steps 1-5 as needed. Repetition of Step 1 is optional, but is recommended as a good rule for setting the theme of any paragraph (Hodges and Whitten, 1982:347-349). Iterate the process until sufficient detail has been revealed; however, there are trade-offs to consider.

Too much detail could force the reader down a path of numerous links. Even within a hierarchy, some form of disorientation due purely to excessive depth of detail is possible. At this point, the reader may neither understand nor remember the trail of relationships which account for the current level of detail. In addition, with more than a few such excessively long paths, the reader may become exhausted and frustrated. Consequently, he may no longer desire to browse the document, but instead only to finish, thus mitigating the educational effect of the tutorial.

In contrast, too little detail may not sufficiently satisfy the intellectual appetite of the reader. Of course, the academic objective of the tutorial will ultimately govern the depth of knowledge presented. Two or three new links is a suitable average.

7. Encapsulate related concepts in a single object. Recall that in the OOD methodology, the encapsulation of data types and operations specific to a single object provided a modular, abstract representation of a program's structure. One abstract view of an object is as a warehouse of data types and operations which can be used within the object itself or by other objects. In the Ada programming language, this abstraction takes the form of a package or abstract data type (Booch, 1987:218-219). At the same level of abstraction, a modular view may be taken of a hyper-document composed of text objects. Objects encapsulated in this way will be referred to as "object-modules."

8. Implement text links in hypertext. Once the decomposition is complete, the remaining step is to implement the tutorial using hypertext. Using the constructs of a hypertext "language," establish links to connect related paragraphs using the node names selected in Steps 3 and 4. Take advantage of window titles and multiple on-screen windows if the environment provides for them.

3.3.2 Text Object Representation

In the methodology presented in this chapter, an object is a collection of one or more text components, available to other objects or to itself alone. From a hypertext perspective, an encapsulated text object is simply a collection of nodes or link markers. These nodes link to segments of text or graphics, which form the hyperdocument (see Figure 3-3).

It follows from the definition of "hyperdocument" and the concept of hypertext, that the structure of a readable hyperdocument will be determined by the linkages between the

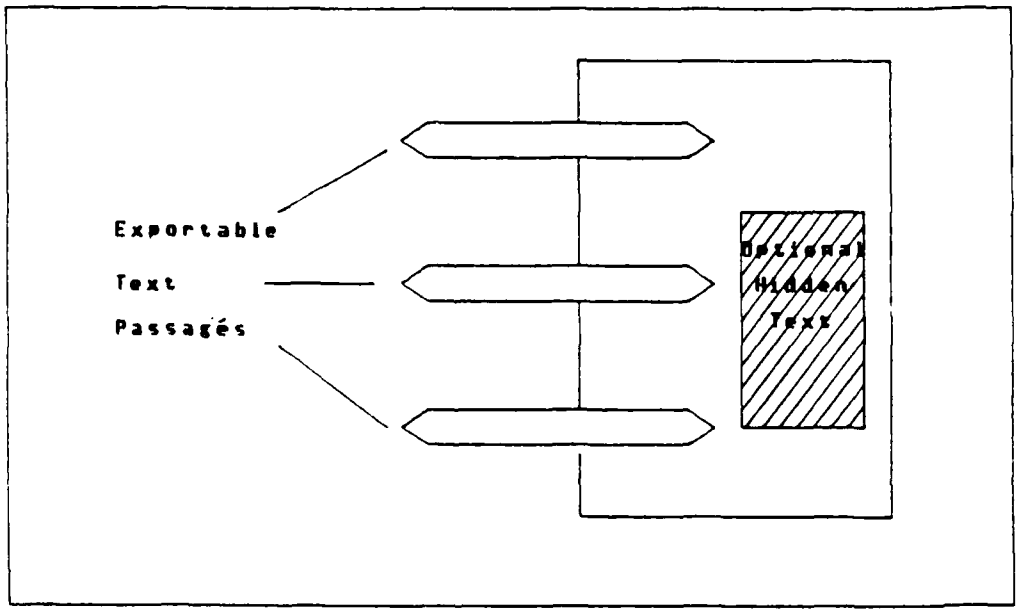

Figure 3-3. Encapsulated Text Object Representation

text objects of which it is composed (see Chapter 2 or Conklin, 1987b:19). In this way, the "packaging" of text in different objects, when it places constraints on links between segments, influences the hyperdocument architecture. The types of constraints considered here are those which conceal one text segment from others, or which make the same segment available to several others. This notion closely models the OOD concept of visibility (Booch, 1987:49) and the software engineering principle of information hiding (Pressman, 1987:336). Figure 3-4 is a depiction of visibility between text objects.

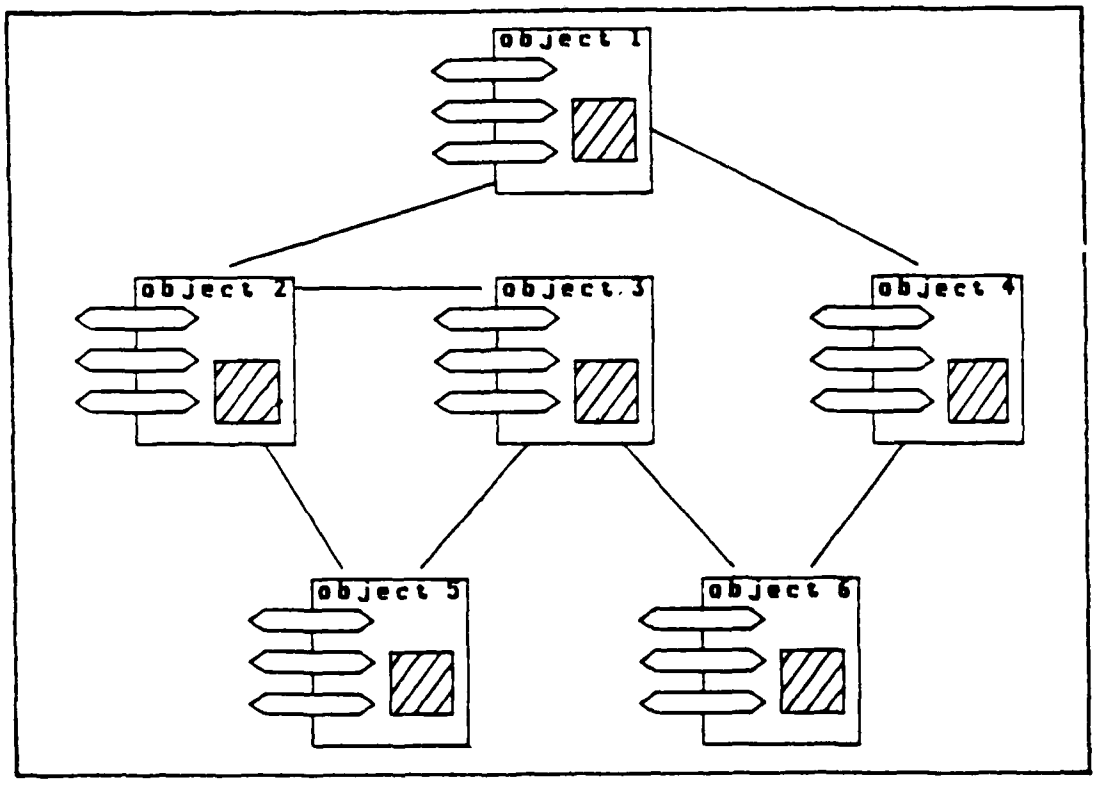

Figure 3-4. Typical Text Object Visibility Structure

This work considers the visibility constraint to be a key to regulating and manipulating the structure of a hypertext document. Now focus shifts to lower levels of abstraction, to the inside of the object-module.

3.3.3 Two Candidate Object Representations

Keying on the influence of visibility on hyperdocument structure, two different, if not antithetical models of an encapsulated text object are considered, each permitting a different visibility structure. This section will detail an object's structure and contents using both models, and discuss their impact on hyperdocument architecture.

3.3.3.1 R-O Model for Text Object Structure. The "Relationship-to-Object" representation views a text object as a module, or object-module, whose subject is a single concept-relationship pair. The subject name is a noun or noun phrase, e.g. "books." The module's callable components are text segments, abstracted as hypertext nodes whose names contain the module's subject.

Nodes are named as object phrases whose object is the module name, e.g. "writes books" and "as opposed to books" (from decomposition Steps 3 and 4). The nodes represent text passages which elaborate on the relationship between the subject of some calling text passage and the object-module, "books" (from Step 2). This representation of an

object will be referred to as the "Relationships-to-Object" (R-O) model, since an object-module's components reflect the relationships of other objects to itself (see Figure 3-5).

Consider as an example of the R-O model, a hyperdocument composed of several text passages (refer to Figure 3-6). One section of text whose subject is "Scholar" contains a hypertext node "writes books." "Writes books" is encapsulated in the "books" object. Similarly, a paragraph with the theme "Hypertext" calls the "books" node named "as opposed to books." In both cases a reader activating either node would be shown (e.g. in a new window) the text which

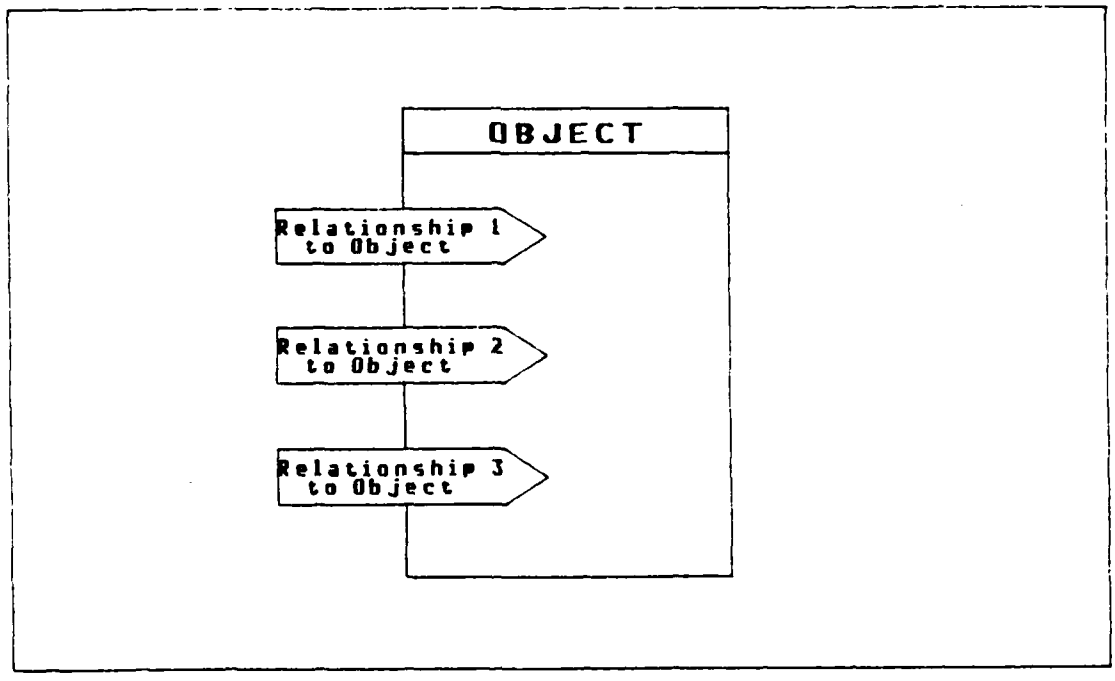

Figure 3-5. R-O Object Model Representation

the nodes represent, which presumably elaborates on the themes of how a scholar writes books and how hypertext is used as opposed to books, respectively.

Continuing the example, let there be other text passages, "collection of books" and "printed in books." These nodes are referenced in paragraphs which discuss "Encyclopedias" and "Photographs," respectively. The objects which encapsulate these text segments, as well as the "Scholar" and "Hypertext" segments, must have visibility to the "books" object-module. In addition, there may easily be visibility between "Hypertext" and "Photographs," based on some (hypothetical) discussion of a relationship between the two concepts.

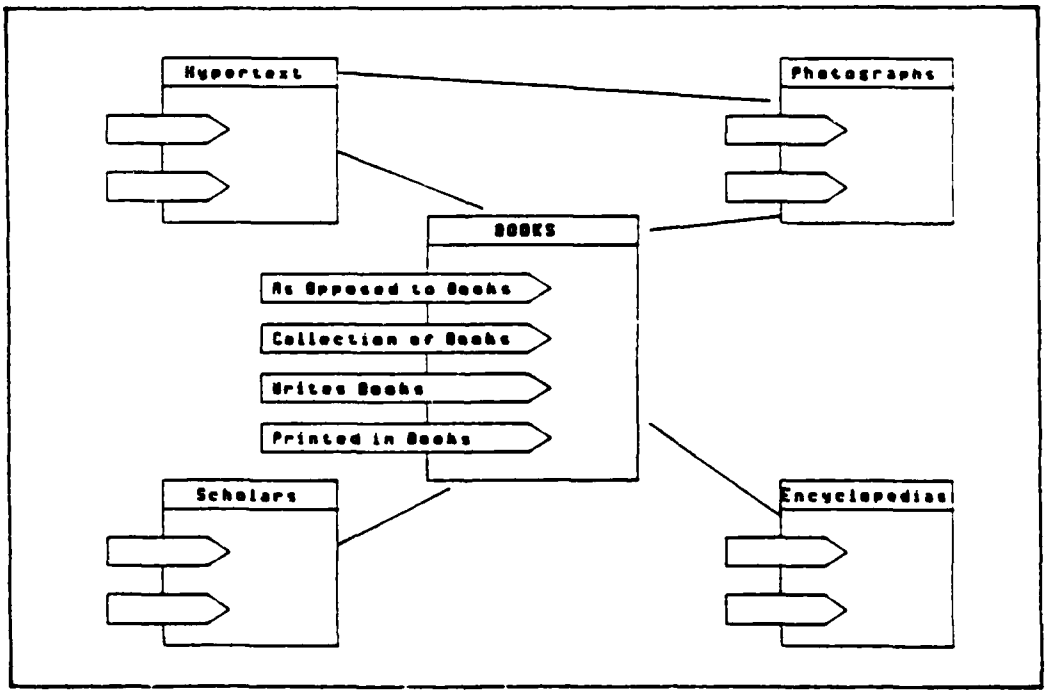

Figure 3-6. R-O Object Model Visibility Structure

3.3.3.2 Evaluation of R-O Model. What are the consequences of the visibility structure of Figure 3-6? From a software engineering perspective, an R-O object parallels almost exactly an OOD object. That is, an R-O object encapsulates a set of components, available to every object to whom they are visible. Consequently, any writer with "access" to the object's contents for the use of one node, could just as easily reference any other node. In the example above, the "Hypertext" paragraph has access to the "printed in books" node and the text to which it is linked, though this may not serve any purpose in the tutorial.

Additionally, as in the "books" example, many objects may require visibility to the same object. This raises the possibility of ambiguity with respect to node names. In the example, suppose another paragraph, entitled "Libraries" also needed to link to a "collection of books" node. Either another "collection of books" paragraph would have to be included in the books object which elaborated the relationship between "Libraries" and "books," or the original "collection of books" paragraph would have to be rewritten to include "Libraries."

In the former solution, how would the two paragraphs, in node representation only, be distinguished one from the other? This situation resembles Raskin's "Monarch" example, and as such reflects a peril of a network structured

hyperdocument (Raskin, 1987). In the latter solution, a reader in the "Encyclopedia" paragraph who activates the "collection of books" node would be forced to read of the significance of "Libraries" as a "collection of books" as well. When the goal of the CAI tutorial is understanding relationships between a couple of concepts at a time, this solution must also be rejected.

The two solutions proposed above assume no sophistication in the hypertext environment which would be able to discern the intended node based on the context in which a particular reference occurred. Indeed, in the unintelligent environments for which this work is targeted, a means is sought where references can be context free, dependent only on the packaging of text segments. To that end, a second object model will be examined in the next section.

Finally, what is the impact of the R-O model on the coupling and cohesion of the text segments encapsulated in a single module? There is generally little or no coupling between individual text segments. However, since visibility between objects is so unrestricted, it is conceivable that one segment in an object could call another segment in the same object. For example, the discussion of "writes books" might reference pictures which are "printed in books," thus coupling the segments "writes books" and "printed in books."

Cohesion in this case is clearly syntagmatic (Hammwöhner and Thiel, 1987). That is, the only reason text passages are included in the same object is that they all share the same grammatical object, by name. This is viewed as poor cohesion, since the text passages as a collection do not combine to edify or clarify the theme "books." Compare this syntagmatic cohesion to the semantic cohesion demonstrated in the O-R object model in the next section.

3.3.3.3 O-R Model for Text Object Structure. The "Object-with-Relationships" (O-R) model is so named because it encapsulates in an object the nodes which express relationships the object has with other objects. In this case, the object has its subject as its name. The subject, a concept in the tutorial, is expressed as a noun or clause containing an object phrase (from Step 4), e.g. "health" or "improves health."

The encapsulated object contains a paragraph and zero or more node names (Figure 3-7). The paragraph references relationships between the subject and other concepts (from decomposition Step 2). The relationships are abstracted as hypertext nodes (from Step 4). In contrast to R-O, the nodes (representing new objects) are visible only from within the object, being referenced only from the object's own text. Clearly, the node visibility structure is nested.

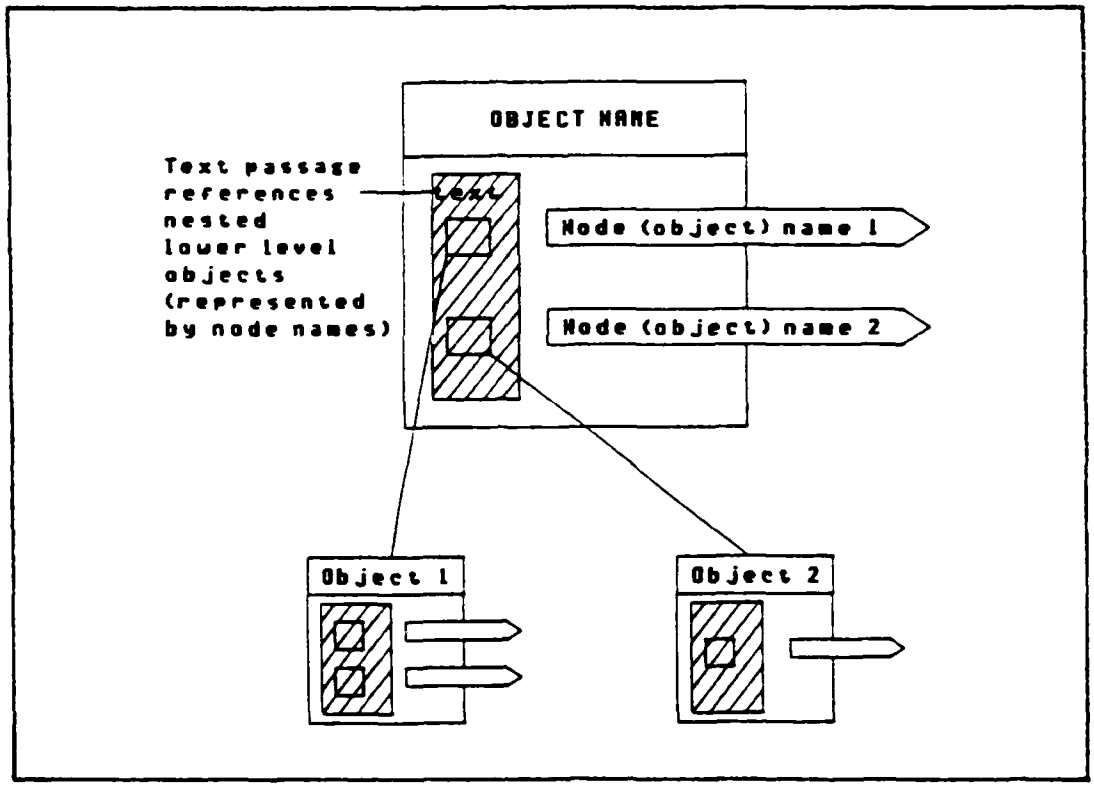

Figure 3-7. O-R Object Model Representation.

As an example of using an O-R model, consider the concept, "seed." "Seed" is the subject of an object, say "Planting Seeds," and may contain nodes "eaten as food," "become plants," and "produced by vegetation." These nodes link to paragraphs which elaborate on the relationship between a "seed" and "food," "plants," and "vegetation," respectively. Notice that the nodes encapsulated in "Planting Seeds" are exactly those nodes to which the "seed" paragraph will make reference. This is a distinguishing characteristic of the O-R model of object representation.

3.3.3.4 Evaluation of O-R Model. What are the consequences of the O-R model? From an OOD standpoint, an O-R object does not function exactly the same as, say, an Ada object. The O-R organization results in a module which demonstrates loose coupling between its components. In the example presented above, the paragraphs on "eaten as food," "become plants", and "produced by vegetation" are independent and can be altered with no effect on each other.

The O-R model demonstrates semantic cohesion. This holds since each component of an O-R module is related, in context and in content, to the module's subject concept. Intuitively, this semantic relationship (Hammwöhner and Thiel, 1987) between each passage and the object's subject is more cohesive than the syntagmatic cohesion of the R-O model. The researcher will refrain from extending the comparison beyond relative differences, however, since the theory of cohesion and coupling of text modules is beyond the scope of this work (see Hammwöhner and Thiel, 1987).

Lastly, and most importantly to this work, is the hierarchical structure which is inherent in the O-R text object representation. In the example above, the "seeds" paragraph could call any of its three subordinate paragraphs, which in turn could call any of their own, and so forth, depending on the number of levels of embedded text. Additionally, this hierarchical structure mitigates

the effects of reader disorientation, since the reader is always firmly within a hierarchical ordering.

3.3.3.5 Contrasting O-R to R-O. With a discussion of both the R-O and O-R object representations complete, a clarification of the distinctions between the two might be helpful. From an abstract perspective, the general structure of the two models was previously presented in Figures 3-5 through 3-7. In addition, a tabular, "compare and contrast" depiction is provided in Table 3-1.

3.3.4 Representation Used in this Thesis

Some of the differences between the O-R and R-O text object representations may seem little more than cosmetic. However, as the evaluation of each model reveals, there are significant differences in the textbase structures the models allow. Depending on the specifics of the hypertext environment used to implement the tutorial, the R-O model may result in a typical hypertext network structure, the pitfalls of which have been presented earlier. On the other hand, the O-R model will enforce a tree-like textbase structure, which we have required from the outset. For this primary reason, the O-R model will be used in the remainder of this thesis.

Table 3-1. Summary of R-O vs. O-R Model Characteristics

Characteristic	Object Model: R-O	Object Model: O-R
contents	node names	text passage and none or more node (object) names
levels of abstraction represented	possibly many	only one
relationship of contents to object name	object name is object of node names	object name is generally (implied) subject of node names
other objects with visibility to contents	possibly many	none
object visibility structure	network	hierarchy
direction of visibility	from others to self	from self to others
intra-object/ inter-text coupling	little or none	none
intra-object/ inter-text cohesion	syntagmatic -- relatively low	semantic -- relatively high
conformity to OOD object	almost exactly	less so
earliest encapsulation	after complete decomposition	after single iteration

3.4 Summary

This this pivotal chapter closes with a review. The need for formal structure in a tutorial textbase was established first. Key motivators were Ausubel's learning theory and some weaknesses of an unchecked hypertext network. Subsequently, the research turned to the field of Software Engineering to borrow the structure produced by the Object-Oriented Design paradigm.

The present knowledge base-to-textbase decomposition is adapted from the OOD methodology. It facilitates an object-oriented approach to the design and construction of a tutorial textbase, for which two alternate object-module representations were presented. The Object-with-Relationships (O-R) model was chosen for use in the remainder of this thesis, since it enforces the hierarchical structure desired of a well-structured hypertext textbase.

4. Example Knowledge Base Decomposition

4.1 Introduction

This chapter presents an example knowledge base decomposition using the methodology described in Chapter 3. Recall that from that chapter, the steps of the methodology:

1. State the thesis of the tutorial in a single sentence.
2. Expand the sentence into a high-level paragraph.
3. Identify the key concepts.
4. Identify the relationships between the key concepts and the paragraph theme.
5. Review and revise.
6. For each new concept repeat steps 1-5 as needed.
7. Encapsulate related concepts in a single object.
8. Implement text links in hypertext.

The rest of the chapter is a step-by-step application of these decomposition steps to a knowledge base. The result will be a set of text passages which are incorporated into a hypertext-based computer-assisted tutorial. Note: The decomposition presented in this chapter is primarily for illustrative purposes. Some lowest levels paragraphs will be omitted here, but will appear in the final tutorial (see Appendix B).

4.2 An Example Using the Decomposition Methodology

The tutorial topic is the Ada programming language. More specifically, the tutorial describes Ada with respect to elements which distinguish it from most other programming

languages. The purpose of the tutorial is to help the reader assimilate the relevant concepts and the hierarchical relationship structure of those concepts. The "expert" knowledge base for the decomposition is an article entitled "Why Ada is Not Just Another Programming Language" (Sammet, 1986). This article was selected because it discusses Ada's distinguishing characteristics in multiple levels of detail, which the decomposition methodology will extract. The contents of the article are excerpted and paraphrased in the passages created below. Additional citation credit will not be given in the remainder of the decomposition.

4.2.1 State the tutorial thesis in a single sentence

> Ada can be distinguished from the majority of other computer languages on many technical and nontechnical merits.

4.2.2 Expand the sentence into a high-level paragraph

> Ada can be distinguished from the majority of other computer languages on many technical and nontechnical merits. On non-technical grounds, the ordered design and development processes which produced Ada are unlike those of any other programming language. Additionally, Ada has attracted strong interest from users and enthusiasts in the international computing community, outside the Department of Defense. With regard to key unique attributes of its technical elements, Ada offers high-level programming features which contribute to Ada's support for software engineering principles.

4.2.3 Identify the key concepts

The concepts introduced in the high level paragraph are shown underlined. They are essentially the nouns and objects in the text. Keeping in mind that a high level of

abstraction is sought early, only a subset of those concepts qualify as high-level concepts. That is, although the passage may contain many concepts, only a few are at the level of abstraction consistent with the focus of the paragraph. The subset of key concepts can be narrowed to:

design and development processes
strong interest
high-level programming features
software engineering principles

The selection is not arbitrary. Notice that although "language" is an abstract concept, it does not exhibit a relationship with the theme of why Ada is different from other languages. The same holds for "Department of Defense," which, although an essential high-level concept, is not expressed in a relationship which edifies the paragraph's theme. Thus, abstraction maintenance sometimes requires carefully considering the role of a concept in a paragraph, even in the whole document.

4.2.4 Identify the relationships between the key concepts and the paragraph theme

Following up on the results of Step 3, identification of the relationships of each concept to the paragraph theme requires little more than capturing the object phrases in which they appear. In this case, the result is:

OBJECT	RELATIONSHIP
processes	processes which produced Ada
strong interest	Ada has attracted strong interest
programming features	Ada offers high level...features
principles	Ada's support for...principles

As a point of clarification, the first relationship may seem at first glance to be improper. Indeed, it does appear reversed, with the theme, "Ada," as the object and the object, "processes," as the subject. This apparent breach of the methodology is accepted for now, and left for Step 5, where such issues are addressed. The remaining relationships appear straightforward, and clearly relate their respective objects to the paragraph theme.

4.2.5 Review and Revise

At this point, the tutorial theme has been clearly stated and the scope of the tutorial has been bounded in a single paragraph -- or have they? This is the step in the process reserved for sufficient review to answer such questions. Should more concepts be added? Should the main paragraph be reworded to make the relationship between Ada and the Department of Defense explicit? In other words, has this paragraph established here everywhere it intends to take the reader for the rest of the tutorial? Do only lower level concepts, details, elaborations, and explanations remain? There should be no high-level surprises for the reader later in the document.

What about the selection of key concepts and their relationship to the subject? Are they firm? No. The question of the "process which produced Ada" relationship was raised earlier. Technically, the sentence could read "Ada was produced by a unique design and development process." Although the sentence could be reworded to correct this transposition, in this case it was not. The wording was chosen for the sake of style and readability, and to bring up this point: ideally, every object selected should actually be a grammatical object; however, variety in sentence structure and readability sometimes dictate otherwise.

4.2.6 Iterate the Process for each new concept

Before any attempt is made at encapsulation, the process is iterated for each selected concept. However, it is noteworthy that at this point in the process, putting off encapsulation is not entirely necessary. The reason is that under the Object-with-Relationship (O-R) model, the top-level object titled "Ada is different" could be created already. It would include nodes named for the four relationships listed in Step 4. The convenience of as-you-go encapsulation is an advantage of the O-R object model. However, in keeping with the methodology, all encapsulations will be saved for the stated place in the sequence of steps.

Before beginning iteration of the decomposition, a decision must be made whether to proceed in a depth-first or breadth-first fashion. Since a depth-first approach will pursue a particular concept to its greatest level of detail before starting another decomposition, a depth-first decomposition is appropriate. Therefore, the decomposition begins with the "processes which produced Ada" relationship, extracting progressively more detail througout the process.

The Second Iteration. For convenience, and as a rule of thumb, the one sentence topic statement (Step 1) will be just an adaptation of the sentence used in the higher level paragraph. That is:

> The ordered design and development processes which produced Ada are unlike those of any other programming language.

Now expand the sentence to a full paragraph (Step 2) which will bound the next lower level of detail. The result is:

> The Ada design and development process initially resulted from a software crisis in the DoD. Uniquely, the process began with a formal requirements statement for a language to meet the DoD's embedded system software needs. Also unique was the fact that the initial language design was contracted to competitive bidders, some from outside the U.S. and developed by a corporate team, as opposed to the typical committee or individual effort.
>
> The DoD also established administrative controls to govern implementation of the language product, both its syntax and its compilers. And to a degree that had not been experienced before, through all of these processes, the DoD encouraged public feedback, actually implementing valid recommendations when possible.

From the text passage, these new objects (Step 3) and relationships (Step 4) are the result:

OBJECT	RELATIONSHIP
software crisis	resulted from...software crisis
requirements statement	began with...statement
competitive bidders	contracted to competitive bidders
corporate team	developed by a corporate team
administrative controls	established administrative controls
public feedback	encouraged public feedback

Assuming that the new concepts and their relationships are a sufficient set and are satisfactorily worded (Step 5), the decomposition process begins again (Step 6), proceeding depth first from the "resulted from a software crisis" relationship. The topic sentence (Step 1) is simply:

> An impetus for the creation of Ada was a software crisis within the DoD.

The elaborative paragraph (Step 2) becomes:

> An impetus for the creation of Ada was a software crisis within the DoD. The crisis was recognized in 1974 when a DoD report was issued, estimating a software development and maintenance cost of over three billion dollars. The DoD crisis was caused by a proliferation of languages within the department, which severely restricted reusability of both software and programmers.

The single new relationship (Steps 3 and 4) is expressed in this next lower-level paragraph:

> The DoD recognized a proliferation of programming languages as a cause of its software crisis. The DoD estimated there were literally hundreds of programming languages and dialects in use on DoD projects. Each

new language required its own contracted maintenance teams, which accounted for the astronomical cost.

Since no lower-level concepts were selected from this paragraph for further decomposition, this portion of the depth-first process ends. Note that even after having reached a "dead end" in the decomposition process, the subordinate relationship-concept pairs are not yet encapsulated within their parent objects. Although with the O-R model, encapsulation is now possible, this action is deferred until near the end of the chapter. The process continues again from the top of the chain of relationships.

The next relationship to expound on is the "process began with...requirements statement" relationship. It is the second of the "processes which produced Ada" paragraph's new relationships. The topic sentence is:

> Uniquely, the design for the Ada language was based on a set of formal language requirement specifications.

Expanding the sentence into a more detailed paragraph:

> The design for Ada was based on language requirement specifications originally established in the DoD's STRAWMAN document. The intent of the 1975 document was to detail specifications for a language to be used in DoD embedded computer systems. After some revision, the document was re-released as WOODENMAN in early 1976. Later that year, the newer TINMAN revision was released. TINMAN was used as a basis for <u>evaluating existing languages</u> for suitability as the new standard DoD language. TINMAN later underwent revision to become the 1978 IRONMAN document. The IRONMAN requirements were used at the time the design and development process formally began, but a still later

revision, STEELMAN, was the final version which governed the language development.

The single new relationship, "used as a basis for evaluating existing languages," becomes:

> The TINMAN document was used as a basis for evaluating existing languages for suitability as the DoD's single language for use in DoD software systems. Eventually, ALGOL68, Pascal, and PL/I were selected as baseline languages from which the new language would be patterned, but from which it need not be compatible. Among those considered were:
>
> DoD/Embedded Computer Languages
> CMS-2, CS-4, HAL/S, JOVIAL, J73, SPL/1, TACPOL
>
> Process Control/Embedded Computer Languages (Europe)
> CORAL66, LIS, LTR, PEARL, PDL2, RTL/2
>
> Research-Oriented Languages
> ECL, EL-1, EUCLID, MORAL
>
> Widely Used/General Languages
> ALGOL60/68, COBOL, FORTRAN, Pascal, PL/I, SIMULA67

With no new relationships selected from this paragraph, the decomposition begins again at a higher level, returning to the "processes which produced Ada" paragraph. The next subtopic to expand is the "design contracted to competitive bidders" relationship. The topic sentence is:

> Unique to Ada, the design for the language was offered to several competing contractors who were to come up with independent language designs.

And the expanded paragraph is:

> Unique to Ada, the design for the language was offered to several competing contractors who were to come up with independent language designs. Of over 15 original bidders, only four were selected to continue. In an effort to keep the design evaluations as unbiased as possible, each design was referred to only by a color (red, green, blue, or yellow). After the four preliminary designs were reviewed by both military and civilian agencies, the Red team (Intermetrics) and Green team (Honeywell Bull) were selected to continue designing. In the end, the french based Honeywell Bull team won the development contract.

Since no new lower detail is pointed to, as before, the process returns again to the parent passage to begin a new depth-first decomposition.

The fifth relationship from the "processes..." passage is "developed by a full corporate team." Dispensing this time with the topic sentence (since it is optional), the new paragraph is:

> In contrast to the "normal" development by an individual or committee, Ada was developed by an entire team of programmers from the France-based Honeywell Bull corporation. Design decisions were made or approved by the team leader, Jean Ichbiah.

Continuing the process, again without the topic sentence, the next relationship to expand is "established administrative controls."

> The DoD placed administrative controls on the language to prevent the same types of language non-portability problems it currently faced. Additionally, the DoD intended to save Ada from the fate of languages like FORTRAN and JOVIAL, which have numerous versions and

dialects. To control growth of and changes to the language syntax, the DoD early in the process secured an Ada trademark and forbade subsets or supersets of the language.

On a larger scale, and to provide for controls on language implementation, the DoD set requirements for compilers including their development and validation. Lastly, the DoD planned for a programming support environment, nearly from the outset. It is the consideration of these issues so early in the design and development processes which contributes to Ada's distinction from other languages.

The new relationships were revealed in this passage are listed below. Since this example is not intended to be exhaustive, and since the complete decomposition is presented in Appendix B, the decomposition of these new relationships will not be continued here.

CONCEPT	RELATIONSHIP
Ada trademark	secured an Ada trademark
supersets/subsets	forbade Ada supersets/subsets
compilers	set requirements for compilers
environment	planned for...support environment

Rounding out the "processes which produced Ada" paragraph, the "encouraged public feedback" relationship, the sixth concept-relationship pair selected from the parent paragraph, is detailed. The new paragraph is:

A distinguishing element of the Ada design and development processes was that throughout, the DoD solicited and considered review and comment from the international programming community. Public comments were used in the series of revisions from STRAWMAN to STEELMAN and in the evaluation of candidate language designs. Also, the prototypical Preliminary Ada was published in ACM SIGPLAN for commentary and review.

Since no new relationships were selected from this paragraph, and since this was the last object in the "processes which produced Ada" passage, the process moves up the chain of text to the original highest level paragraph to pursue a new branch of the hierarchy of concepts.

The Third Iteration. The next higher level relationship to elaborate on is the "Ada has attracted strong interest..." relationship. The topic sentence is:

> Unlike other programming languages, Ada has attracted strong interest from the international military and civilian programming communities.

The more detailed paragraph is:

> Unlike other DoD sponsored programming languages, (e.g. JOVIAL, CMS-2, and TACPOL), Ada has attracted strong interest from the international programming community. Ada has been utilized in commercial and industrial sectors in the U.S. and in military and commercial applications in Europe. Additionally, Ada is the only language which has sparked publications and conferences which are dedicated solely to the advancement of the language.

The new objects and relationships are:

OBJECT	RELATIONSHIP
commercial sector	accepted in the commercial sector
industrial sector	accepted in the industrial sector
Europe	accepted in Europe
conferences	sparked the establishment...conferences

The paragraphs associated with each of these new relationships are presented in succession below, with little introduction.

Utilized in commercial...

> Ada's wide acceptance has made it of interest to the international commercial sector. Ada compilers, PC versions, and software packages are becoming commercial items, available via bulletin boards as well as through commercial software vendors.

Utilized in industrial...

> Ada is being used in industrial applications which are not related to military projects. Its readability coupled with facilities for numerical and data processing have gained Ada a foothold in industry, in both embedded and non-embedded software systems applications.

Accepted in Europe...

> Ada's applicability to embedded software systems accounts for its growing popularity in foreign militaries. European and NATO countries are beginning to use Ada in embedded weapons systems and for research.

Sparked the establishment of...conferences...

> Ada is the subject of several publications and special interest groups. ACM SIGAda is dedicated to Ada applications and research. In addition, Ada has to its credit international conferences and quarterly special interest group meetings. No other language has received this intense international attention.

The Fourth Iteration. Since none of the above paragraphs introduced any new concepts to be further elaborated, the methodology returns attention to the main paragraph to begin detailing the "Ada offers...high-level features" relationship. The topic sentence is:

> From a technical perspective, Ada offers many specific high-level programming features provided by no other single language.

The extended paragraph becomes:

> From a technical perspective, Ada offers many specific high-level programming features provided by no other single language. Ada combines some of the features and constructs of Pascal, ALGOL, and PL/I, with some of its own unique features. The most important of Ada's features is the facility of packaging. Ada also offers:
>
> strong data typing
> generics
> tasking
> numeric processing
> representation clauses
> real-time processing
> exceptions
> overloading
> separate compilation

This high-level paragraph does not reference any new concepts that are within the scope of the tutorial, so we return to the highest level paragraph and begin again.

The Fifth Iteration. The last remaining object-relationship in the highest level paragraph is "support for software engineering principles." The topic sentence is:

> In many ways, Ada directly supports the principles of software engineering.

The more detailed paragraph is:

> In many ways, Ada directly supports the principles of software engineering. Ada is designed around the software component, as reflected in the modular properties of separately compilable packages and subprograms. Ada supports abstraction and information hiding by providing strong data typing and private types. These constructs govern visibility or access to code. Ada generics and stubbed package specifications lend support to reusability and modularity concerns. Overall, Ada supports many software engineering desirables including:

structured programming	reusability
top-down development	modularity
strong data typing	portability
abstraction	readability
information hiding	verifiability
encapsulation	
separately compilable specification and body	

While included in the original knowledge base (Sammet, 1986), further elaborations on Ada's support for each of the listed software engineering principles are beyond the scope of this tutorial. Therefore, no further details will be extracted.

4.2.7 Encapsulate the Objects and Relationships

As referenced earlier, under the constraints of the O-R object model, encapsulation is a simple matter. A single object contains its theme passage and the node names of the objects it references. In the example presented in this chapter, including the sections deferred to Appendix B, the last objects elaborated on in each depth-first decomposition will be encapsulated first. In the end, the main module will hierarchically encapsulate all others. The encapsulated objects for this tutorial are depicted in Appendix A from the highest level down to the lowest.

4.2.8 Establish Text Links in Hypertext

The actual hypertext nodes (link markers) are taken directly from the concept-relationship pairs selected for further decomposition during each iteration of the process.

For example, in the main paragraph, link markers bear the names:

processes which produced Ada
attracted strong interest
offers high-level programming features
support for software engineering principles

The implementation of this example tutorial used the facilities provided in the KnowledgePro expert system environment. KnowledgePro offers the capability to title the new windows which appear when nodes are activated. In the hypertext implementation of this tutorial, the window titles were used to emphasize the relationships which are expounded on in the window text. For example, in the window which was opened through the "processes which produced Ada" node, the window title was "Unique Design and Development Process." In this manner, the titles of embedded windows express the chain of relationships which have led to the current window (see Figure 4-1).

Additionally, a brief introductory instructional session was incorporated into the finished tutorial product. Its intent was to introduce the student to the peculiarities of the KnowledgePro environment, and give the reader practice at selecting and activating nodes, as well as opening and closing windows.

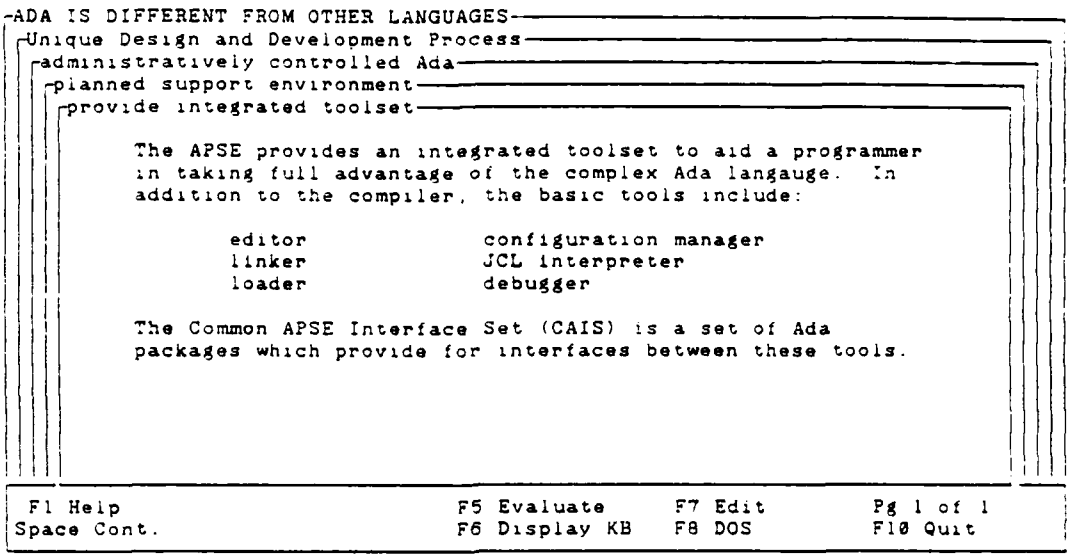

Figure 4-1. Use of Window Titles to Re-enforce Relationships.

4.3 Summary

This chapter has presented an example knowledge base decomposition using the methodology proposed in Chapter 3. The actual textbase of the tutorial product is included in Appendix B. By actually stepping through the formal process, we have gained an appreciation for some of the exceptions which must be made when actually putting the methodology to use. In sum, 1) object phrases must sometimes be worded for clarity and variety, despite an apparent conflict with the rules of the methodology; 2) it

is not always the case that every concept be detailed to numerous lower levels of abstraction; and 3) sometimes the elaboration of a seemingly germane concept must be postponed to a lower level, where it fits in a more explicit relationship to some other concept at a more appropriate level of abstraction or detail.

5. Validation of the Decomposition Methodology

5.1 Introduction

This chapter presents the evaluation of the tutorial design methodology presented in Chapter 3, and demonstrated by example in Chapter 4. In validating the mechanized process, whose end product applies to the cognitive domain, both objective and subjective measures were necessary. The validation process used in this chapter involved four evaluation tools, which in the aggregate provided both types of measures. Three of the four validation procedures were accomplished through an experiment involving graduate students. The remaining evaluation enlisted the attention and expertise of two graduate instructors. The format and results of each of these evaluation procedures are discussed in the remainder of the chapter.

5.2 Data Extraction by Experiment

This section describes the set up and execution of the experiment involving a single class section of graduate computer science students. Collection of data for all but the first of the evaluation tools was directly linked to the experiment. Discussion of the single exception is deferred until the next major section, when all four tools are discussed in detail.

5.2.1 Experimental Design Overview

The one-time experiment was designed around Campbell and Stanley's "Pretest-Posttest Control Group Design" (Campbell and Stanley, 1963:13-24). The basic experimental model requires random assignment of subjects to either a control or experimental group. Random assignment facilitates pre-treatment equivalency. All subjects in each group are given a pretest, the control or experimental treatment (as applies), and a posttest. The variable of statistical interest is the average gain score for each group. Assuming the key threats to the validity of the experiment have been controlled or eliminated (Campbell and Stanley, 1963:13-24), the experimental group is expected to show a statistically significant increase in the pretest-posttest gain score, over that of the control group.

5.2.2 Execution of the Experiment

The subjects in the experiment were a single class section of students. The whole process was conducted over two class periods and a laboratory session. Both the testing and the administration of the control and experimental treatments were conducted as routine class assignments. This was done to mitigate the "Hawthorne effect," i.e., to remove any bias associated with the subjects' knowledge of participation in an experiment (Woolfolk and McCune-Nicolich, 1984:36).

Random Assignment. Ten members each of the class of 20 students were randomly assigned to either the control or experimental group using the simple randomization process outlined below. The random numbers used in the assignment routine came directly from a table of random numbers (CRC, 1973:629-632).

1. Proceed sequentially across the rows of the random number table and down an alphabetized list of students, assigning a random number to each student.
2. Determine the median of the random numbers assigned to students in the list.
3. Assign students whose random number is below the median to the control group.
4. Assign students whose random number is equal to or above the median to the experimental group.

Assigning students from the same class section in this manner satisfied the requirement for randomization. Note: in keeping with a conscious effort to mask the presence of an experiment, students were neither informed of their assignment to any group, nor that such groups existed, until after the posttest phase of the experiment had been completed.

Pretest. The pretest was given by the class instructor in lieu of a regularly scheduled in-class feedback quiz. Students were given the assignment explained in Section 5.3.2.

Treatments. The control and experimental treatments were administered by the course instructor in a regular laboratory following the class in which the pretests were given. Both treatments were computer-based versions of the Ada-related article used as a knowledge base in Chapter 4 (Sammet, 1986). The control treatment was several main sections from the article entered directly into a computer file, which could be read only in a strict "page-up," "page-down" manner. The sections used were those whose subject content was also represented in the hypertext-based tutorial, the experimental treatment. The experimental group was asked to use the hypertext-based tutorial, as designed via the Chapter 3 methodology. (Note: the content equivalency of the two versions was verified and is discussed in the next main section). Only one group treatment was administered at a time, and each group of students was unaware that its treatment differed from that of the other group.

Posttests. The posttest and two additional measures were given to the students in the following class meeting, four days later. The contents and intent of each of these evaluation tools is provided in the next section.

5.3 Evaluation Tools and Their Uses

It is important to clarify at the beginning of this section that no formal process was undertaken to validate the correctness and completeness of the steps which comprise the decomposition process. This apparent breach is not viewed as a deficiency in the overall evaluation. The justification of the omission is that the decomposition and design process is itself based largely on the proven Object-Oriented Design (OOD) paradigm, as discussed in Chapter 3. (For a more rigorous discussion of objects and OOD, see Pressman, 1987 or Bralick, 1988). Instead, attention was turned to the evaluation of the effects of the methodology's end product, the hierarchically structured, relationship-driven tutorial.

5.3.1 Pre-experiment Content Equivalency Check

The four-part evaluation process began with a validity check on the content of the tutorial developed in Chapter 4. Two instructors, considered experts on the tutorial subject matter, were given a copy of the hypertext-based tutorial and the original article from which it was derived (see Sammet, 1986). Through a series of questions (see Figure 5-1), the instructors evaluated whether the non-linear tutorial version and the linear printed version covered the same material. In addition, the instructors were asked to provide comments with respect to

Computer-Based Tutorial Validation Questionnaire

1. INSTRUCTORS: Your assistance is required for a portion of the validation of a computer-based tutorial. The subject matter is the Ada programming language. (Lt. Talbert will assist you in accessing the tutorial).

2. Please read the attached article, "Why Ada is Not Just Another Programming Language" by Jean Sammet. Then read through the tutorial whose information content was taken from the article.

3. After reading the two versions, please thoughtfully answer the questions presented below. Please return your responses to Lt. Mike Talbert by 26 Aug 88.

QUESTIONS

1. Were the concepts at the highest level of abstraction (most encompassing concepts/ideas) consistent between the two media? Be specific if you do not answer "yes."

2. Did the computer-based tutorial present the same level of detail as the article, especially with regard to the topics addressed, examples, illustrations, etc.? If not, please be specific in your response.

3. How did you perceive the physical presentation of the tutorial impacts a reader with respect to ease of reading, visual effect of the video display terminal, color, reverse video, etc.?

4. What SINGLE recommendation would you make for improving the tutorial, with regard to issues not addressed in questions 1-3?

Figure 5-1. Tutorial vs. Article Content-Equivalency Questionnaire

the understandability, ease of reading, and physical presentation (e.g. color, typography) of the computer-based tutorial. The instructors' recommended changes were incorporated into what became the final version of the tutorial.

5.3.2 Pretest and Posttest Concept Diagrams

The second, and primary method of evaluation was, like the first, a combination objective/subjective measure. Its purpose was to evaluate what effect the tutorial, i.e. the experimental treatment, had on the students' method of structuring the concepts presented in it. The question of interest was: did the group who viewed the non-linear tutorial begin, with the posttest, to structure concepts in a hierarchy of abstraction, as was the structure reflected in the tutorial? If so, did they do so more than did the group who read the article in a linear fashion, without the emphasis on a hierarchy of relationships?

As a source of numerical and subjective data from which to derive an answer to the above questions, the students were given the pretest and posttest shown in Figure 5-2. This concept diagram approach to testing, personally recommended by Gowin (1988) for the present work, provides a "snapshot" of the students' understanding of the relationships among a set of concepts, both before and after the treatments (Novak and Gowin 1986:93-108). In this case,

IN-CLASS ASSIGNMENT

1. Consider the following alphabetical list of concepts which are related to Ada, with respect to Ada's distinctive place among the many other computer programming languages.

Ada conferences
Ada publications
Ada subsets
Ada supersets
Ada trademark
administrative controls
commercial sector
competitive bidding
compiler specifications/validation
corporate language development team
design/development process
DoD software crisis
existing programming languages
formal language requirements
high-level features
industrial sector
integrated toolset
proliferation of programming languages
programming environment
public involvement/feedback
software engineering principles
STONEMAN document
strong interest outside DoD
TINMAN document

2. Using the node provided below as a start, hierarchically organize the concepts in a concept map as you perceive or understand the relationships between them. Place more broad concepts near the top and more detailed concepts near the bottom of the diagram. Use explicit relationships between concepts as links in the concept map.

Ada differs
from other languages

3. Draw your concept diagram on a separate sheet of paper.

4. Turn in both your diagram and the assignment handout.

Figure 5-2. Pretest and Posttest In-Class Assignment

the concepts of interest were those listed on the tests. Their concept diagrams were compared to a master diagram (Figure 5-3), which depicts the concepts and their relationships as expressed in the tutorial. Scores were awarded to the diagrams on a node-by-node basis, with respect to how the students' diagrams resembled the master.

Concept Diagram Scoring Scheme. There exists little guidance on objective evaluation of subjective concept diagrams, but Novak and Gowin have outlined a two-dimensional scheme which combines both objective and subjective measures (Novak and Gowin, 1984:105-108). The dimensions are 1) the placement of a concept in the hierarchy and 2) the wording of the relationship between two concepts. The scoring methodology used for this thesis is a variation on Novak and Gowin's theme.

In accordance with Novak and Gowin, the first dimension receives a multiplicatively higher weight than the second (Novak and Gowin, 1896:107). In this case, the relative placement of concepts in the hierarchy receives a weight between three and five times that of the relationship wording. Specifically, with an arbitrarily chosen maximum of 30 points for each relationship-concept pair, node placement is scored from 0-25 points (in values of 0, 5, 15, 20, and 25). Relationship wording is scored from 0-5 points (in values of 0, 1, 3, and 5).

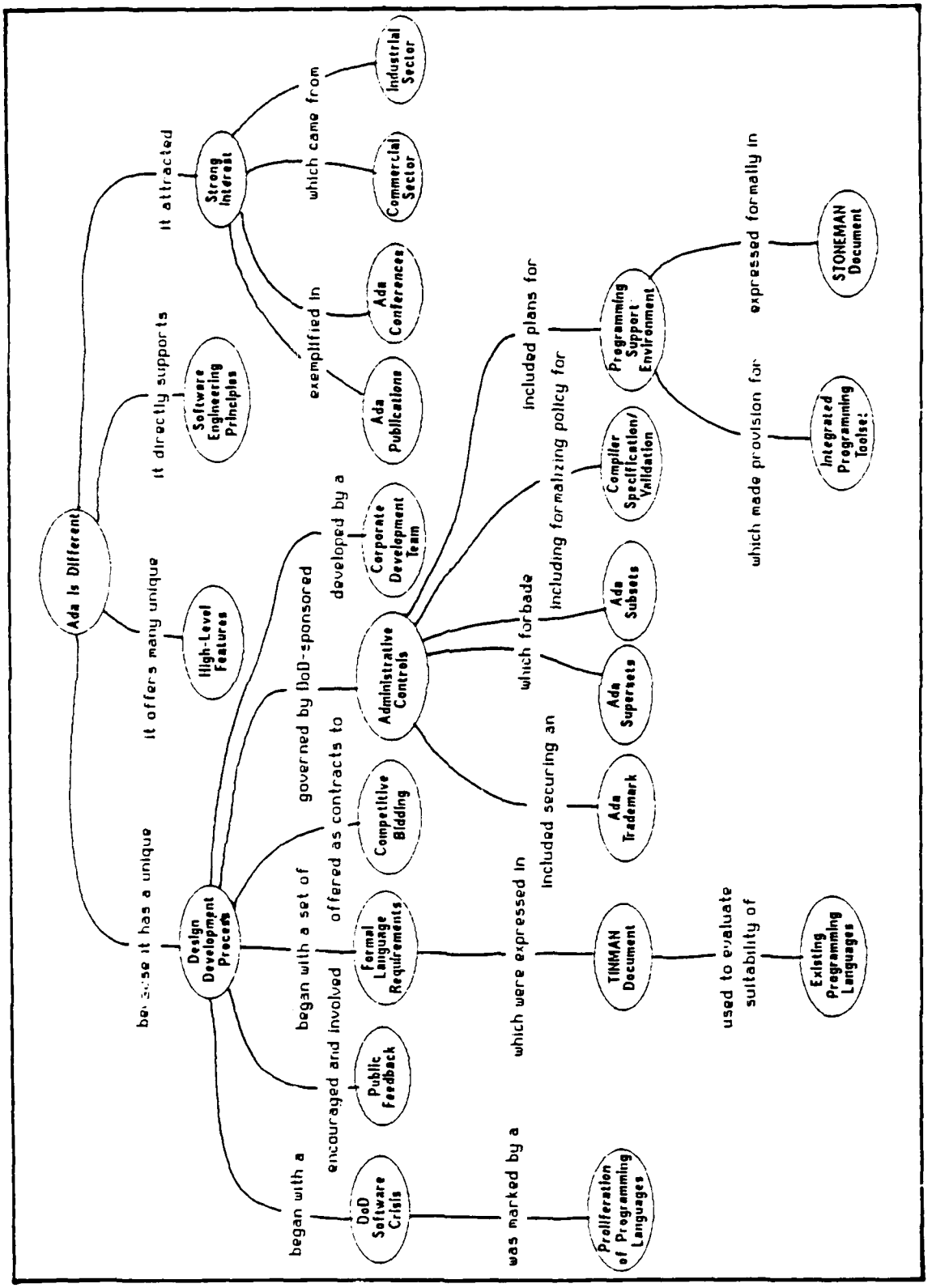

Figure 5-3. Master Concept Diagram

In general, the scoring methodology compares the placement of a concept beneath a parent concept in the hierarchy to the same placement in the master diagram. Highest point values are assigned to exact parent-child placement, and lower values are given for approximations (inverting parent and child or leaving out an intermediate node). These approximations are incorporated in the scheme to allow for instances where students recognize that a concept is related to another, but do not represent the relationship in the correct hierarchy of abstraction.

Other incorrect node placement is judged subjectively; three possibilities are included in this scoring system. If the relationship is true but not expressly stated in the treatment readings, it receives five points. If it is logical based on a misinterpretation or perceived ambiguity of the concept wording as expressed in the pretest and posttest, then five points is also awarded. The award of 5 points for these logical errors, as compared to 15-25 points for exactness or approximation errors, will provide a numerical basis for indicating whether a student has received and begun to recognize hierarchical relationships between concepts by the time of the posttest, as a result of the treatment. Lastly, for concept placement that is blatantly wrong or omitted entirely, a score of zero is

given. The algorithm by which a numerical score is assigned to a concept node is provided in Nassi-Shneiderman format in Figure 5-4.

Relationship scoring is less stringent, since exact wording of relationships between nodes is not as easily reproduced as is exact placement of nodes. For relationships which have been omitted or express a wrong or illogical relationship (an admittedly subjective measure) between two correctly placed concept nodes, zero points is given. For any attempt at all at expressing a relationship, even "e.g.," "is a," etc., one point is scored. For a more

For each node in diagram

Concept node position is

exactly same as in master concept diagram	missing inter-mediate node	inverted with respect to parent	incorrect but logical	wholly incorrect or missing
score is 25	score is 20	score is 15	score is 5	score is 0

Relationship wording is

full verb or preposition phrase	strong verb or preposition	any valid attempt	wholly incorrect or missing
score is 5	score is 3	score is 1	score is 0

Figure 5-4. Concept Diagram Scoring Algorithm.

thoughtful, more correct, or more exact attempt, especially one which includes a verb or a prepositional phrase, three points are given. The maximum of five points is scored for fully developed verb phrases in "correct" relationships or for fully developed relationships between nodes judged earlier to be logical, but not explicitly expressed in the reading.

A summary of numerical results from the concept diagram portion of the evaluation is included in the "Findings" section.

5.3.3 Posttest Factual Recall Quiz

The third measure of the tutorial design scheme's effectiveness was purely objective in nature and intent. It was a ten-question quiz (see Figure 5-5), presented on the same day as the posttest concept diagram portion of the evaluation. Its purpose was to provide data to indicate whether or not the relationship-based text structure facilitated recall of specific information better than did the strict linear presentation.

The results of this testing portion were not considered critical to the overall evaluation, but were included simply to gain some purely objective results. This is primarily so, since recall of specific facts in the reading material does not necessarily reflect a student's process of structuring concepts in a hierarchy of abstraction. Indeed,

IN-CLASS ASSIGNMENT -- PART II

The following questions were taken from the reading you were assigned in the laboratory last week. Answer the questions based on the information presented in the reading.

---

1. Who headed the corporate team which developed Ada?

2. What caused the DoD to initiate the process which produced Ada?

3. Non-DoD interest came from what public/international sectors?

a) c)
b) d)

4. Succinctly, but specifically, what role did each of these documents play in the design/development/implementation of Ada?

TINMAN :
STONEMAN :
STEELMAN :
PEBBLEMAN:

5. What publication(s) has Ada caused to be created? Under whose auspices?

Publication(s):
Auspices :

6. Name 3 things that are unique about the Ada design and development process.

a)
b)
c)

7. Circle the languages which served as baselines for Ada.

TACPOL	JOVIAL	ALGOL
SNOBOL	PL/I	CMS-2
Pascal	FORTRAN	COBOL

8. In what areas did the DoD assert control on the implementation of Ada?

a) c)
b) d)

9. What color was given Intermetrics' proposed Ada design?

10. What top-level issues make the Ada language distinct from other programming languages?

a) c)
b) d)

Figure 5-5. Objective Portion of the Posttest

a student's use of some mnemonic device may similarly account for his ability to recall details. Additionally, there is evidence that a deficiency in the administration of the experimental treatment introduced a negative bias in the experimental group's ability to recall specific information from the reading. Further discussion of this issue is deferred until a more appropriate section.

As for the test format, there were 5-1/2 relationship-oriented questions. The remaining 4-1/2 questions required recall of specific details embedded in both readings.

5.3.4 Posttest Attitudinal Surveys

The final tools used to evaluate the tutorial design methodology were purely subjective attitudinal surveys. One survey took the form of a written questionnaire and the other, a group interview session. The primary purpose of the survey portion of the evaluation was to collect subjective data concerning the use of the non-linear presentation of information in the tutorial.

The written survey (Figure 5-6) was presented to the experimental group at the end of the posttesting phase. The group interview session took place approximately one week later. The interview was conducted by the researcher in the instructor's absence. It was in this forum that the students were informed that they had taken part in an experiment not of the instructor's making.

Attitudinal Survey

This questionnaire accompanies the hypertext-based tutorial you used in your programming laboratory. Please thoughtfully but briefly provide answers to the questions below: (use the back or extra paper if needed).

1. How was the computer-based presentation of information better or worse than reading a journal article covering the same information? Be specific, please.

2. What percentage of the hypertext nodes did you explore as you read the information in the tutorial?

0-25% 26-50% 51-75% 76-99% 100%

3. With respect to the number of nodes, there were... (circle one of a-c, plus d if needed)

a. too few/too sketchy in content/detail
b. about enough to present sufficient detail
c. too many/too much detail irrelevant to the topic
d. other comment (please elaborate)

4. When you used F4 to activate a link to additional text, did the text which ensued contain the content and relevant detail of information you expected/needed? Elaborate.

5. Briefly comment on the tutorial with respect to:

Readability:

Comprehensiveness:

Understandability:

6. What SINGLE change would you recommend for the tutorial which would help you, the reader, understand the relationships between and relative detail associated with the concepts presented in the tutorial?

Figure 5-6. Attitudinal Survey for Experimental Group.

Information compiled from the questionnaires and interview will be used to propose recommendations for use in future tutorials designed using the methodology presented in this thesis (see Chapter 6). Transcripts of the written surveys are provided in Appendix C, and a summary of both the written and oral feedback is presented in the "Findings" section of this chapter.

5.4 Findings

As mentioned earlier, the instructors' feedback on the content equivalency questionnaire was incorporated into the final version of the tutorial presented to the students in the experimental group. Individual comments will not be included here.

The tallied results of each of the three evaluation methods which were linked to the experiment are presented in Tables 5-1a and 5-1b. In general, the combination subjective/objective and purely subjective evaluations indicate that the tutorial's relationship-oriented approach was well-received. Further, it had a measurably positive effect on the experimental group readers' process of mentally structuring the concepts presented in it. Numerically speaking, however, the purely objective measure did not indicate that the tutorial facilitated verbatim retention of specific information, any more than did a linear journal article.

Table 5-1a. Control Group Scores

Student	Scores			
	Pretest	Posttest	Gain	Quiz
subject 1	88	228	140	72
subject 2	34	170	136	30
subject 3	314	369	55	80
subject 5	250	270	20	132
subject 7	324	509	185	132
subject 8	255	344	89	100
subject 9	374	431	57	52
subject 10	82	240	158	76
subject 14	166	226	60	117
subject 20	170	386	216	50
Avg	205.7	317.3	111.6	84.1
Var	11992.8	10400.2	3759.0	1127.3
Std	109.5	102.0	61.3	33.6

Table 5-1b. Experimental Group Scores

Student	Scores			
	Pretest	Posttest	Gain	Quiz
subject 4	186	445	259	62
subject 6	310	454	144	89
subject 11	158	318	160	79
subject 12	224	373	149	49
subject 13	189	523	334	137
subject 15	252	325	73	65
subject 16	206	399	193	61
subject 17 *	341			
subject 18	256	351	95	91
subject 19	71	238	167	79
Avg	219.3	380.7	174.9	79.1
Var	5343.0	6492.2	5710.1	590.8
Std	73.1	80.6	75.6	24.3

* Missed posttests (not figured in those statistics)

5.4.1 Results of Concept Diagram Testing

The use of concept diagramming as the primary evaluation tool offered a plausible way to determine if the tutorial design methodology actually influenced the structure students used in assimilating the concepts discussed in reading material. Analysis of the average gain scores for each group evidences that the experimental treatment positively affected the concept structuring technique of the experimental group members. Statistically, with alpha = .05 the experimental group began to view the concepts in the tutorial reading in a hierarchy of abstraction, more so than did the control group.

5.4.2 Results of Objective Post-treatment Quiz

As referenced earlier, a weakness in the instructions accompanying the administration of the experimental treatment essentially voided any objective measure of the treatment's effect on recall of facts and details. Some comments expressed the post-treatment attitudinal surveys, and reconfirmed in the group interview, reveal that the experimental group students did not fully understand the course-related, information transfer purpose of their laboratory assignment. The students commented that they understood their purpose was to evaluate the hypertext-based method of presentation, and <u>not</u> to read for understanding

the content of the tutorial. This sentiment was overwhelmingly expressed in the group interview session.

The effects of this misinterpretation are reflected in the experimental group's individual and average scores on the objective portion of the posttest. The control group's average score surpassed the experimental group's by 5 points (see Tables 5-1a,b). Statistically, a 5-point spread in the average scores (out of 170 possible points) is only significant with alpha <= .35, considering the variances of the raw scores (see Tables 5-1a,b). However, the wording of the answers themselves, and written remarks such as "don't remember," in conjunction with the comments noted in the attitudinal surveys, are indicators that the experimental group, by and large, was not able to recall specific information, based on the reading, with the same accuracy as the control group. Also, it is noteworthy that even the control group averaged just under 50 percent of the maximum test score.

In the final analysis, we must consider both the raw test scores and the evidence of a deficiency in the administration of the experimental treatment. In light of these factors, there is not sufficient evidence to show that the scores of the objective test were positively influenced by the experimental treatment.

5.4.3 Results of the Attitudinal Surveys

The students' written responses to the survey questionnaires are presented in Appendix C. A general summary of the comments is included here. On the positive side, the students approved of the enhanced topic organization and key topic highlighting which the relationship-driven approach in the tutorial provided. Also, the "select only the nodes you want" opportunity provided by a hypertext structure was noted as beneficial. On the negative side, some students were frustrated by the distraction of the reverse video image of the hypertext nodes. Additionally, some students would have preferred the written medium over the video display terminal, so quick scanning of the entire document could be possible.

Overall, many of the negative comments were reflections of the students' frustrations caused by the Knowledgepro environment itself. The reverse video, color background, absence of a browsing map, etc. are all restrictions of the environment. Also, since several students commented that the hypertext experience was "not the usual way of reading," there is more evidence that factors other than the hierarchical approach to text structuring were the key sources of user displeasure.

In the interview session, all ten of the experimental group students admitted that they approved of the relation-

driven text structuring approach. Also, with more familiarity with the hypertext way of reading, they would have enjoyed the experience more, and gained more benefit from the tutorial. This can be attributed to the researcher's failure to ensure that the reading content alone was to be evaluated, and not the hypertext experience itself. Overall, and of most significance to this thesis, the attitudes concerning the portions of the tutorial which can be directly linked to the decomposition methodology were overwhelmingly positive.

6. Conclusions and Recommendations

This concluding chapter draws together the research described in the previous chapters. It focuses on how the research represents a well-rounded effort to improve the learning effectiveness of a microcomputer-based computer-assisted tutorial. Firstly, the chapter reviews and summarizes the results of the individual research products. Secondly, it enumerates the conclusions which may be drawn from the research. Finally, it lists recommendations both for improvements and for follow-on efforts to this research.

6.1 Summary

The broadbased literature survey revealed that in both military and civilian arenas, research in the last few decades has contributed to improved computer-based instructional products. Modern efforts have brought about products which employ high-tech features such as artificial intelligence, enhanced high-resolution graphics, and interactive video discs. In addition there has been a recent emphasis on hypertext- and hypermedia-based products which facilitate linking of text, graphics, and audiovisual resources.

Cognitive theory is concerned with the ability of an instructional method to facilitate assimilation of the

concepts presented by that method. To that end, this research was undertaken with learning theory, particularly Ausubel's theory, as a guide. This thesis has proposed a methodical approach to structuring the information content of a hypertext-based CAI tutorial to enhance a reader's assimilation of concepts. The methodology also incorporates knowledge engineering, software engineering and hypertext.

The methodology was demonstrated by transforming the contents of a professional journal article into a hierarchically structured hypertext document. The resulting tutorial product was evaluated by way of a single-case experiment. Subjective measures of the experimental results indicated that the tutorial had a positive effect on enhancing its reader's assimilation process. Objective measures were inconclusive.

6.2 Conclusions

Upon the completion and analysis of this multiphase effort, several conclusions can be drawn.

1. A methodology has been developed which facilitates the mapping from a printed source or expert knowledge base to the textbase of a computer-assisted tutorial. The methodology is solidly founded on the Object-Oriented Design (OOD) paradigm, a proven problem decomposition and software design methodology. In the same way OOD maps from a problem

description to a structured network of computer software components, the methodology offered in this work facilitates a decomposition of an expert knowledge base into a structured network of text components. Particularly, this network of passages represents a hierarchy of abstraction, which learning theorists advocate as an effective presentation of instructional material.

2. The methodology provides a means for designing formal, relationship-driven structure into a hypertext document. In doing so, it emphasizes the establishment of nodes and inter-text links, solely on the basis of relationships between concepts in the text. Pre-establishing text links in this manner provides three major benefits to the reader:

a. Expressing node names as phrases which reveal a relationship between the current topic and a related topic explicitly draws attention to and reinforces the relationship between the two topics;

b. With "up front" information about the text to which a node is linked, the reader is offered additional motivation for exploring the link, i.e. the reader knows beforehand the general theme of the linked text, and will not be surprised by seemingly unrelated information; and

c. Since the nodes are hierarchically structured and cannot be altered by the user, the potential for the reader to get lost in a web of text links, as is possible in unstructured hypertext, is removed.

3. The experimental group members retained the hierarchical, relationship-based structure of the tutorial's key concepts better than the members of the control group. The significance is this: the students who used the structured tutorial began to view the concepts in the reading material as being related through a hierarchy of abstraction. This result can be attributed to the Ausubelian approach to structuring the information presented in the tutorial.

4. Objective measures requiring the recall of specific facts or details were inconclusive. This result is attributed to a weakness in the instructions provided to the members of the experimental group prior to undergoing the treatment. Comments in the students' critiques of the tutorial experience indicate that the students did not understand that their role in using the tutorial was to absorb and understand its information content. Instead, the students concentrated on the novelty of the hypertext presentation itself. Consequently, they were not successful in retaining the tutorial's content, as measured by an objective post-treatment test.

5. An important observation with implications for future research was made as a result of implementing Object-with-Relations (O-R)-type objects in hypertext. To enforce the visibility constraints in software, the hypertext environment must be sophisticated enough to provide the nesting of one text node inside another. The KnowledgePro environment used in this thesis provides this nesting capability, but TextPro, a hypertext-generating sister product to KnowledgePro, does not. The Relationships-to-Object (R-O) encapsulation model can at least be simulated using TextPro, though again, there is no means within the software to enforce visibility restrictions.

6. A few other conclusions can be drawn concerning the KnowledgePro expert system shell. During the administration of the control and experimental treatments, both of which made use of KnowledgePro capabilities, key limitations of the environment were pointed out by the students:

a. The system-supplied instructions for operating within the shell are not complete enough for first-time computer users. When the initial screen is displayed indicating the available compiled textbases, there are no on-screen instructions for how to begin exploring a base. Instead, the function-key menu at the bottom of the screen indicates the existence of a HELP function, which explains how to select a textbase and begin.

There is, however, no direct on-screen reference to the help facility.

b. The system is not robust; hitting a single incorrect key can cause a user to involuntarily terminate a session. The "legal" keys which can be used to browse an open textbase are the familiar <PgUp> and <PgDn> screen scrolling keys, and the less familiar F3, F4, and <space>. The <return> key, (often used as a reflex instead of F4 by new users attempting to activate a node), closes the file and returns the user to the initial display menu. According to post-treatment interviews with the students, this was a source of frustration. For that reason, it may have been a key cause for many students' lack of attention to the tutorial's information content. Additionally, the capability to scroll the screen one line at a time using the arrow keys was not provided by the environment. It should be noted in all fairness that the KnowledgePro product is not designed primarily as a hypertext product. It is an expert system shell, a subset of whose features facilitates the use of hypertext nodes and links.

7. Lastly, review of the evaluation process revealed a possible source of bias which could not be filtered from the experimental portion of the research. Admittedly, portions

of the tutorial's subject matter were already familiar to (albeit possibly misunderstood by) the students who took part in the experiment. The potential source of bias was revealed during the group interview session which followed all testing. Several students noted that what they read in the tutorial did not always agree with some information they had previously read or heard in classroom lectures. The result appears to have been both frustration with and in some cases hostility toward the tutorial. For this reason, the students lost some confidence in the tutorial's value as a learning tool. This factor, in addition to the previously addressed weakness in instructions, is seen as a negative bias, possibly manifest in the scores of the objective portion of the posttest.

6.3 Recommendations

As a result of the conclusions stated above and experience in conducting this research, the following recommendations are offered to those who would follow up on the work presented here.

1. In a future evaluation of the methodology described in Chapter 3, an attempt should be made to decompose a knowledge base directly from an expert, instead of from a printed article. The process may involve a preliminary knowledge acquisition phase which employs concept mapping or other techniques proposed by Novak and Gowin (1986).

Indeed, similar knowledge extraction techniques could be employed at the initial phase of the OOD process, which relies on an "a priori" understanding of the requirements and eventual components of the system being developed.

2. In the same exploratory sense as the previous recommendation, the decomposition methodology should be employed on a much larger knowledge base. However, based both on feedback from the students who used the tutorial and on the results of other research (e.g. Beeman, et al., 1987; Conklin, 1987a; Landow, 1987; Oren, 1987), a document that is too large and complex is a greater source of frustration to the reader than it is a viable medium of knowledge transfer.

3. In an attempt to remove the potential bias associated with presenting concepts already familiar to the subjects, the same experiment should be conducted using a tutorial which addresses concepts not familiar to the participants. Doing so may permit the researcher to gain deeper insight into the true measurable effects of the proposed approach to the presentation of instructional material.

4. The experiment should be replicated using a group who is both familiar with the concept of hypertext and adept at using the KnowledgePro environment as a browsing tool. In this way, the subjects' energy and time may be spent concentrating on the information content of the tutorial,

instead of on the particulars of the software environment in which the tutorial is hosted.

5. In view of the limitations or weaknesses of the KnowledgePro shell as used as a hypertext generator, both the research and the students may be better served by the use of a more sophisticated, more robust hypertext environment.

6.4 Remarks

This thesis has been an effort to promote and enforce the structure of hypertext documents as a means of facilitating the meaningful transfer of information from instructor to student. In so doing, this research has endeavored to further the cause of hypertext as a useful contribution to the field of computer-aided instruction.

Appendix A: Encapsulated Objects from Example Knowledge Base Decomposition

This appendix of figures presents the encapsulated objects which resluted from Step 7 of the decomposition presented in Chapter 4. There are two basic O-R object depictions used in this appendix. Figure A-1a depicts an object which encapsulates other objects. Figure A-1b represents an object which contains text only.

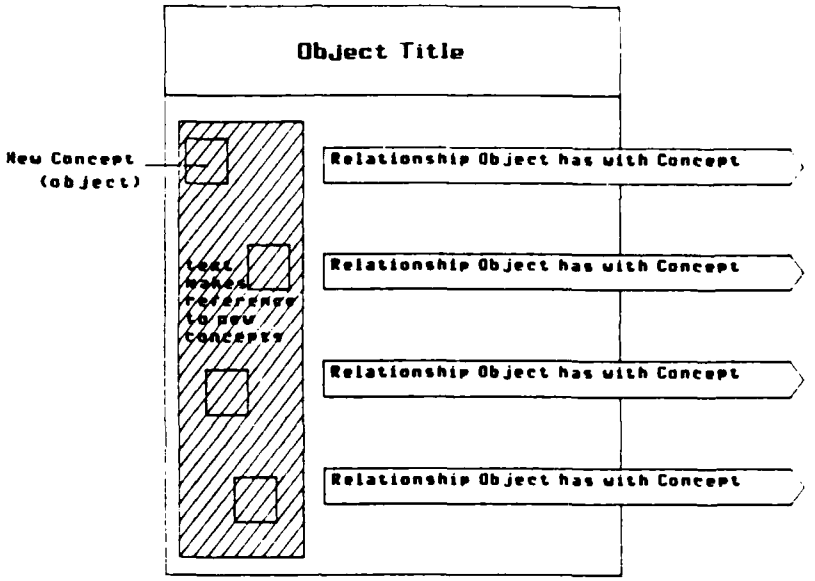

Figure A-1a. O-R Object Encapsulating Other Objects

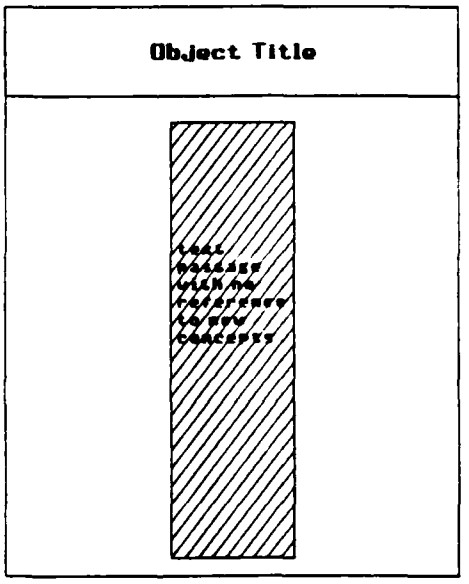

Figure A-1b. O-R Object Encapsulating Text Only

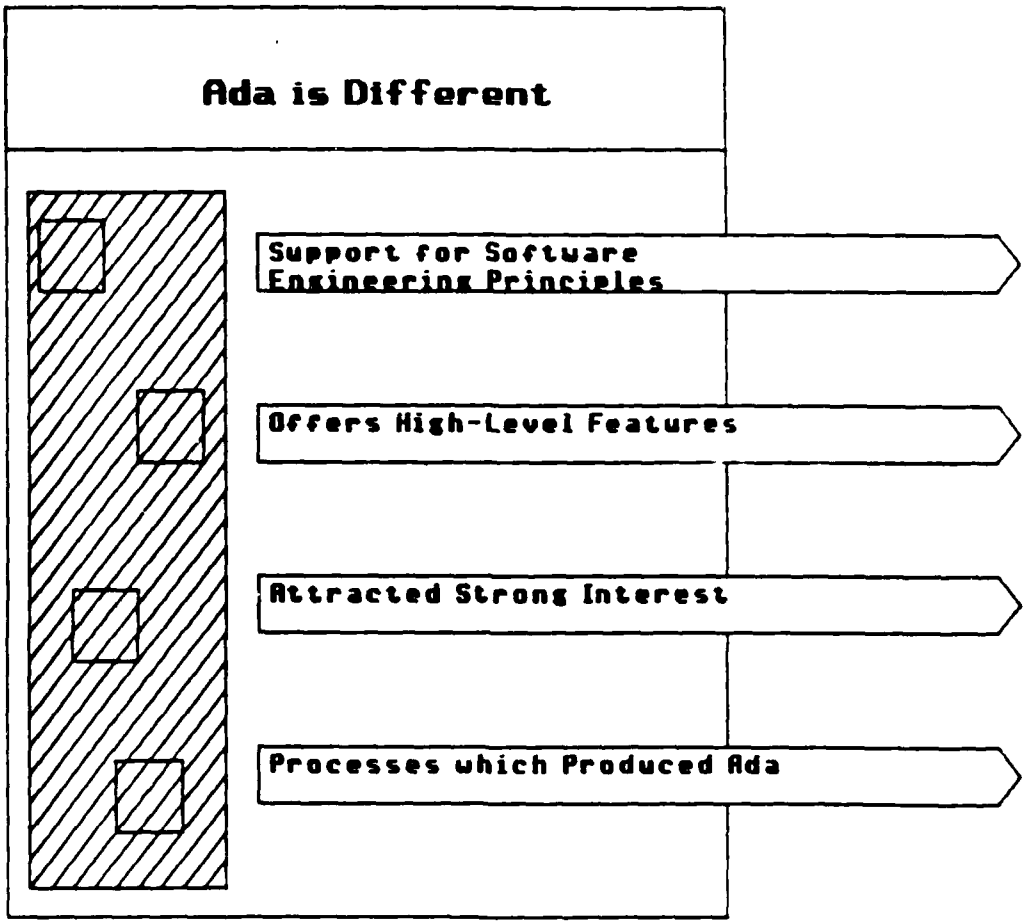

Figure A-2. Main Encapsulated Object

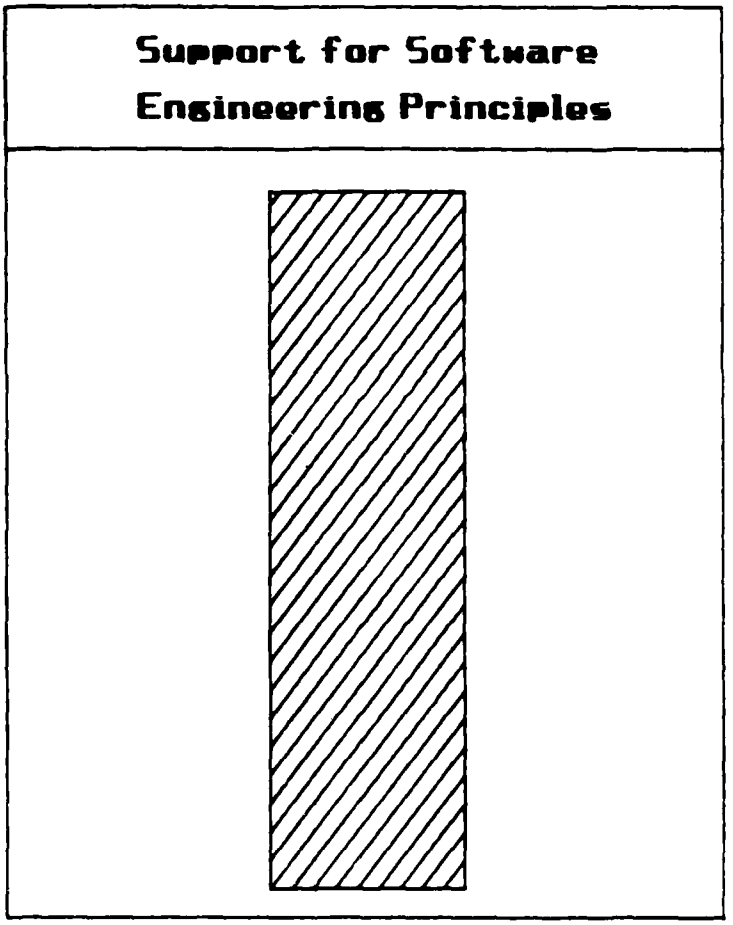

Figure A-3a. First-Level Object (1)

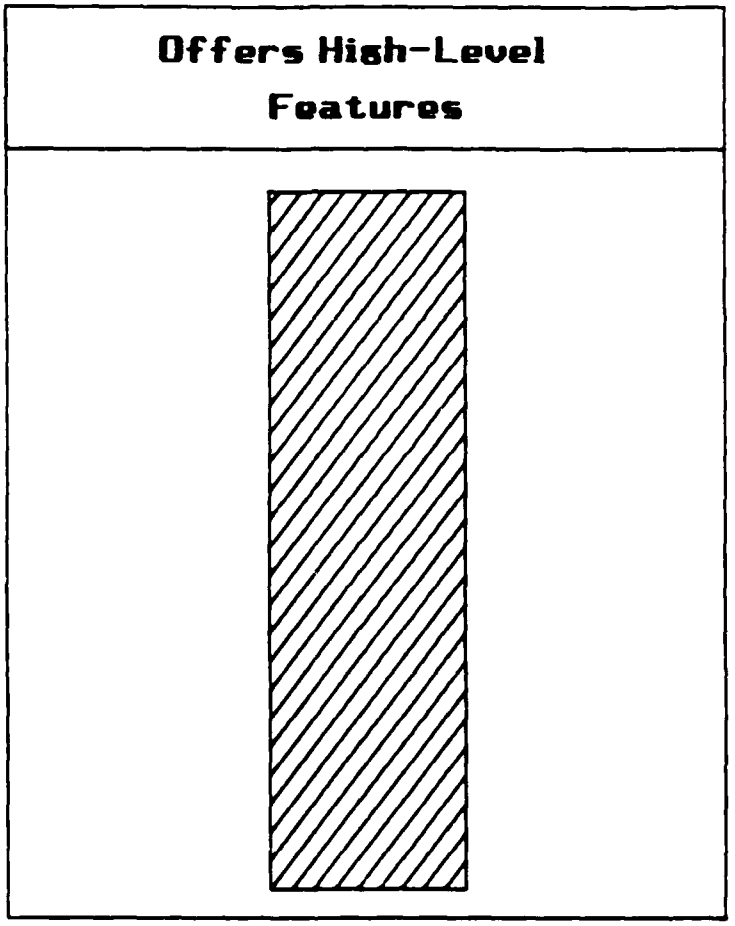

Figure A-3b. First-Level Object (2)

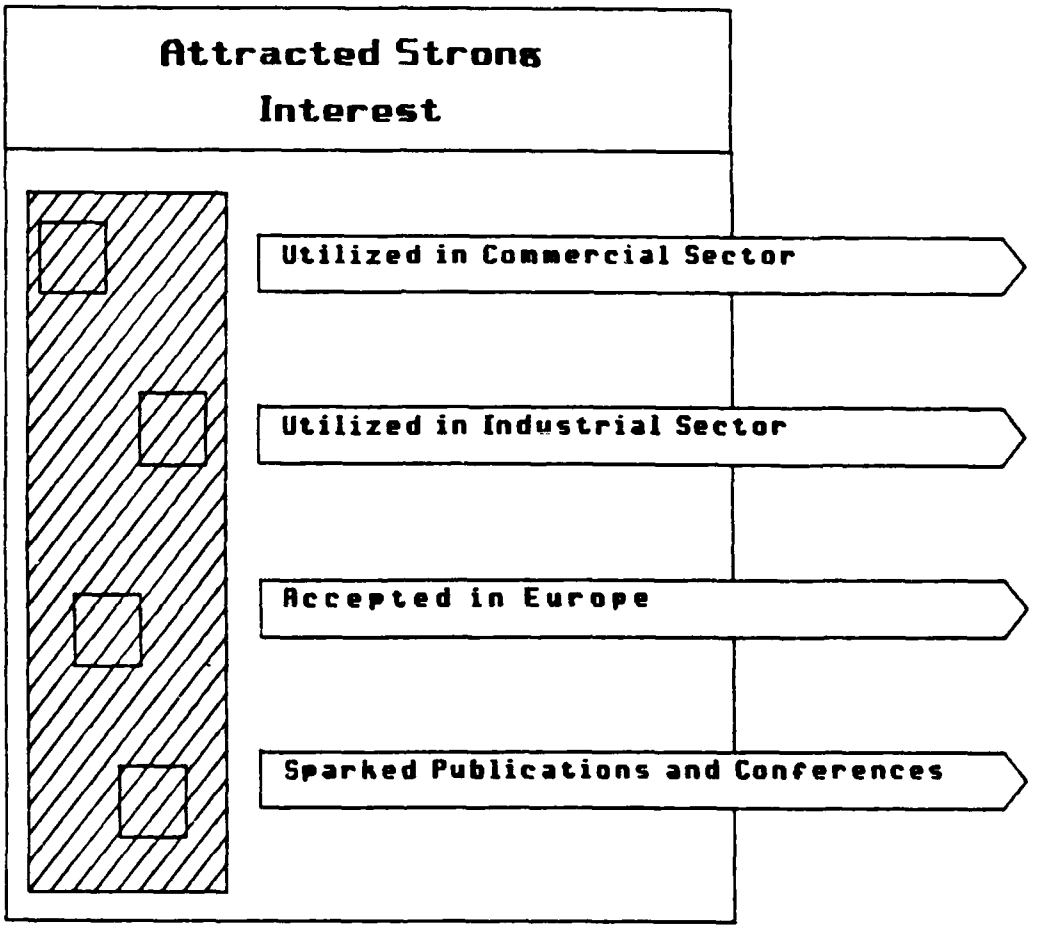

Figure A-3c. First-Level Object (3)

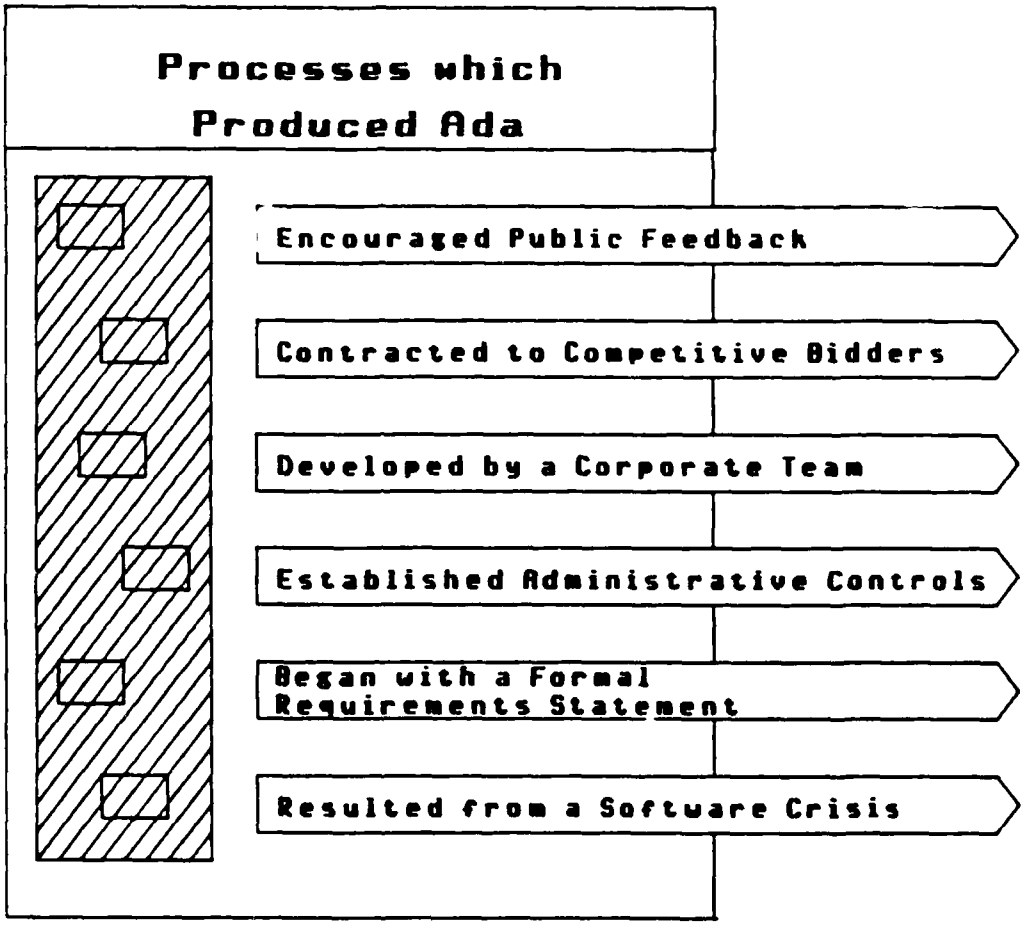

Figure A-3d. First-Level Object (4)

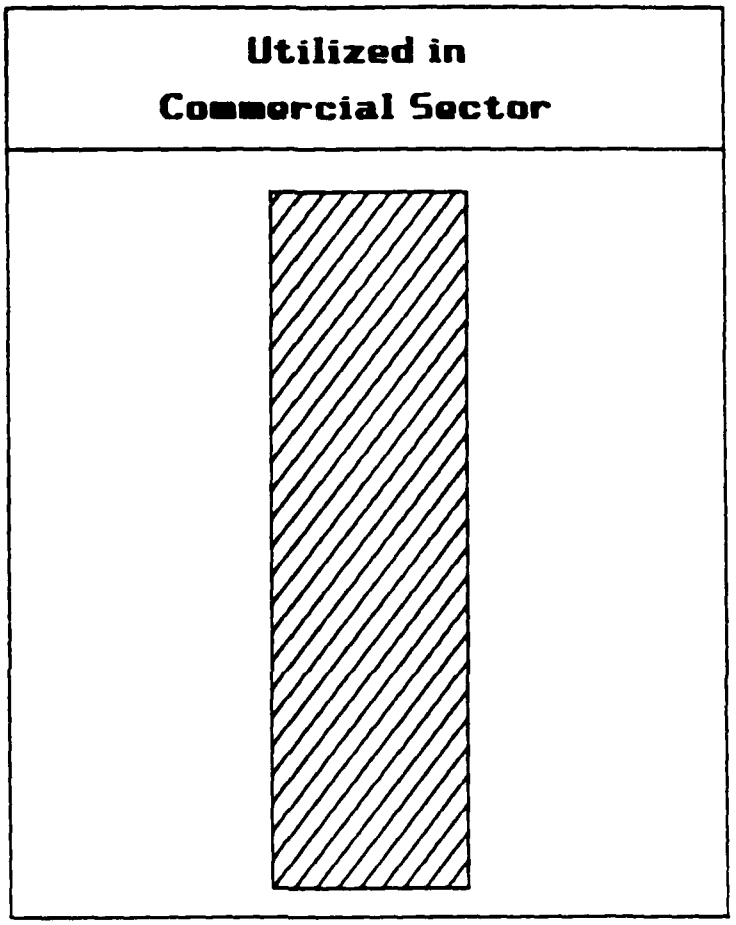

Figure A-4a. Second-Level Object (1)

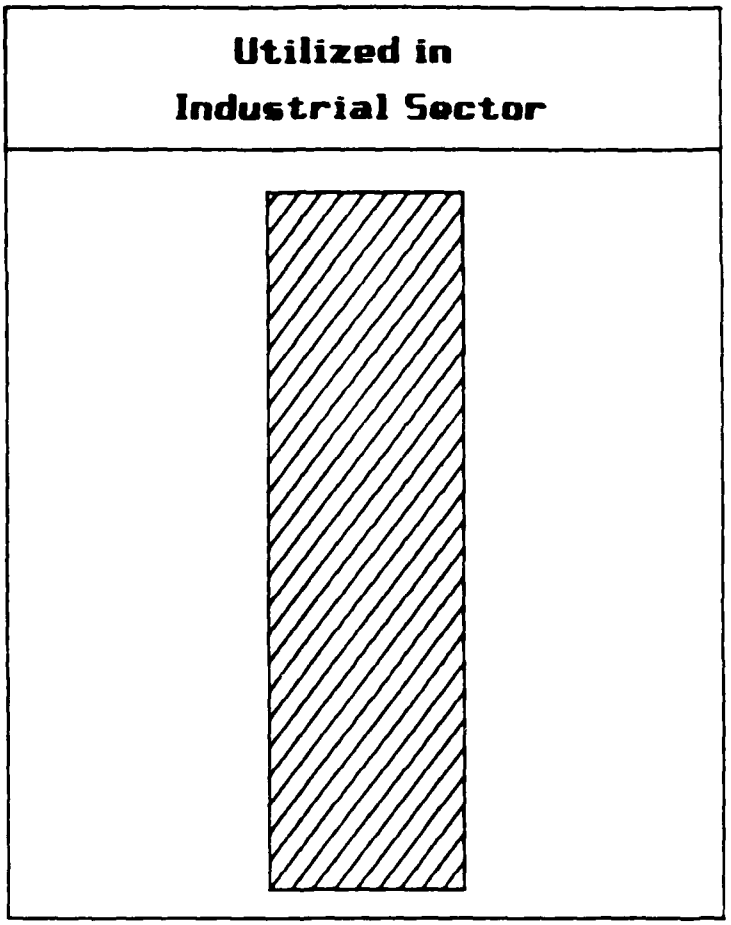

Figure A-4b. Second-Level Object (2)

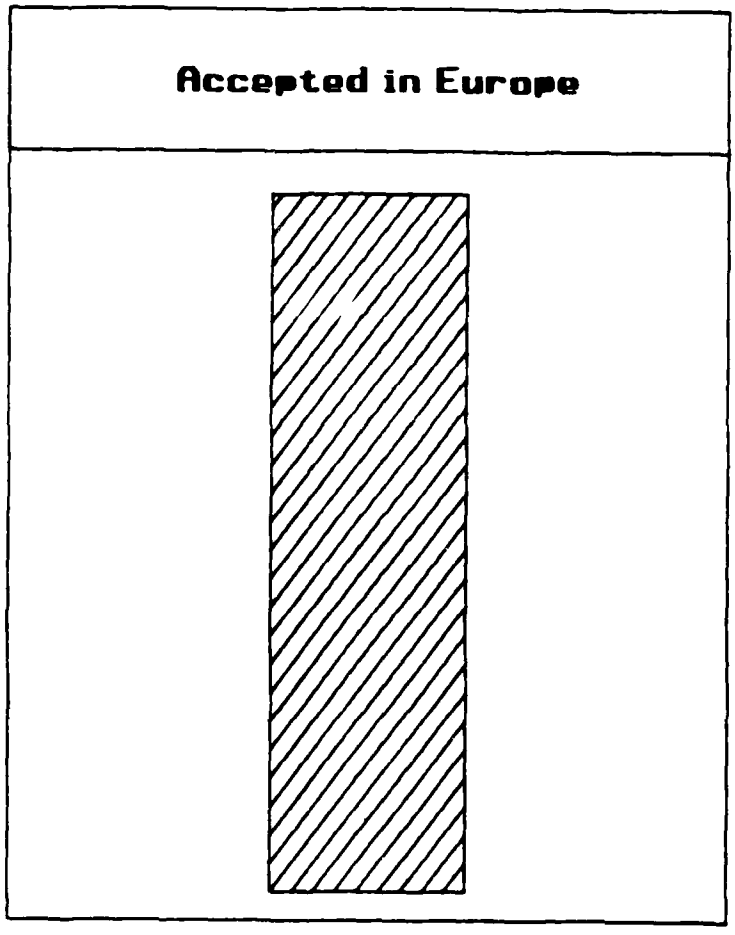

Figure A-4c. Second-Level Object (3)

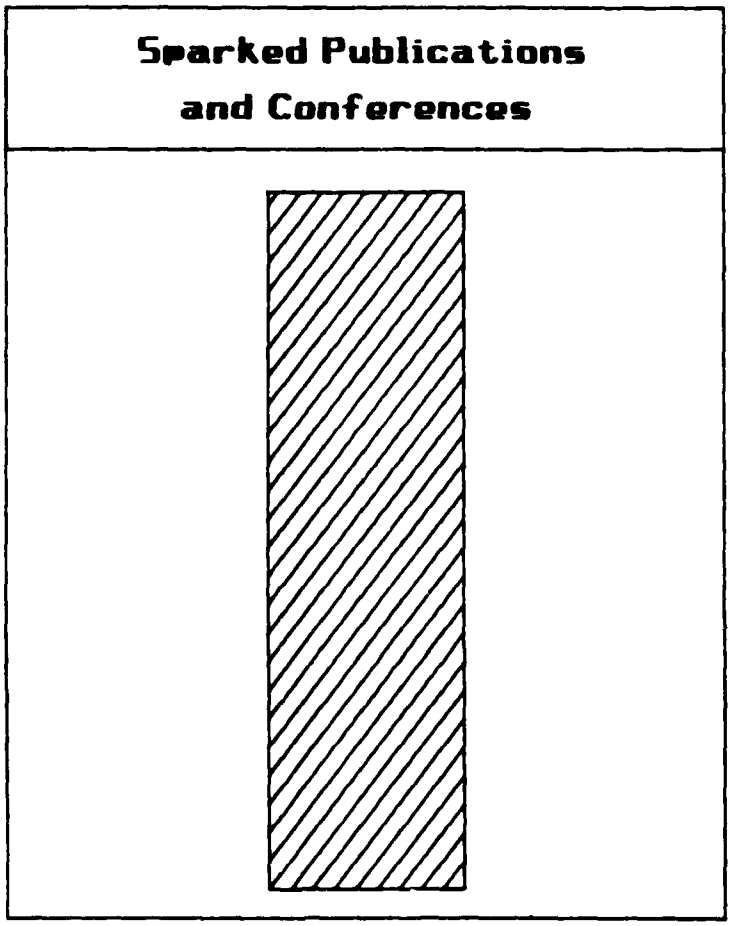

Figure A-4d. Second-Level Object (4)

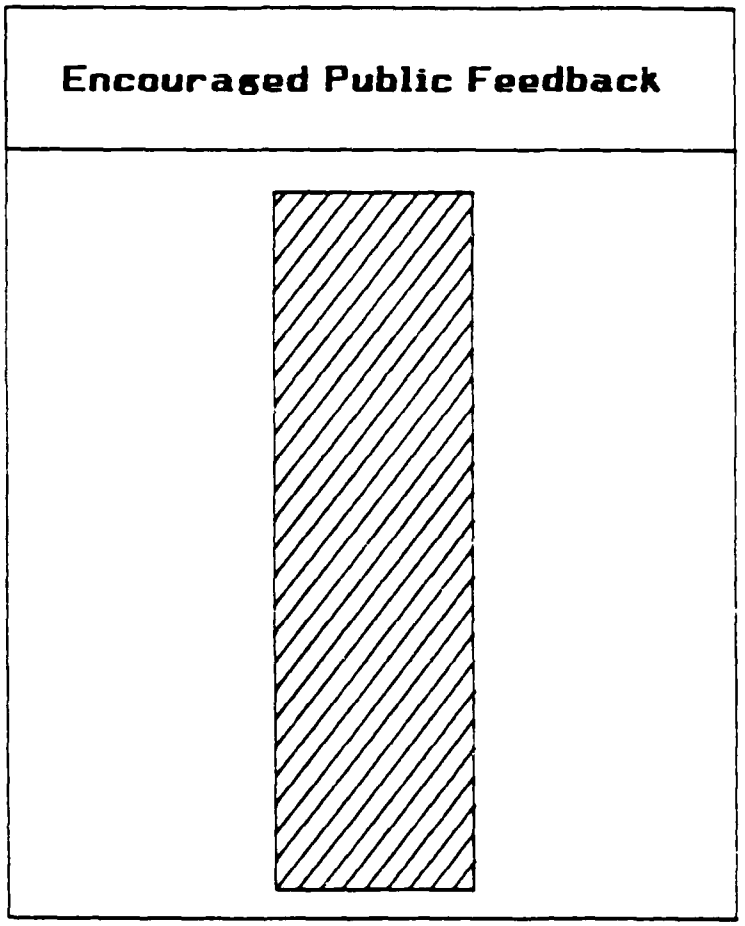

Figure A-4e. Second-Level Object (5)

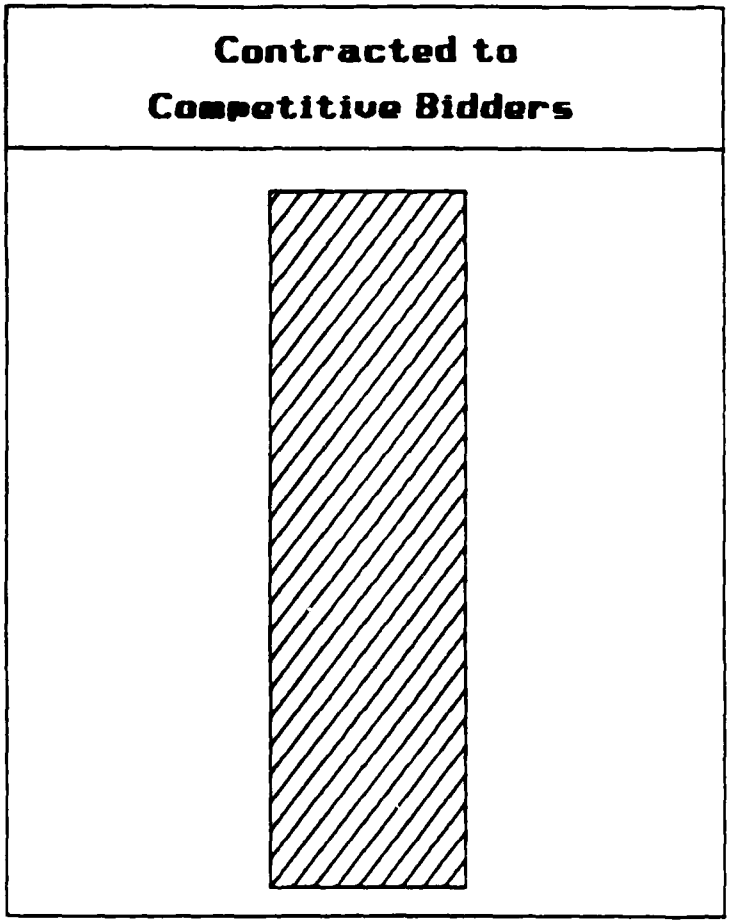

Figure A-4f. Second-Level Object (6)

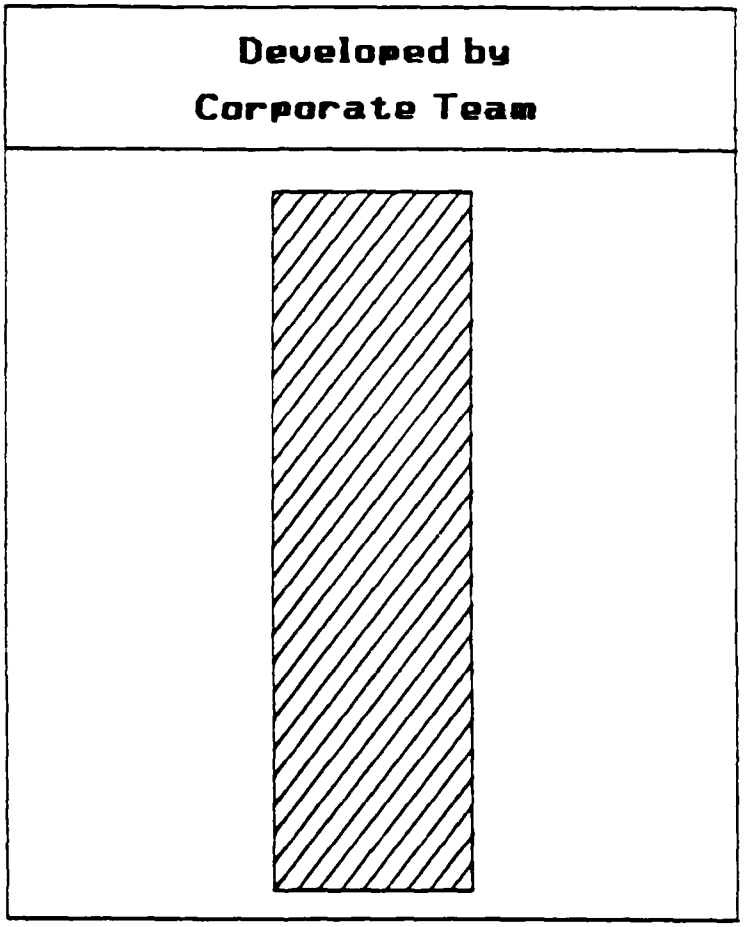

Figure A-4g. Second-Level Object (7)

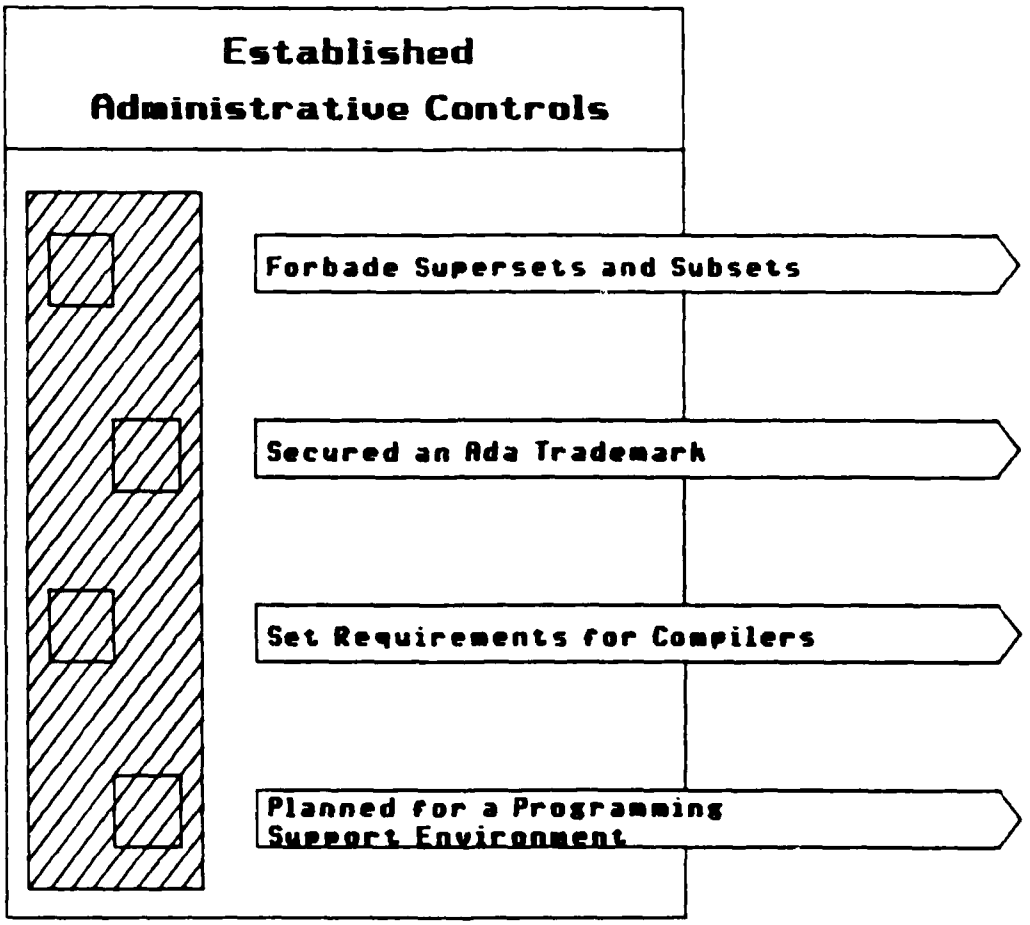

Figure A-4h. Second-Level Object (8)

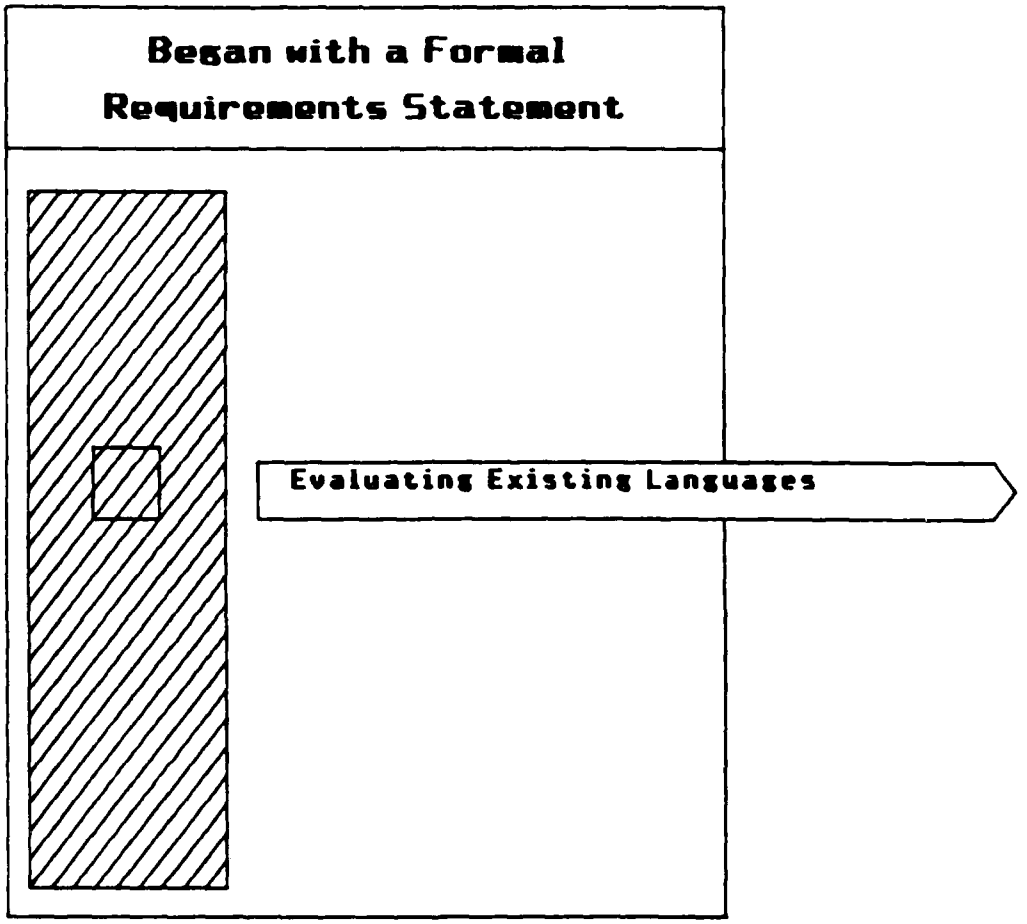

Figure A-4i. Second-Level Object (9)

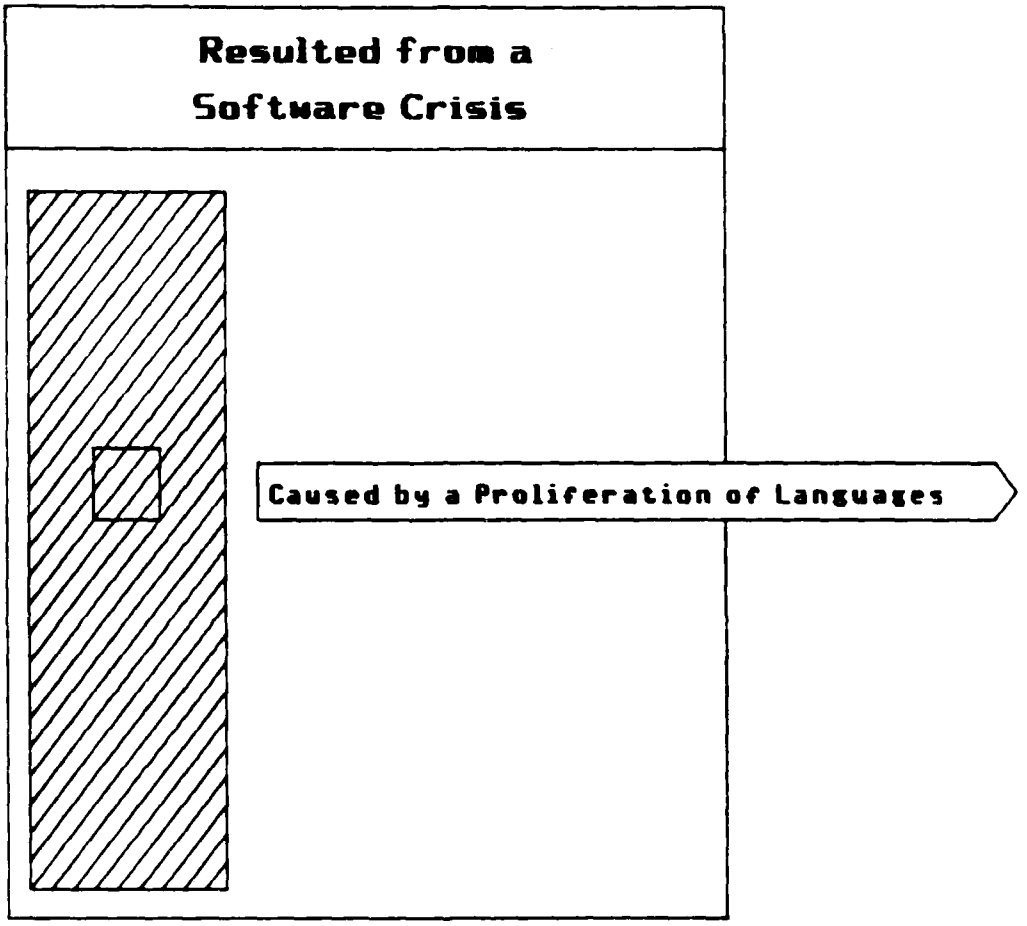

Figure A-4j. Second-Level Object (10)

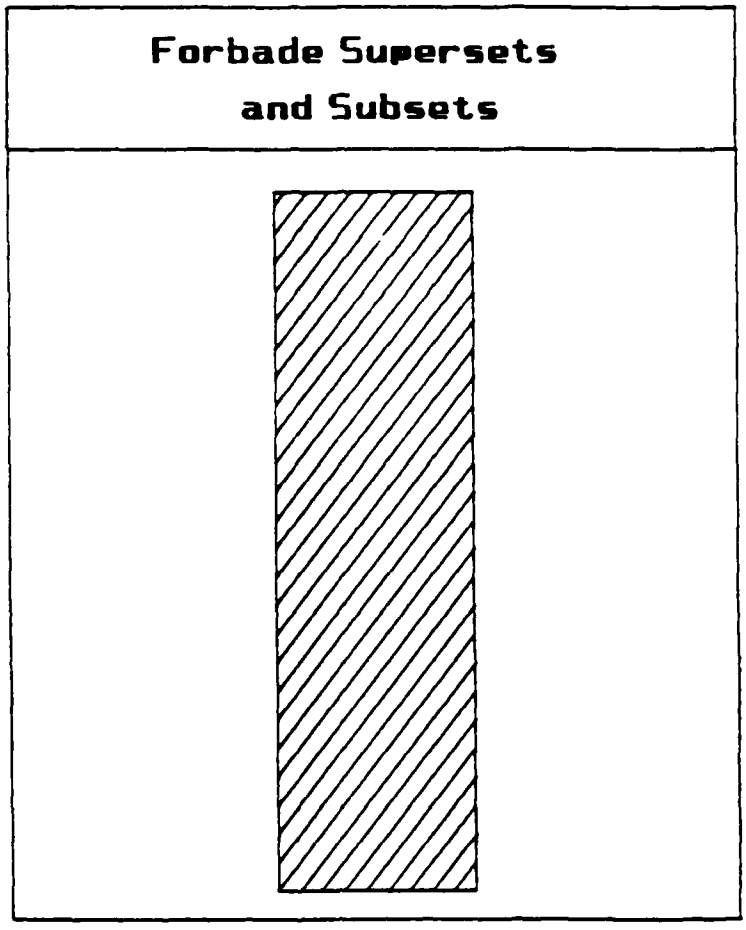

Figure A-5a. Third-Level Object (1)

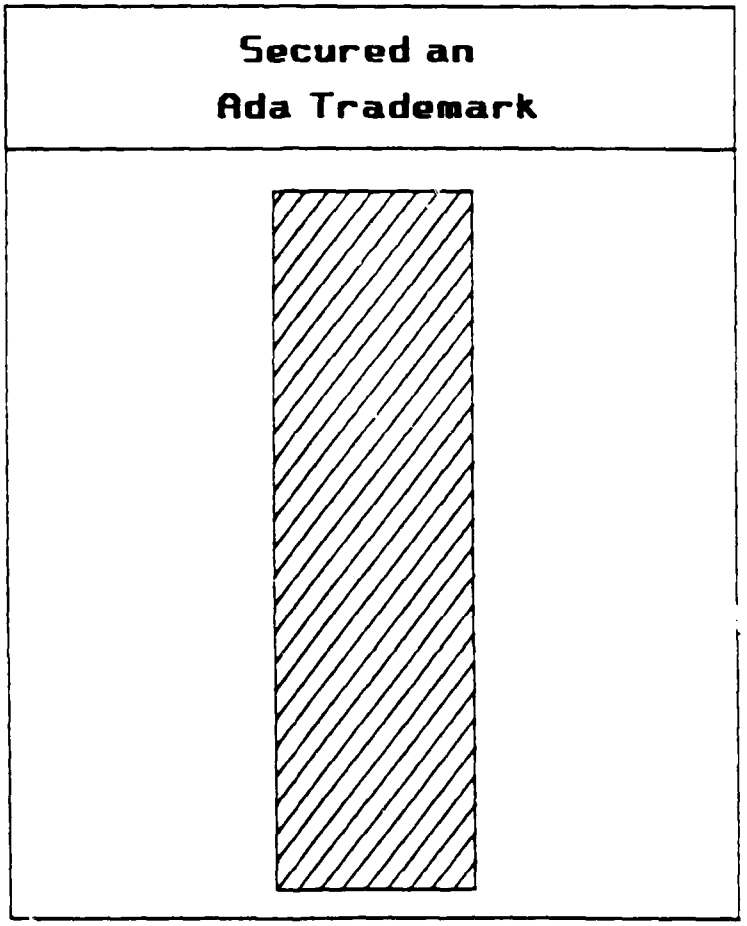

Figure A-5b. Third-Level Object (2)

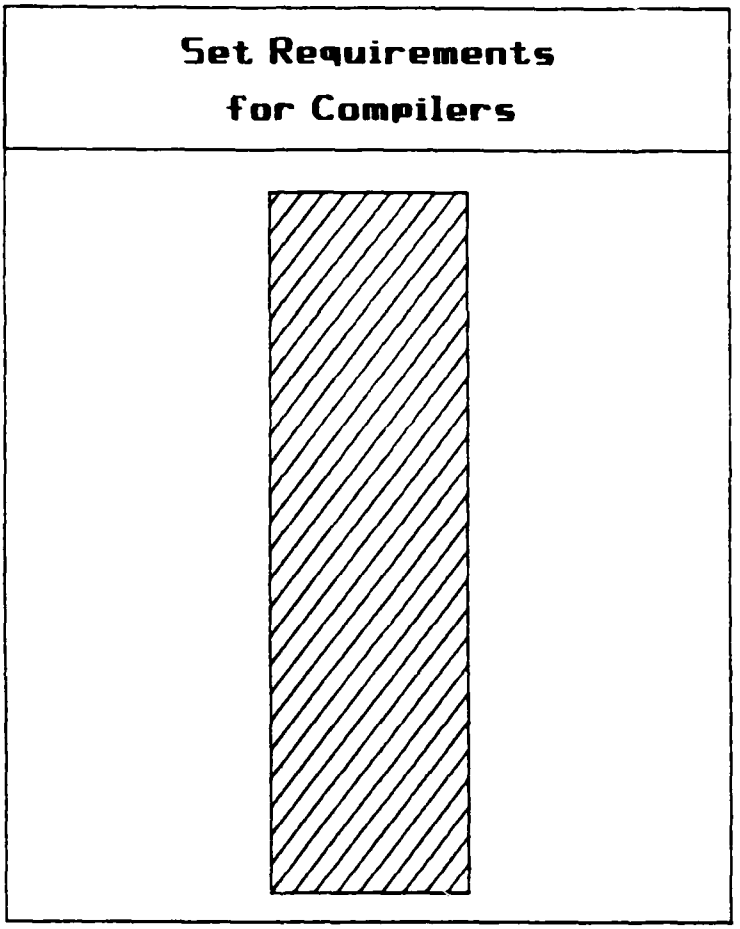

Figure A-5c. Third-Level Object (3)

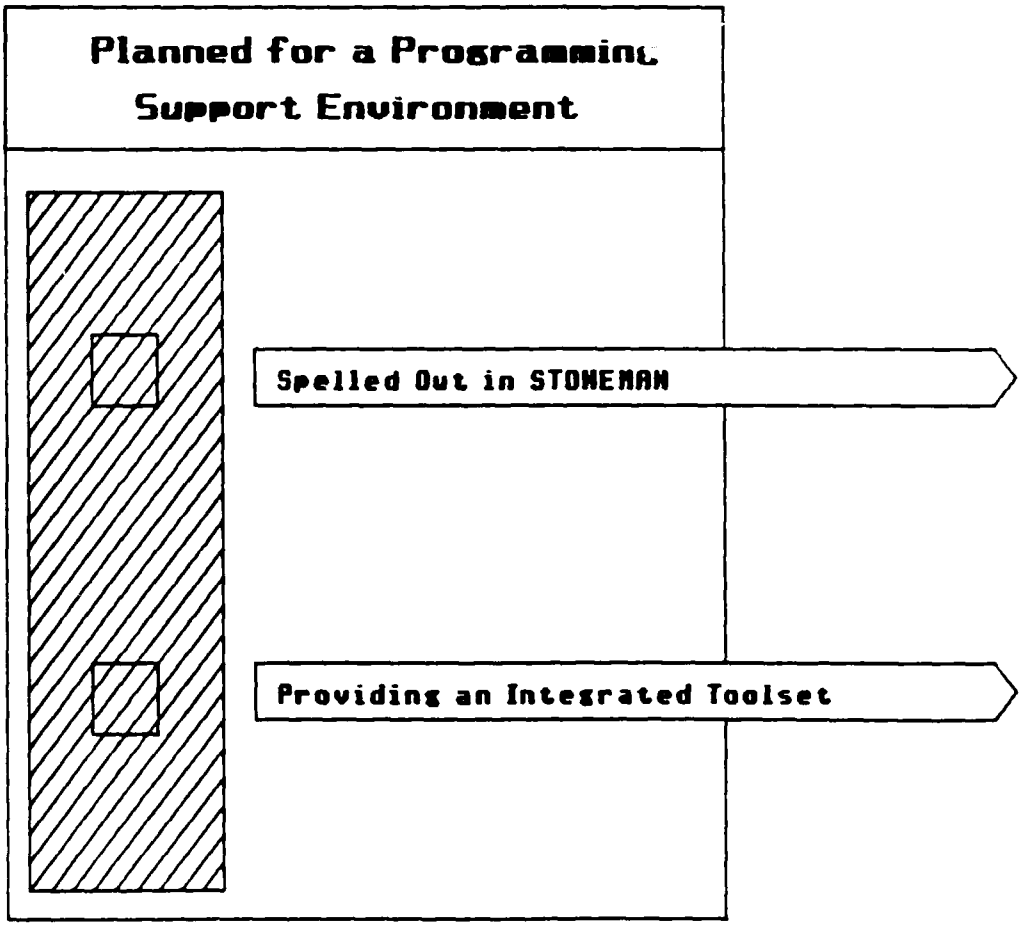

Figure A-5d. Third-Level Object (4)

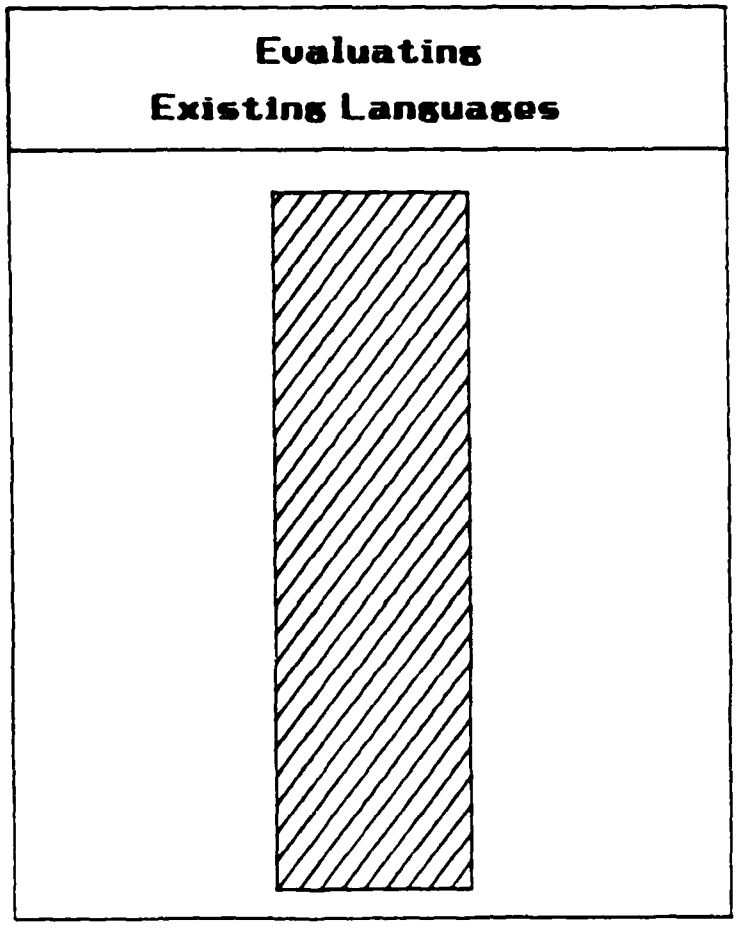

Figure A-5e. Third-Level Object (5)

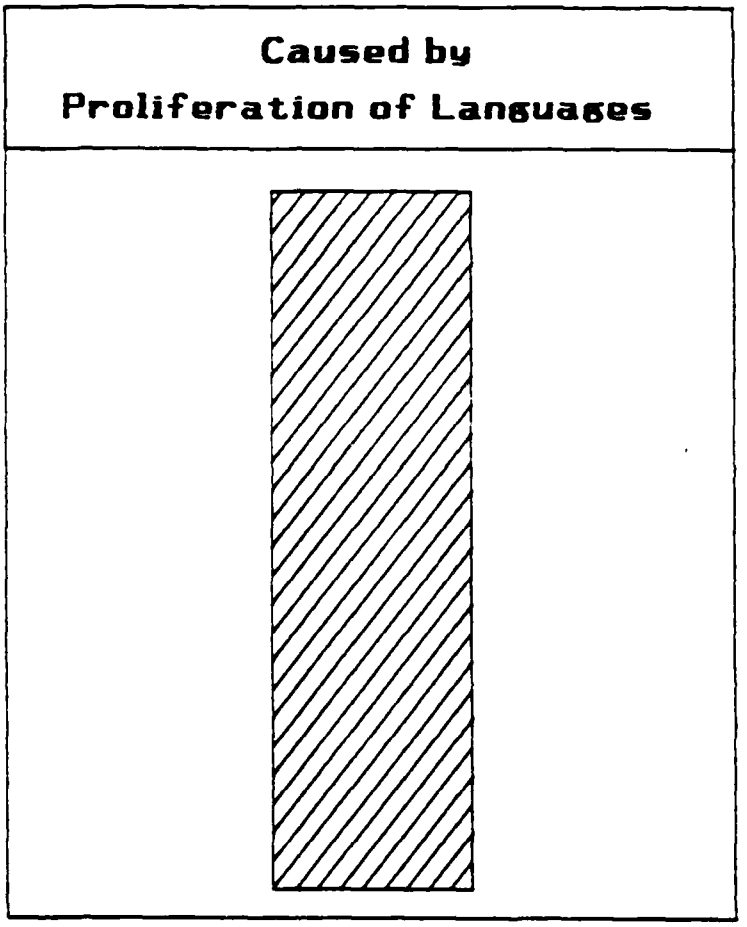

Figure A-5f. Third-Level Object (6)

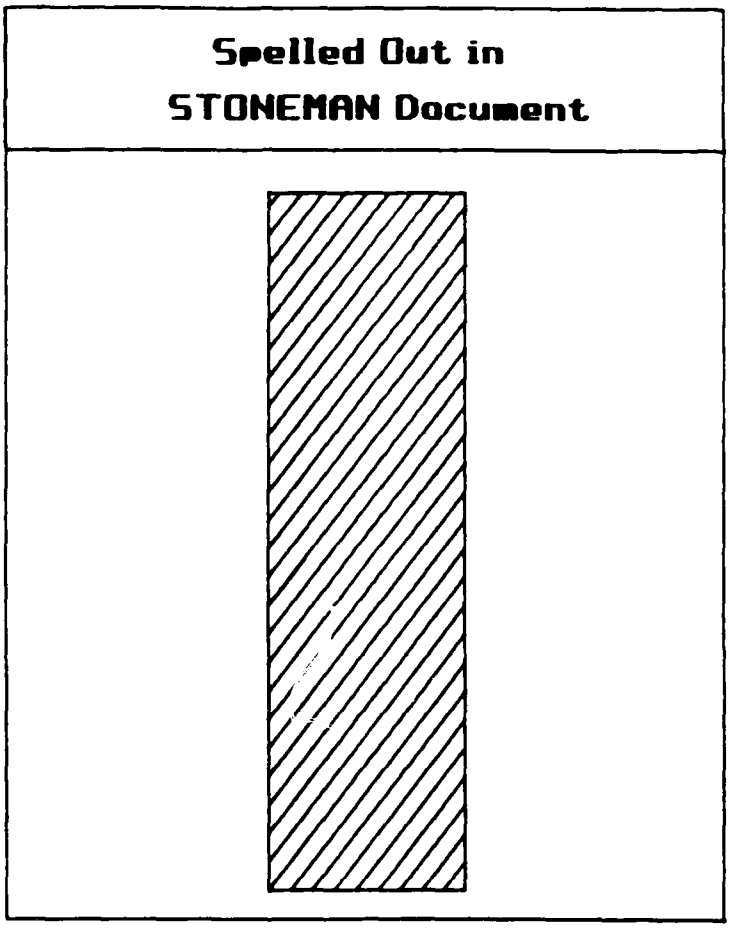

Figure A-6a. Fourth-Level Object (1)

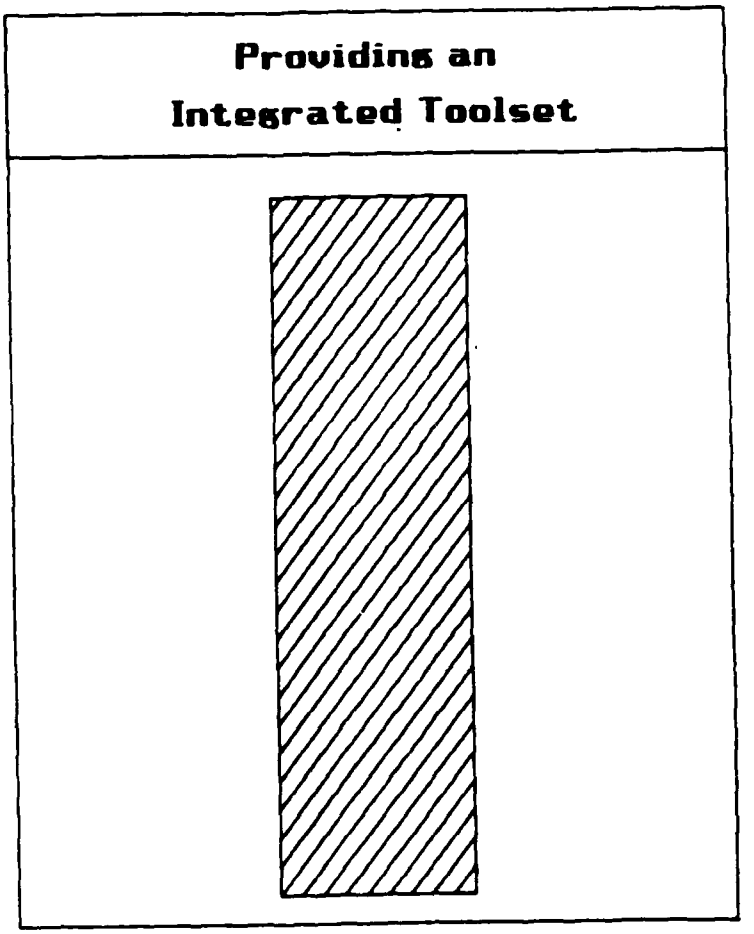

Figure A-6b. Fourth-Level Object (2)

Appendix B. Complete Hypertext-Based Tutorial

This is the hypertext-based tutorial created from the decomposition begun in Chapter 4. The remainder of the text in this appendix appears in the format of KnowledgePro commands and constructs (Knowledge Garden, 1988). The start and end of each text object is indicated at the left margin. Lower level objects are indicated by two or more asterisks, with more asterisks indicating a more deeply nested object.

```
(* The tutorial begins with a simple practice session, to *)
(* familiarize the user with the KnowledgePro hypertext   *)
(* functions.                                              *)

ask ('                Welcome to the MATH 555 tutorial!

*********************************************************
                Select INSTRUCTIONS for instructions
                              or
                Select QUIT to exit
*********************************************************'
,action,[INSTRUCTIONS,QUIT]).

if ?action is_not QUIT then
do (?action).

topic 'INSTRUCTIONS'.
window ('HOW TO USE THIS TUTORIAL',,,,1,1,78,20).
say ('
        This is a HYPERTEXT based computer-assisted
tutorial.

     Words appearing in a #mdifferent colors#m or
     #munderlined#m are hypertext nodes.  Activating nodes
     makes a window pop up, just like the one you''re
     reading.

        * Use the F3 key to move from node to node.
        * Use the F4 key to activate a node.
        * Use the <Space> bar to return from a new node.
        * Use <PgUp> and <PgDn> to scroll in a window.
              (look for "page 1 of ?" in lower menu)

              ********* TRY IT YOURSELF *********

         Use F3 to select the "different color " node.
             (a selected node appears in yellow on black)
         Then use F4 to activate it.  If you''re successful,
         you get a message saying so.

                    use <PgDn> to see more

         Now use F3 to select the "underlined" node.
         Use F4 to activate it and if you get a message,
         you''re ready to go.

               use <PgUp> to see original screen
').
```

```
   topic 'different color'.
   window ('Example hypertext node action',,,,5,5,73,15).
   say ('
                          YOU DID IT!
                Color only works on a color monitor.

                     Use <space> to return
').
   close_window ().
   end. (* different color *)

   topic 'underlined'.
   window ('Another practice hypertext node
action',,,,5,5,73,15).
   say ('
                          YOU DID IT!

          Nodes appear underlined on a monochrome monitor

                     Use <space> to return
').
   close_window ().
   end. (* underlined *)

close_window ().

window ('Are You Ready to Begin?',,,,5,3,71,12).
ask ('
       * Select ADA to begin the lesson
                      or
       * Select INSTRUCTIONS to return to instructions
                      or
       * Select QUIT to exit'
     ,desire,[ADA,INSTRUCTIONS,QUIT]).

if ?desire is_not QUIT then
do (?desire).

close_window ().
end. (* INSTRUCTIONS *)

(* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * )
(* The next section begins the actual tutorial textbase  * )
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * )
```

```
(* BEGIN ADA IS DIFFERENT *)

topic 'ADA'.
window ('ADA IS DIFFERENT FROM OTHER LANGUAGES'
        ,,,,1,1,78,20).
say ('
          Ada can be distinguished from the majority of
     other computer languages on many technical and
     nontechnical merits.  On non-technical grounds, the
     ordered design and development #mprocesses which
     produced Ada#m are unlike those of any other
     programming language.  Additionally, unlike other
     languages, Ada has #mattracted strong interest#m from
     users and enthusiasts in the international computing
     community, outside the Department of Defense (DoD).
     With regard to key unique attributes of its technical
     elements, Ada #moffers high-level programming
     features#m which contribute to Ada''s #msupport for
     software engineering principles#m.
').

(** BEGIN: PROCESSES WHICH PRODUCED ADA*)

   topic 'processes which produced Ada'.
   window ('Unique Design and Development
            Process',,,,3,3,76,19).
   say ('
     The Ada design and development process basically
     #mresulted from a software crisis#m in the DoD.
     Uniquely, the process #mbegan with a formal
     requirements statement#m, for a language to meet the
     DoD''s embedded system software needs.  Also unique was
     the fact that the initial language design was
     #mcontracted to competitive bidders#m, some from
     outside the U.S.  and #mdeveloped by a corporate
     team#m, as opposed to the typical committee or
     individual effort.

     The DoD also #mestablished administrative controls#m
     to govern implementation of the language product, both
     its syntax and its compilers.  And to a degree that had
     not been experienced before, through all of these
     processes, the DoD #mencouraged public feedback#m,
     actually implementing valid recommendations when
     possible.
').
```

```
(*** BEGIN: RESULTED FROM A SOFTWARE CRISIS *)

      topic 'resulted from a software crisis'.
      window ('Initiated Because of DoD Software
               Crisis',,,,4,4,74,18).
      say ('
     The impetus for the creation of Ada was a software
     crisis within the DoD.  The crisis was recognized in
     1974 when a DoD report was issued, estimating a
     software development and maintenance cost of over three
     billion dollars.  The crisis was #mcaused by a
     proliferation of languages#m within the department,
     which severely restricted reusability of both software
     and programmers.
').

(**** BEGIN: CAUSED BY A PROLIFERATION OF LANGUAGES *)

            topic 'caused by a proliferation of languages'.
            window ('Caused by Too Many Programming
                     Languages',,,,5,5,72,17).
            say ('
     The DoD recognized a proliferation of programming
     languages as a cause of its software crisis.  The DoD
     estimated there were literally hundreds of programming
     languages and dialects in use on DoD projects.  Each
     new language required its own contracted maintenance
     teams, which accounted for the astronomical cost.
').
            close_window ().
            end. (* caused by a proliferation of languages*)

(**** END : CAUSED BY A PROLIFERATION OF LANGUAGES *)

         close_window ().
         end. (* resulted from a software crisis *)

(*** END: RESULTED FROM A SOFTWARE CRISIS *)

(*** BEGIN: BEGAN WITH A FORMAL REQUIREMENTS STATEMENT*)

         topic 'began with a formal requirements statement'.
         window ('began with formal requirements
                  statement',,,,4,4,74,18).
         say ('
     The design for Ada was based on language requirement
     specifications originally established in the DoD''s
     STRAWMAN document.  The intent of the 1975 document was
     to detail specifications for a language to be used in
     DoD embedded computer systems.  After some revision,
     the document was re-released as WOODENMAN in early
     1976.  Later that year, the newer TINMAN revision was
```

```
        released.  TINMAN was used as a basis for #mevaluating
        existing languages#m for suitability as the new
        standard DoD language.  TINMAN later underwent revision
        to become the 1978 IRONMAN document.  The IRONMAN
        requirements were used at the time the design and
        development process formally began, but a still later
        revision, STEELMAN, was the final version which
        governed the  language development.
').

(**** BEGIN: EVALUATING EXISTING LANGUAGES *)

              topic 'evaluating existing languages'.
              window ('Evaluated Existing Languages Against
                       Requirements',,,,5,5,72,17).
              say ('
        The TINMAN document was used  as a basis for evaluating
        existing languages for suitability as the DoD''s single
        language for use in DoD software systems.  Eventually,
        ALGOL68, Pascal, and PL/I were selected as baseline
        languages, from which the new language need not be (and
        is not) compatible.  Those languages considered were:

        DoD/Embedded Computers
        CMS-2, CS-4, HAL/S, JOVIAL 3B, JOVIAL J73, SPL/1,TACPOL

        Process control and/or Embedded Computers (Europe)
        CORAL66, LIS, LTR, PEARL, PDL2, RTL/2

        Research-Oriented Languages
        ECL, EL-1, EUCLID, MORAL

        Widely Used and/or General Languages
        ALGOL60, ALGOL68, COBOL, FORTRAN, Pascal, PL/I, SIMULA67
').
              close_window ().
              end. (* evaluating existing languages *)

(**** END: EVALUATING EXISTING LANGUAGES *)

            close_window ().
            end. (*began with a formal requirements statement*)

(*** END: BEGAN WITH A FORMAL REQUIREMENTS STATEMENT *)
```

```
(*** BEGIN: CONTRACTED TO COMPETITIVE BIDDERS *)

        topic 'contracted to competitive bidders'.
        window ('Contracted to Bidders',,,,4,4,74,18).
        say ('
    Unique to Ada, the language design contracts were
    offered to competing contractors who were to come up
    with independent language designs.  Of over 15 original
    bidders, only four were selected to continue.

    In an effort to keep the design evaluations as unbiased
    as possible, each design was referred to only by a
    color (red, green, blue, or yellow). After the four
    preliminary designs were reviewed by both military and
    civilian agencies, the Red team (Intermetrics) and
    Green team (Honeywell Bull) were selected to continue
    designing.  In the end, the French based Honeywell Bull
    team won the development contract.
').
        close_window ().
        end. (* contracted to competitive bidders *)

(*** END: CONTRACTED TO COMPETITIVE BIDDERS *)

(*** START: DEVELOPED BY A CORPORATE TEAM *)

        topic 'developed by a corporate team'.
        window ('Developed by Corporate Team'
                ,,,,4,4,74,18).
        say ('
    In contrast to the "normal" development by an
    individual or committee, Ada was developed by an entire
    team of programmers from the french based Honeywell
    Bull corporation.  Design decisions were made or
    approved by the team leader, Frenchman Jean Ichbiah.
').
        close_window ().
        end. (* developed by a corporate team *)

(*** END: DEVELOPED BY A CORPORATE TEAM *)

(*** BEGIN: ESTABLISHED ADMINISTRATIVE CONTROLS *)

        topic 'established administrative controls'.
        window ('administratively controlled Ada'
                ,,,,4,4,74,18).
        say ('
    The DoD placed administrative controls on the language
    to prevent the same types of language non-portability
    problems it currently faced.  Additionally, the DoD
    intended to save Ada from the fate of languages like
    FORTRAN and JOVIAL, which have numerous versions and
```

```
dialects.  To control growth of and changes to the
language syntax, the DoD early in the process #msecured
an Ada trademark#m and #mforbade subsets or supersets#m
of the language.

On a larger scale, and to provide for controls on
language implementation, the DoD #mset requirements for
compilers#m including their development and validation.
Lastly, the DoD #mplanned for a programming support
environment#m, nearly from the outset.  It is the
consideration of these issues so early in the design
and development processes which contributes to Ada''s
distinction from other languages.
').

(**** BEGIN: SECURED AN ADA TRADEMARK *)

            topic 'secured an Ada trademark'.
            window ('secured Ada trademark',,,,5,5,72,17).
            say ('
    Ada was granted a trademark in 1981 to prevent the
    validation of compilers which do not conform to the
    standard Ada language.  TRAC and SIMSCRIPT are perhaps
    the only other trademarked languages.
').
            close_window ().
            end. (* secured an Ada trademark *)

(**** END: SECURED AN ADA TRADEMARK *)

(**** BEGIN: FORBADE SUBSETS OR SUPERSETS *)

            topic 'forbade subsets or supersets'.
            window ('forbade subsets or supersets'
                    ,,,,5,5,72,17).
            say ('
    The DoD insisted from early on in the development
    process that no subsets or supersets of Ada would be
    allowed.  This requirement would greatly enhance the
    chances of Ada''s portability.  However, some
    exceptions had to be made for the earliest prototypes
    of Ada and its compilers.  But now, no subset or
    superset of Ada can be recognized as true Ada,  and no
    compiler which allows a subset or superset can be a
    validated Ada compiler.   The only other language to
    forbid  these dialects was MIT''s COMIT, which was
    eventually  superseded by SNOBOL.
').
            close_window ().
            end. (* forbade subsets or supersets *)

(**** END: FORBADE SUBSETS OR SUPERSETS *)
```

```
(**** BEGIN: SET REQUIREMENTS FOR COMPILERS *)

            topic 'set requirements for compilers'.
            window ('established requirements for Ada
                     compilers',,,,5,5,72,17).
            say ('
      The DoD handled the validation of Ada compilers in a
      manner different from any other language.
      Specifically, the DoD planned the validation process as
      an integral part of the entire language design and
      development process.  The DoD expressly requires Ada
      compilers used on DoD projects to be validated, which
      means they compile only true Ada as expressed in the
      STEELMAN document.  Compilers not used on military
      projects are not required to be validated, but most
      companies procure validated compilers to ensure as much
      portability with other compilers as possible.
').
            close_window ().
            end. (* set requirements for compilers *)

(**** END: SET REQUIREMENTS FOR COMPILERS *)

(**** BEGIN: PLANNED FOR A PROGRAMMING SUPPORT ENVIRONMENT*)

            topic 'planned for a programming support
                   environment'.
            window ('planned support environment'
                     ,,,,5,5,72,17).
            say ('
      The DoD established requirements for a formal Ada
      programming Support Environment (APSE).  APSE
      requirements are #mspelled out in STONEMAN#m, the last
      in a series of revisions of environment requirements
      documents.  An APSE aids in Ada portability and promote
      more effective use of the complex language by
      #mproviding an integrated toolset#m for the Ada
      programmer.
').

(***** BEGIN: SPELLED OUT IN STONEMAN *)

               topic 'spelled out in STONEMAN'.
               window ('enumerated in STONEMAN document'
                        ,,,,6,6,70,16).
               say ('
      The STONEMAN document is so far the DoD''s last word on
      requirements for an integrated Ada programming support
      environment (APSE).  STONEMAN describes a three-part
      APSE, consisting of progressively higher-level
```

```
     sections, the KAPSE, MAPSE, and APSE, respectively.
     Briefly,

         KAPSE: Kernel APSE
                Unique to specific host operating system

         MAPSE: Minimal APSE
                Minimal set of support tools
                Built on top of KAPSE

         APSE:  User interface to MAPSE toolset
                Additional user tools
').
               close_window ().
               end. (* spelled out in STONEMAN *)

(***** END: SPELLED OUT IN STONEMAN *)

(***** BEGIN: PROVIDING AN INTEGRATED TOOLSET *)

               topic 'providing an integrated toolset'.
               window ('provide integrated toolset'
                        ,,,,6,6,70,16).
               say ('
     The APSE provides an integrated toolset to aid a
     programmer in taking full advantage of the complex Ada
     language.  In addition to the compiler, the basic tools
     include:
            editor              configuration manager
            linker              JCL interpreter
            loader              debugger

     The Common APSE Interface Set (CAIS) is a set of Ada
     packages which provide for interfaces between these
     tools.
').
               close_window ().
               end. (* providing an integrated toolset *)

(***** END: PROVIDING AN INTEGRATED TOOLSET *)

            close_window ().
            end. (* planned for a programming support
                    environment *)

(**** END: PLANNED FOR A PROGRAMMING SUPPORT ENVIRONMENT *)

         close_window ().
         end. (* established administrative controls *)

(*** END: ESTABLISHED ADMINISTRATIVE CONTROLS *)
```

```
(*** BEGIN: ENCOURAGED PUBLIC FEEDBACK *)

        topic 'encouraged public feedback'.
        window ('Solicited Public Review and Comment'
                ,,,,4,4,74,18).
        say ('
    A distinguishing element of the Ada design and
    development processes was that throughout, the DoD
    solicited and considered review and comment from the
    international programming community.  Public comments
    were used in the series of revisions from STRAWMAN to
    STEELMAN and in the evaluation of candidate language
    designs.  Also, the prototypical Preliminary Ada was
    published in ACM SIGPLAN for commentary and review.
').
        close_window ().
        end. (* encouraged public feedback *)

(*** END: ENCOURAGED PUBLIC FEEDBACK *)

     close_window ().
     end. (* processes which produced Ada *)

(** END: PROCESSES WHICH PRODUCED ADA *)

(** BEGIN: ATTRACTED STRONG INTEREST *)

     topic 'attracted strong interest'.
     window ('Attracted Strong Interest Outside DoD'
             ,,,,3,3,76,19).
     say ('
    Unlike other DoD sponsored programming languages, (e.g.
    JOVIAL, CMS-2, and TACPOL), sponsored languages, Ada
    has attracted strong interest from the international
    programming community.  Ada has been #mutilized in
    commercial#m and #mindustrial#m sectors in the U.S. and
    in #mEurope#m, in military and research applications.
    Additionally, Ada is the only language which has
    #msparked publications and conferences#m which are
    dedicated solely to the advancement of the language.
').

(*** BEGIN: UTILIZED IN COMMERCIAL *)

        topic 'utilized in commercial'.
        window ('Accepted in Commercial Setting'
                ,,,,4,4,74,18).
```

```
        say ('
    Ada''s wide acceptance has made it of interest to the
    international commercial sector.  Ada compilers, tools,
    and software packages are slowly becoming commercial
    items, available through commercial software vendors.
    Additionally, there are several companies whose sole
    business is developing Ada compilers and tools.
').
        close_window ().
        end. (* commercial *)

(*** END: UTILIZED IN COMMERCIAL *)

(*** BEGIN: UTILIZED IN INDUSTRIAL *)

        topic 'industrial'.
        window ('Accepted in Industrial Sector'
                ,,,,4,4,74,18).
        say ('
    Ada is being used in industrial applications which are
    not related to military projects.  For example, a non-
    military related trucking company has used Ada for its
    data processing needs.  Ada''s readability coupled with
    facilities for numerical and data processing have
    gained Ada a foothold in industry, in both embedded and
    non- embedded software systems applications.
').
        close_window ().
        end. (* industrial *)

(*** END: UTILIZED IN INDUSTRIAL *)

(*** BEGIN: ACCEPTED IN EUROPE *)

        topic 'Europe'.
        window ('Accepted in European Community'
                ,,,,4,4,74,18).
        say ('
    Ada''s applicability to embedded software systems
    accounts for its growing popularity in foreign
    militaries.  European Economic Community (EEC) and NATO
    countries are beginning to use Ada in embedded weapons
    systems and for research.
 ').
        close_window ().
        end. (* military *)

(*** END: ACCEPTED IN EUROPE *)
```

```
(*** BEGIN: SPARKED PUBLICATIONS AND CONFERENCES *)

        topic 'sparked publications and conferences'.
        window ('Cause for International Publications and
                 Conferences',,,,4,4,74,18).
        say ('
    Ada is the subject of several publications and special
    interest groups.  ACM SIGAda is dedicated to to Ada
    applications and research.  In addition, Ada has to its
    credit international conferences and quarterly special
    interest group meetings.  No other language has
    received this intense international attention.
').
        close_window ().
        end. (* sparked publications and conferences *)

(*** END: SPARKED PUBLICATIONS AND CONFERENCES *)

     close_window ().
     end. (* attracted strong interest *)

(** END: ATTRACTED STRONG INTEREST *)

(** BEGIN: OFFERS HIGH-LEVEL FEATURES *)

   topic 'offers high-level programming features'.
   window ('Offers High-Level Programming Features'
            ,,,,3,3,76,19).
   say ('
     From a technical perspective, Ada offers many specific
     high- level programming features provided by no other
     single language.  Ada combines some of the features of
     Pascal, ALGOL, and PL/I, with some of its own unique
     features.  The most important of Ada''s features is the
     facility of packaging.  Ada also offers:

           strong data typing      real-time processing
           generics                exceptions
           tasking                 overloading
           numeric processing      separate compilation
                     representation clauses
').
   close_window ().
   end.  (* offers specific high-level features *)

(** END: OFFERS HIGH-LEVEL FEATURES *)
```

```
   (** BEGIN: SUPPORT FOR SOFTWARE ENGINEERING PRINCIPLES *)

   topic 'support for software engineering principles'.
   window ('Supports Software Engineering Principles'
           ,,,,3,3,76,19).
   say ('
     In many ways, Ada directly supports the principles of
     software engineering. Ada is designed around the
     software component, as reflected in the modular
     properties of separately compilable packages and
     subprograms.  Ada supports abstraction and information
     hiding by providing strong data typing and private
     types.  These constructs govern visibility or access to
     code.  Ada generics and stubbed package specifications
     lend support to reusability and modularity concerns.
     Overall, Ada supports many software engineering
     desirables including:

        structured programming          reusability
        top-down development            modularity
        strong data typing              portability
        abstraction                     readability
        information hiding              verifiability
        encapsulation
        separately compilable specification and body
').
   close_window ().
   end. (* support for software engineering principles *)

     (** END: SUPPORT FOR SOFTWARE ENGINEERING PRINCIPLES *)

close_window ().
end. (* Ada is different *)

(* END: ADA IS DIFFERENT *)
```

Appendix C: Student Comments from Experimental Group Attitudinal Surveys

This appendix presents the comments provided on the attitudinal surveys referenced in Chapter 5. The original surveys were not satisfactorily reproducible by photocopy, so the students' verbatim comments and responses to each question are transcribed here.

QUESTION # 1. How was the computer-based presentation of information better or worse than reading a journal article covering the same information? Be specific, please.

RESPONSES:

Better than reading a journal, BUT in a journal the reader does not need to be disciplined as much. 1) a journal follows a logical sequence without the need to skip around. Using hypertext is not natural since it is not the way we are trained to read (i.e. beginning to end). I think it would be more effective but it would take some definite practice to reach a level of competence using that method.

I would rather have had the journal, since then it would be easier to scan the information. At times, during the lab, I would be in a leg of the hypertext facility and want to quickly review something in another leg therefore I had to back all the way out and back down the correct leg. Cumbersome!

If you visit nodes in wrong order, text seems disjointed, and there are no clues to the proper order.

Much better in conveying organization of material.

Better -- special attention is brought to key points.

At this point, being a new form of information processing, it just takes time to get used to. I'm used to reading in continuous paragraphs without jumping back and forth, so it's worse than a normal journal article.

It was better, but it was confusing at first to use F3, F4 keys to change topics.

It is better in that it allows you to read only what interests you. If you don't want to link to another node, you don't have to. I do not like reading on computers for more than 10 or 15 minutes. If it takes any longer I would rather have a piece of paper in front of me.

It did provide an outline using windows, though one could lose frame of reference or "big picture."

QUESTION # 2. What percentage of the hypertext nodes did you explore as you read the information in the tutorial? (Number of responses indicated in parentheses below)

0-25%	26-50%	51-75%	76-99%	100%
(0)	(0)	(1)	(4)	(4)

RESPONSE from single student who marked "51-75%":

> I think. It was hard to remember what nodes I had already looked at.

QUESTION # 3. With respect to the number of nodes, there were... (circle one of a-c, plus d if needed)

a. too few/too sketchy in content/detail
b. about enough to present sufficient detail
c. too many/too much detail irrelevant to the topic
d. other comment (please elaborate)

Number of responses for each choice were:

a. 1 b. 7 c. 1 d. 1*

* One student circled both "a" and "d" with the comment:

> Some seemed incomplete.

QUESTION # 4. When you used F4 to activate a link to additional text, did the text which ensued contain the content and relevant detail of information you expected/needed? Elaborate.

RESPONSES:

Yes, but it caused you to lose your previous train of thought so if there were several levels of information when you returned, you needed to reread the information to return your train of thought at each level.

Usually

Yes, although sometimes I was surprised to get more information than I expected.

Levels of "detail" was consistent across all nodes: good.

The content appeared to coincide with the "topic heading." Regarding relevant detail it would seem that if a topic is new to you then you wouldn't know what level of detail existed, or what if any of it was relevant.

Yes, but as you used F3 on that screen then F4 then F3 on next screen it started to get confusing.

Usually too much. I think nodes off the main path should be very brief.

Yes, but using F4 broke up the frame of thought from sentence you were reading.

QUESTION # 5. Briefly comment on the tutorial with respect to:

Readability:

Comprehensiveness:

Understandability:

RESPONSES:

Readability:

Easy to read, but lose train of level as you go back up the levels.

Reverse video nodes make the text very unreadable. After 15 minutes I had had enough.

OK, but assumed knowledge of Ada "jargon" in some cases.

Very easy/good.

Technical note: The highlighted fields could have been displayed with different colors. The black boxes interrupted the flow of my reading.

Flowed very well -- easy to read.

OK

Very good.

It was easy to read.

Comprehensiveness:

Could have retained more with a little more practice with the new method.

Too many legs/nodes. Seeing the nodes at each level made me feel I had to look at them.

Some seemed incomplete.

Good.

Easy to learn. Very efficient with 2 keystrokes.

Too many screens.

Very good. A good article.

I understood it since we are in the last couple weeks of [MATH] 555. Might need more comprehensiveness if it was the beginning of the quarter.

Understandability:

No problem once my train of thought was re-established.

Below average.

Very good. Written well.

Without the tutorial it might have been hard to understand what the program did.

Flowed very well, easy to read.

OK.

Very good. The article was good, but the traversing of nodes tended to detract from the main article.

I understood it since we are in the last couple weeks of [MATH] 555. Might need more comprehensiveness if it was the beginning of the quarter.

QUESTION # 6. What SINGLE change would you recommend for the tutorial which would help you, the reader, understand the relationships between and relative detail associated with the concepts presented in the tutorial?

RESPONSES:

One thing that would help is a distinguishing mark that allows the reader the ability to mark previously read text. A couple of times, I found myself rereading text because as I moved up the levels I couldn't remember what I had read and what I hadn't.

If there was some way to remove the reverse video, and subsequently use any mouse click on the 1st sentence to send the reader to a more detailed section; any mouse click on the 2nd sentence (or particular phrases of the second sentence); 3rd, 4th.... This way the reader could thoroughly digest the 1st page and subsequently return to the beginning to select words, ideas, sentences which he wanted more information on, not simply what the developer felt the user needed.

Flesh out some areas more, otherwise OK.

None comes to mind. Since the tutorial was presented as "evaluate this" rather than "learn the material in the tutorial," some of the detailed questions were difficult.

A drawing encompassing all the highlighted fields would give the reader an overall view of what the tutorial was trying to present. NOTE: It is likely that you noticed that I failed to read and maintain the information presented. When asked to evaluate a piece of software, I did not make an effort to retain what was presented. I only focused on the How! Sorry.

I would not allow more than 2 or 3 levels in depth.

A glossary of terms for the person being introduced to Ada.

Bibliography

Barney, Cliff. "Hypertext pioneer van Dam recalls two decades of research, progress," MacIntosh Today, 1: 26 (November 23, 1987a).

-----. "Hypertext plays to packed house," MacIntosh Today, 1: 1+ (November 23, 1987b).

Beck, J. Robert, and Donald Z. Spicer. "Hypermedia in Academia," Academic Computing, 2: 22+ (1988).

Beeman, W. O., Kenneth T. Anderson, Gail Bader, James Larkin, Anne P. McClard, Patrick McQuillan, and Mark Shields. "Hypertext and Pluralism: From Lineal to Non-Lineal Thinking," Hypertext '87 Papers. 67-88. Chapel Hill, NC: Hypertext Planning Committee, November 1987.

Booch, Grady. Software Engineering with Ada. Menlo Park, CA: Benjamin-Cummings, 1983.

-----. Software Engineering with Ada (Second Edition). Menlo Park, CA: Benjamin-Cummings, 1987.

Boulet, Marie-Michèle. "Educational Software Design Using a Diagnostic Approach," Computers and Education, 11: 219-228 (1987).

Bralick, Capt. William A. An Examination of the Theoretical Foundations of the Object-Oriented Paradigm. MS thesis, AFIT/GCS/MA/88M-01. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1988.

Burns, Hugh, Commander, Intelligent Systems Branch. Telephone interview. USAF Human Resources Laboratory, Brooks AFB TX, 20 April 1988.

Campbell, D. T. and J. C. Stanley. Experimental and Quasi-Experimental Designs for Research. Chicago: Rand Mcnally, 1963.

Charniak, Eugene and Drew McDermott. Introduction to Artificial Intelligence. Reading, MA: Addison-Wesley Publishing Company, 1984.

Charney, Davida. "Comprehending Non-Linear Text: The Role of Discourse Cues and Reading Strategies," Hypertertext '87 Papers. 109-120. Chapel Hill, NC: Hypertext Planning Committee, November 1987.

Collier, H. W., Carl B. McGowan, and William T. Ryan. "Microcomputers: a Successful Approach to Teaching Business courses," Computers and Education, 11: 143-148 (1987).

Conklin, Jeff. A Survey of Hypertext. MCC TR No. STP-356-86, Rev. 2. Austin: Microelectronics and Computer Technology Corporation, 3 December 1987a.

-----. "Hypertext: An Introduction and Survey," IEEE Computer, 20: 17-41 (September 1987b).

CRC. Standard Mathematical Tables (Twenty-first edition). Cleveland: The Chemical Rubber Co. Press, 1973.

Dickinson, LtCol Michael T. "Whither CAI? Instructional Design Aspects of Computer Assisted Instruction," 1985 Air Force Conference on Technology in Training and Education (TITE) Proceedings. III-1 - III-5. Colorado Springs: (publisher unavailable) 1985.

Enger, Maj Rolf C., Maj R. E. Swanson, Maj L. W. Schrock, and Capt M. V. Tollefson. "Innovations in Physics Teaching at the USAF Academy Curriculum Changes to Incorporate Computer Aided Instruction," 1985 Air Force Conference on Technology in Training and Education (TITE) Proceedings. I-33 - I-52. Colorado Springs: (publisher unavailable) 1985.

EVB. An Object Oriented Design Handbook. Rockville, MD: EVB Software Engineering, Inc., 1985.

Gardner, J. R., A. McEwen, and C. A. Curry. "A Sample Survey of Attitudes to Computer Studies," Computers and Education, 10: 293-298 (1986).

Garrett, L. N., Karen E. Smith, and Norman Meyrowitz. "Intermedia: Issues, Strategies, and Tactics in the Design of a Hypermedia Document System," Computer-Supported Cooperative Work (CSCW '86) Proceedings. Austin: (publisher unavailable) 1986.

Gowin, D. Bob, Author. Telephone interview. Cornell University, Ithaca, NY, 18 August 1988.

Hammwöhner, Rainer and Ulrich Thiel. "Content Oriented Relations between Text Units -- a Structural Model for Hypertexts," Hypertertext '87 Papers. 155-174. Chapel Hill, NC: Hypertext Planning Committee, November 1987.

Hativa, Nira. "The Microcomputer as a Classroom Audio Visual Device: the Concept, and Prospects for Adoption," Computers and Education, 10: 359-368 (1986).

Hayward, S. A., B. J. Wielinga, and J. A. Breuker. "Structured Analysis of Knowledge," International Journal for Man-Machine Studies, 26: 487-498 (April 1987).

Hershey, W. "Guide," Byte, 12: 244-246 (October 1987).

Hodges, J. C. and M. E. Whitten. Harbrace College Handbook (Ninth edition). New York: Harcourt Brace Jovanovich, 1982.

Knowledge Garden, Inc. KnowledgePro. Instruction Manual. Nassau, NY: 1988.

-----. Welcom[illegible] o TextPro. Instruction Manual. Nassau, NY: 1987.

Landow, George P. "Relationally Encoded Links and the Rhetoric of Hypertext," Hypertertext '87 Papers. 331-344. Chapel Hill, NC: Hypertext Planning Committee, November 1987.

Lawler, Robert W. and Masoud Yazdani. Artificial Intelligence and Education Volume One. Norwood, NJ: Ablex Publishing, 1987.

Linn, Marcia C. Hypermedia and Programming Instruction: Opportunities for Meeting Department of Defense Training Needs. Position paper. Instructional Technology Program, University of California, Berkeley, CA, February 1988.

MacNiven, Major Donald B. Computer-Based Training Systems: Organizing to Use Them. Student Report 87-1615. Air Command and Staff College/EDCC, Maxwell AFB, AL, April 1987.

Marchionini, Gary and Ben Shneiderman. "Finding Facts vs. Browsing Knowledge in Hypertext Systems," IEEE Computer, 21: 70-80 (January 1988).

Marshall, Catherine. "Exploring Representation Problems Using Hypertext," Hypertertext '87 Papers. 253-268. Chapel Hill, NC: Hypertext Planning Committee, November 1987.

Mayes, J. Terrance, Michael R. Kibby, and Hugh Watson. "The Development and Evaluation of a Learning-by-Browsing System on the MacIntosh," Computers and Education 12: 221-229 (1988).

McCaughley, M. P. "Synchronizing a Tape Recorder to an Educational Computer Program," Educational Technology Systems, 15: 401-406 (1986-87).

McFarren, Capt Michael R. Using Concept Mapping to Define Problems and Identify Key Kernels During the Development of a Decision Support System. MS thesis, AFIT/GST/ENS/87J-12. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, June 1987.

Neuwirth, Christine, David Kaufer, Rich Chimera, and Terilyn Gillespie. "The Notes Program: A Hypertext Application for Writing from Source Texts," Hypertext '87 Papers. 121-142. Chapel Hill, NC: Hypertext Planning Committee, November 1987.

Norman, Kent L., Linda J. Weldon, and Ben Shneiderman. "Cognitive Layouts of Windows and Multiple Screens for User Interfaces, International Journal of Man-Machine Studies 25: 229-248 (August 1986).

Novak, Joseph D. A Theory of Education. Ithaca: Cornell University Press, 1977.

Novak, Joseph D. and D. Bob Gowin. Learning How to Learn. Cambridge: Cambridge University Press, 1986.

Ohlsson, Stellan. "Some Principles of Intelligent Tutoring," Artificial Intelligence and Education, Volume One, edited by Robert W. Lawler and Masoud Yazdani. Norwood, NJ: Albex Publishing Company, 1987.

Oren, Tim. "The Architecture of Static Hypertexts," Hypertertext '87 Papers. 291-306. Chapel Hill, NC: Hypertext Planning Committee, November 1987.

O'Shea, Tim and John Self. Learning and Teaching with Computers. Englewood Cliffs, NJ: Prentice-Hall, 1983.

Plambondon, Réjean, and Jean-Guy Déschênes. "Course Design Using Software Engineering Methods," Computers and Education, 10: 417-428 (1986).

Pressman, Roger S. Software Engineering: A Practitioner's Approach (Second edition). New York: McGraw-Hill, 1987.

Psotka, Joseph. "AI Applications to Machine Translation and Language Instruction," (Published in a confidential report of Knowledge-Based Information Systems, title unavailable). The Hague: SHAPE Technical Center, October 1987. Copy furnished by author.

Rambally, G. K. and R. S. Rambally. "Human Factors in CAI Design," Computers and Education, 11: 149-154 (1987).

Raskin, Jef. "The Hype in Hypertext," Hypertertext '87 Papers. 325-330. Chapel Hill, NC: Hypertext Planning Committee, November 1987.

Raymond, J. "The Computerized Overhead Projector," Computers and Education, 11: 181-195 (1987).

Roman, David. "Computer-Based Training Tutorials Well-Taught," Computer Decisions, 17: 82-84+ (30 July 1985).

Shasha, Dennis. "When Does Non-linear Text Help?," Expert Database Systems. Edited by Larry Kerschberg. (city unavailable): The Benjamin/Cummings Publishing Co., Inc., 1987.

Shaw, Donna G., K. M. Swigger, and J. Herndon. "Children's Questions: a Study of Questions Children Ask while Learning to Use a Computer," Computers and Education, 9: 15-20 (1985).

Shulman, L. S. and C. Ringstaff. "Current Research in the Psychology of Learning and Teaching," Designing Computer-Based Learning Materials, edited by Harold Weinstock and Alfred Bork. New York: Springer-Verlag, 1985.

Smith, John B., Stephen F. Weiss, and Gordon J. Ferguson. "A Hypertext Writing Environment and its Cognitive Basis," Hypertertext '87 Papers. 195-214. Chapel Hill, NC: Hypertext Planning Committee, November 1987.

Smith, Karen E. "Hypertext -- Linking to the Future," Online, 12: 32-40 (March 1988).

Thompson, Bev and Bill Thompson. "Hyping Text: Hypertext and Knowledge Representation," AI Expert, 2: 25-28 (August 1987a).

-----. Announcing KnowledgePro. Company Brochure. Knowledge Garden, Inc., Nassau, NY, 1987b.

Tombaugh, J., A. Lickorish, and P. Wright. "Multi-window Displays for Readers of Lengthy Texts," International Journal for Man-Machine Studies, 26: 597-615 (May 1987).

Verano, Capt Miguel. "Interactive Videodisc and the Computer," 1985 Air Force Conference on Technology in Training and Education (TITE) Proceedings. I-71 - I-79. Colorado Springs: (publisher unavailable) 1985.

Walker, Swen A. Student. Figure provided by author. Air Force Institute of Technology, Wright-Patterson AFB OH, 1 October 1988.

Webster's New Collegiate Dictionary. Springfield, MA: G. & C. Merriam Co., 1981.

Wilcox, Kenneth R. Student. Figure provided by author. Air Force Institute of Technology, Wright-Patterson AFB, OH, 1 February, 1988.

Williams, Gregg. "HyperCard," Byte, 12: 109-117 (December 1987).

Winston, Patrick H. Artificial Intelligence. Reading, MA: Addison-Wesley Publishing Company, 1984.

Woolfolk, Anita E. and Lorraine McCune-Nicolich. Educational Psychology for Teachers. Englewood Cliffs, NJ: Prentice-Hall, 1984.

Yankelovich, Nicole, Bernard J. Haan, Norman K. Meyrowitz, and Steven M. Drucker. "Intermedia: The Concept and the Construction of a Seamless Information Environment," IEEE Computer, 21: 81-96 (January 1988).

Yankelovich, Nicole, George Landow, and Peter Heywood. Designing Hypermedia "Ideabases" -- The Intermedia Experience. IRIS Technical Report: 87-4. Providence: Institute for Research in Information and Scholarship, Brown University, 1987.

Yankelovich, Nicole, Norman Meyrowitz, and Andries van Dam. "Reading and Writing the Electronic Book," IEEE Computer, 18: 15-29 (October 1985).

Yazdani, Masoud. "Intelligent Tutoring Systems: An Overview," Artificial Intelligence and Education, Volume One, edited by Robert W. Lawler and Masoud Yazdani. Norwood, NJ: Albex Publishing Company, 1987.

Additional Sources

Chasse, James and Joseph J. Rogowski. "The Paperless Manual," The Army Logistician (PB 700-87-4): 40-42 (September/October 1986).

Collier, George H. "Thoth-II: Hypertext with Explicit Semantics," Hypertertext '87 Papers. 269-290. Chapel Hill, NC: Hypertext Planning Committee, November 1987.

Conklin, Jeff. Telephone interview. Microelectronics and Computer Technology Corporation, Austin TX, 7 January 1988.

Martin, Merle P. and Willian L. Fuerst. "Using Computer Knowledge in the Design of Interactive Systems," International Journal for Man-Machine Studies, 26: 333-342 (March 1987).

Montgomery, D. C. Design and Analysis of Experiments (Second edition). New York: Wiley and Sons, 1984.

Vita

First Lieutenant Michael L. Talbert was born on [redacted] In 1981 he graduated as class co-valedictorian from Great Falls High School in Great Falls, South Carolina. In 1985 he was graduated Magna Cum Laude from North Carolina State University with a Bachelor of Science degree in meteorology. Lt. Talbert received his commission through the Air Force ROTC program at N. C. State, where he was named a Distinguished Graduate. He then served as Wing Weather Officer to the 97th Bombardment Wing at Blytheville (now Eaker) Air Force Base, Arkansas, until entering the School of Engineering, Air Force Institute of Technology, in May 1987.

Permanent Address:

[redacted]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/MA/GCS/88D-01	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (*If applicable*) AFIT/ENC	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (*City, State, and ZIP Code*) Air Force Institute of Technology WPAFB, OH 45433-6583	7b. ADDRESS (*City, State, and ZIP Code*)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Institute of Tech. Dept. of Math and Comp. Sci. 8b. OFFICE SYMBOL (*If applicable*) AFIT/ENC	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*) Air Force Institute of Technology WPAFB, OH 45433-6583	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
AN OBJECT-ORIENTED APPROACH TO THE DEVELOPMENT OF COMPUTER-ASSISTED INSTRUCTIONAL MATERIAL USING HYPERTEXT

12. PERSONAL AUTHOR(S)
Talbert, Michael Lane, 1Lt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	December 1988	170

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Object, Object-Oriented, Hypertext, Computer-Assisted Instruction (CAI)

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

THESIS CHAIRMAN: David A. Umphress, Capt, USAF
Assistant Professor of Computer Science

ABSTRACT: (see reverse)

JS Rawlinski
10 Jan 89

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Capt. David A. Umphress	22b. TELEPHONE (*Include Area Code*) 513-255-3098 22c. OFFICE SYMBOL AFIT/ENC

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

This thesis combines the concepts of learning theory, knowledge engineering, software engineering, and hypertext. It presents a methodology for creating formally structured hypertext-based documents for use in transfer learning-oriented computer-based tutorials. The methodology, which parallels the proven object-oriented software design paradigm (OOD), facilitates the decomposition of a knowledge base into a hierarchical structure of text passages which form a tutorial. Application of the methodology results in encapsulated text objects which demonstrate many of the desirable software characteristics (e.g. modularity, cohesion). Evaluation of the methodology showed that it produced a computer-based tutorial which facilitates a learner's relationship-oriented assimilation of the concepts presented in the tutorial.