TOWARDS A DOCUMENT STRUCTURE EDITOR FOR SOFTWARE REQUIREMENTS ANALYSIS

Vincent J. Kowalski and
Dr. Anthony A. Lekkos, University of Houston
Clear Lake

1. Introduction

Of the six or seven phases of the software engineering life cycle, requirements analysis tends to be the least understood and the least formalized. Correspondingly, a scarcity of useful software tools exist which aid in the development of user and system requirements.

In this paper we propose that requirements analysis should culminate in a set of documents similar to those that usually accompany a delivered software product. We present the design of a software tool, the Document Structure Editor, which facilitates the development of such documentation.

The requirements analysis phase of the software engineering life cycle may be defined as the phase of software development in which the requirements of the user of a proposed software package are identified in a precise, complete and logically coherent manner [6,7]. System constraints that result from the target hardware as well as non-functional constraints such as budget, time, and human resources must also be a part of a complete requirements analysis.

Two approaches to the problem of representing software requirements that appear frequently in the literature are:

- natural (textual) language approach [10, 12]
- formal representation approach [3, 5, 9, 12, 19]

The first of these attempts to specify requirements in a manner that is easily developed and understood by humans. It has the disadvantage that it may give rise to logically incorrect sets of requirements. The second approach, though it prevents logical inconsistencies, has as its main drawback the fact that a formal language must be used. This is not necessarily a desirable situation since user requirements are best provided by users, not programmers.

Several software packages are spoken of as aids to the requirements analysis phase of the software engineering life cycle. A list of some of the more well-known of these packages is the following:

- PSL/PSA [21]
- SREM [17, 18]
- SADT [15, 16]
- SSA [8]
- HOS [2, 13]
- Gist [1]

A close examination of the above tools has revealed that they are more suitably classified as program design and structure tools. Though the design of code is an essential phase in the software engineering life cycle, it is most appropriately thought of as largely independent of requirements analysis.

Finally, the relative importance of good requirements analysis is the motivation for this work. Several studies have shown that the further a software project is along in the software engineering life cycle, the more difficult and costly it is to fix bugs, make changes, and add new requirements [4, 11]. As we have found, requirements are a difficult part of software development because of the lack of automated tools that specifically aid requirements generation and maintenance

2. Document Structure Editor

2.1 Purpose and Goals

The complete set of documentation that in general accompanies a delivered software package provides a very complete set of requirements for that software package. Such documentation is, however, usually developed after the code for the package has been designed, implemented and tested. Examples of such documentation include:

- General Information Manual
- User Manual
  - Language (or Command) Reference
  - Guide
  - Tutorial
- System Requirements Document

The general goal of the Document Structure Editor is to provide an automated software tool for the development and subsequent management of documents such as those listed above. The most important feature of the DSE is that once the general structure of such a document is determined it may be stored as a Template for use in the generation of other similarly structured documents.

2.2 System Overview

The Document Stucture Editor system is depicted in Figure 1. At the highest level of the system are the users. Next, the users' interface to the system consists of a set of commands supplied by DSE. This interface may be taylored to a user's particular needs and in essence each user has his or her own interface to the DSE. Commands are interpreted at the next lower level in the system. These commands invoke any combination of the lowest-level components of the system. These lowest-level components are:

- Structure file
- DBMS
- Panel Primitives
- Text / Graphics Editor

The Structure File is the internal data structure that reflects the structure of a given document. In most cases, this structure will be hierarchical. The DBMS is used for archival of document Templates and the data associated with particular documents. Panel Primitives are the software packages in the DSE which perform the necessary mappings between the Structure File and a particular CRT or workstation. Finally, the Text / Graphics Editor is the means by which a user enters data (text and digitized graphical objects) into the DSE.

2.3 Basic Terminology

Below are listed the definitions of terms defined in the Document Stucture Editor. Several of the terms defined below are illustrated by Figure 2. Figure 2 is an example of a document or Template stored in the DSE. It should be noted that the document is divided into parts, which are further divided into chapters, which in turn are divided into sections. This structure is typical of most technical writing and is easily developed and stored by the DSE.

Topic The atomic unit of a document. A Topic consists of a Heading and a Body. In actual documents, a Topic may be thought of as a generic term for parts, chapters, sections, and subsections.

Heading This is a line of text that comes at the top of a Topic. A Topic must have a Heading. In a real document, a Heading may be a title, the name of a chapter or the like.

Body The Body is the content portion of a Topic. A Topic does not need to have a Body (although the DSE reserves space for a Body in every Topic).

Level The Level of a Template is how far up or down in a document's hierarchical structure you are. For example, the title of a textbook is its 0th Level, the parts are its 1st Level, the chapters are its 2nd Level, the sections are its 3rd Level, the subsections are its 4th Level and so on.

Depth Given a Level, how many Levels are contained within it. If we talk about a document (or Template) that has a title, chapters, and sections, the Depth of the Level that corresponds to the title is 3 (you include the Level you are looking at).

Breadth The Breadth of a Level is how may Topics are contained within that Level. In other words, if Template has five chapters and the Level being considered is that which corresponds to chapters, that Level has a Breadth of 5.

Template This is the sum total of all the Topics and their assigned Levels-the total document under development.

Menu A Menu is a special Command available to users of the DSE. Menu is used to build selection menus and may be invoked by a Profile or a User-Defined Command.

Profile Profiles are user-written files that consist of DSE commands and system-related commands. A profile is executed when the DSE system is entered at (or enters) some particular point. For instance, when a user logs on the sytem, the User Profile is immediately executed.

Command -Line The typing-in of DSE or User Commands is performed on a space on the terminal screen called the Command Line. Commands entered in this fashion are referred to as Line Commands.

the Level you are looking at).

Breadth The Breadth of a Level is how may Topics are contained within that Level. In other words, if Template

Breadth The Breadth of a Level is how may Topics are contained within that Level. In other words, if Template has five chapters and the Level being considered is that which corresponds to chapters, that Level has a Breadth of 5.

Template This is the sum total of all the Topics and their assigned Levels-the total document under development.

Menu — A Menu is a special Command available to users of the DSE. Menu is used to build selection menus and may be invoked by a Profile or a User-Defined Command.

Profile — Profiles are user-written files that consist of DSE commands and system-related commands. A profile is executed when the DSE system is entered at (or enters) some particular point. For instance, when a user logs on the sytem, the User Profile is immediately executed.

Command -Line — The typing-in of DSE or User Commands is performed on a space on the terminal screen called the Command Line. Commands entered in this fashion are referred to as Line Commands.

Command -User — A User Defined Command is similar to a Profile, except that a User Defined Command may be invoked anywhere in the system a Command Line is available. The User Command consists of DSE and host system commands and is assigned a name by the user who writes it.

Command -General — Commands are the means by which a user tells the DSE what to do. Commands are the basis for the interface between the user and the DSE system.

Scroll — Scrolling a Template is a feature of the DSE that allows a user to view a Template as one continuous piece of text.

DSE — Software that converts DSE or User Commands into instructions that the host computer understands.

Structure File — A file that contains the Level information and hence the structure of a Template. It is defined here for the purpose of completeness.

Currency — The DSE "knows" what template or topic or whatever you might be referring to by keeping Current values for such items. The currency is usually set using some Select command.

Command -Key — A Key Command is an association (or mapping) between a short key sequence and a DSE or User Command. These associations are defined in a Profile and the last Profile executed takes precedence over any previous Profiles with regards to these Key Command definitions.

3. Related Efforts

In many respects, storing the associated structure of a given document is the logical next step for word processing software packages. Several commercial packages have structure editing capabilities. These packages generally fall into one of two categories:

- Automatic Indexers
- Outliners

Automatic indexing software usually is available as an option to many popular word processors. Outliners, on the other hand, have outlining of documents as their primary purpose with limited word processing capabilities. Such packages run on microcomputers exclusively. In addition, the integration of the components of these packages is questionable [20].

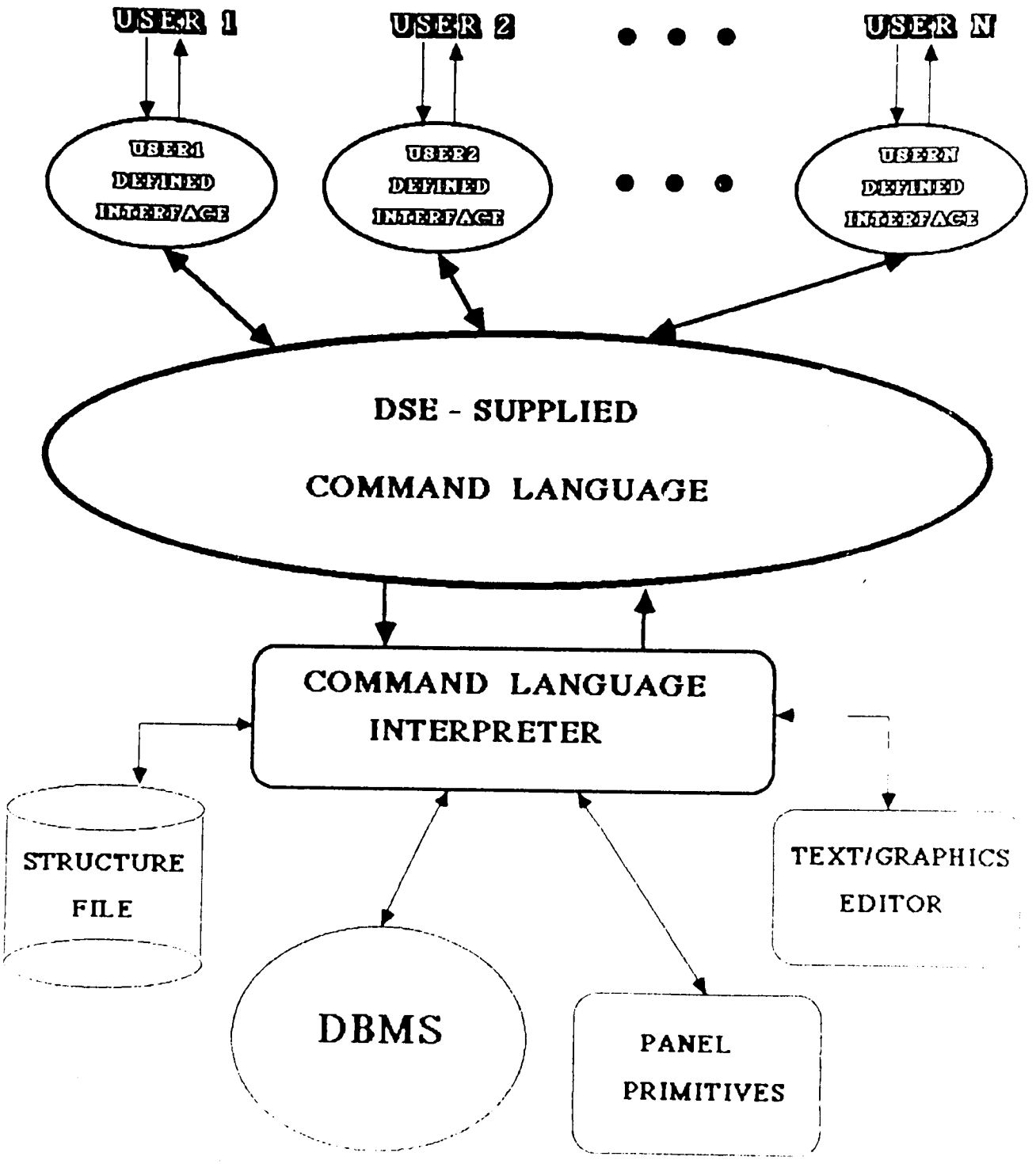
USER 1
USER 2
USER N
USER1 DEFINED INTERFACE
USER2 DEFINED INTERFACE
USERN DEFINED INTERFACE
DSE - SUPPLIED
COMMAND LANGUAGE
COMMAND LANGUAGE INTERPRETER
STRUCTURE FILE
TEXT/GRAPHICS EDITOR
DBMS
PANEL PRIMITIVES

Figure 1.

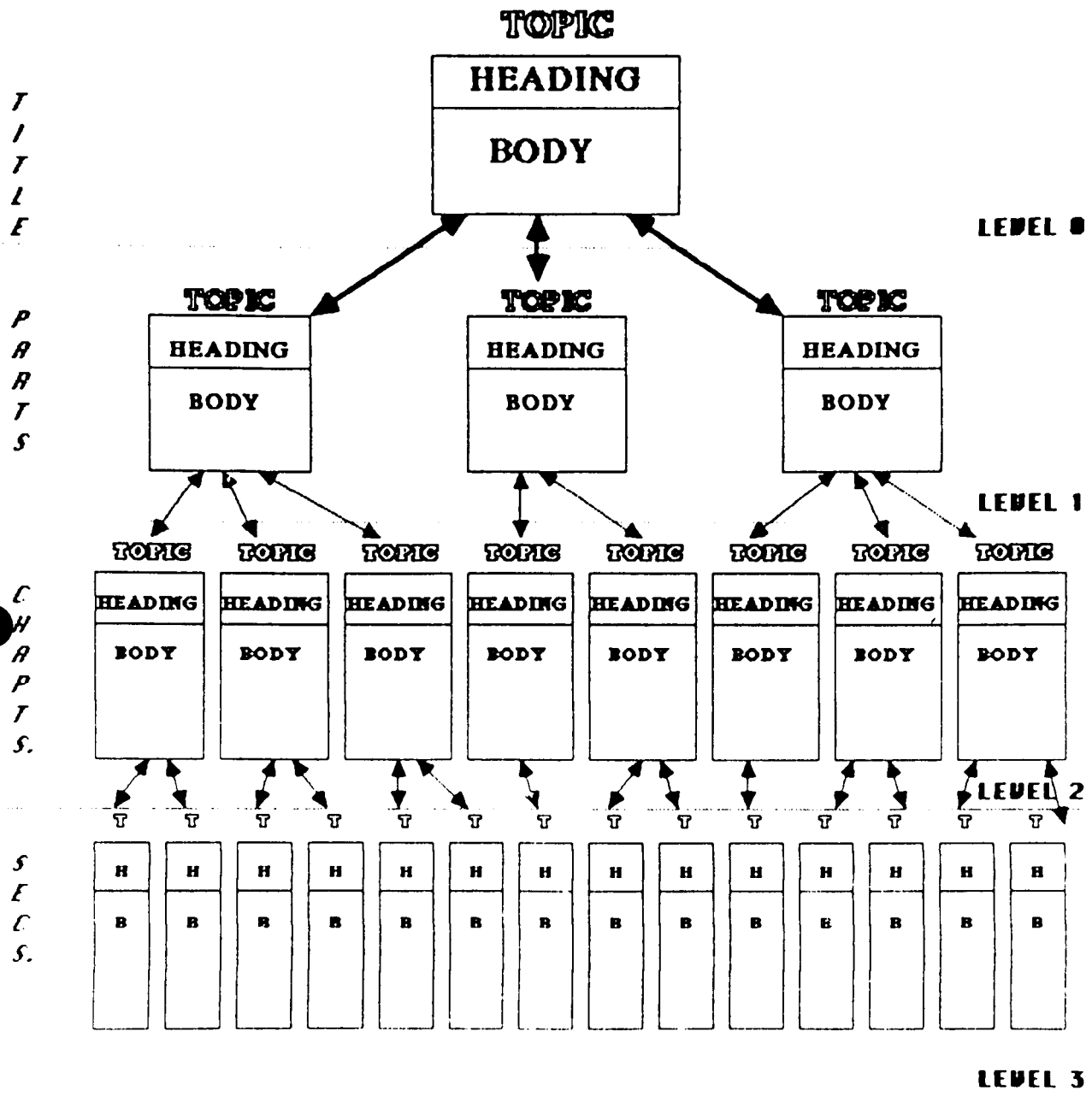
TOPIC
HEADING
BODY
TITLE
LEVEL 0
TOPIC
HEADING
BODY
TOPIC
HEADING
BODY
TOPIC
HEADING
BODY
PARTS
LEVEL 1
TOPIC
HEADING
BODY
CHAPTS.
LEVEL 2
H
B
SECS.
LEVEL 3

Figure 2.

References

[1] Balzer, R., Gist Final Report, Information Sciences Institute, University of Souther California, Feb. 1981.

[2] Barlas, J., "Higher-order computing generates high order business for Cambridge Company," Mass High Tech, Jul. 23, 1984.

[3] Bell, T., et al.: "An Extendable Approach to Computer-Aided Software Requirements Engineering,"Trans. Software Eng., Jan. 1977.

[4] Booch, G., Software Engineering With Ada, Benjamin/Cummings Publishing Co., Menlo Park, California, 1983.

[5] Borgida, A., and Greenspan, S., "Knowledge Representation as the Basis for Requirements Specifications," IEEE Computer, Apr. 1985.

[6] Fairley, R., Software Engineering Concepts, McGraw-Hill Publishing Co., New York, 1985.

[7] Freeman, P., "Requirements Analysis and Specification: The First Step," Advances in Computer Technology, Aug. 1980.

[8] Gane C., and Sarson, T., Structured Systems Analysis: Tools and Techniques, Prentice-Hall, Englewood Cliffs, N.J., 1979.

[9] Greenspan, S., et al., "Capturing More World Knowledge in the Requirements Specification," IEEE Proceedings of the Sixth International Conference on Software Engineering, 1982.

[10] Heninger, K, "Specifying Software Requirements for Complex Systems: New Techniques and Their Application," IEEE Transactions on Software Engineering, Jan. 1986.

[11] Jones, C., Programmer Productivity, McGraw-Hill, New York, 1986.

[12] Levene, A., and Mullery, G., "An Investigation of Requirements Specification Languages: Theory and Practice," IEEE Computer, May 1982.

[13] Martin, J., System Design From Provably Correct Constructs, McGraw-Hill, New York, 1985.

[14] Robinson, L., et al., "A Formal Methodology for the Design of Operating System Software," Current Trends in Programming Methodology, vol I, Prentice-Hall, Englewood Cliffs, N.J., 1977.

[15] Ross, D., "Structured Analysis (SA): A Language for Communicating Ideas," IEEE Transactions on Software Engineering, Jan. 1977.

[16] Ross, D. "Applications and Extensions of SADT," IEEE Computer, Apr., 1985.

[17] Rzepka, W., et al., "Requirements Engineering Environments: Software Tools for Modeling User Needs," IEEE Computer, Apr 1985

[18] Scheffer, P., et al., "A Case Study of SREM," IEEE Computer, Apr 1985

[19] Shaw, A., "Software Specification Languages Based on Regular Expressions," in Software Development Tools, Springer-Verlag, Berlin, 1980.

[20] Spezzano, C., "Unconventional Outliners", PC World, Mar. 1986

[21] Teichrow, D., et al., "PSL/PSA: A Computer Aided Technique for Structured Documentation and Analysis of Information Processing Systems," Trans. Software Engineering, Jan. 1977