HYPERTEXT AND SOFTWARE MAINTENANCE

Spencer Rugaber

College of Computing
and
Software Research Center
Georgia Institute of Technology

1. INTRODUCTION

Hypertext is an appropriate and promising technology that addresses many of the problems confronting software maintainers. Software maintenance is a process that involves analysis, hypothesis construction, and confirmation[2,5]. Hypertext technology offers tools and information structures that can be used to support these activities. Moreover, "Programming in the Large" issues such as version control and requirements tracing are appropriate candidates for the application of hypertext technology.

2. ANALYSIS

The analysis phase of software maintenance can be viewed as a process of exploration within a large collection of information consisting of program documentation and source text. The exploration may have various purposes depending on the specific maintenance task, the experience of the maintainer with the system, and the stage within the maintenance process. For example, inexperienced maintainers may study the information space sequentially, trying to understand the overall structure and purpose of the system. A more experienced maintainer, however, needs to see less of the total information but may need to interleave viewing of the documentation with the source code, trying to understand how a particular section of code implements a specific functional requirement.

Hypertext systems are primarily intended to support exploration of "non-linear text"[3] (also called "hyperspace"). Hyperspace can be thought of as the visual manifestation of a complex information structure. Some hypertext systems are specifically designed to support a specific information structure or application arena. For example, the SAM system facilitates exploration and report generation of a single, albeit large, document that provides guidelines for designing user interface software.[7] Other hypertext systems, such as NoteCards, are designed to deal with general text, allowing the user or system administrator to specify "links" (explicitly designated, possibly attributed, connections) among text "nodes" (sections of text).

Hypertext can manage the large amounts of information associated with a software project and facilitate the exploration described above. Software contains internal structure that can be used to automate the construction of the information structure that underlies a hypertext system. For example, a maintainer may be interested in viewing the statement that declares a program variable, the collection of references (uses) of the variable, or the statements in which the value of the variable can be set[4]. All of this information can be generated relatively easily using standard compiler flow analysis techniques. If the underlying software development methodology supports "requirements tracing" that links user requirements and design information with source code, then those links can readily be incorporated into the information structure.

3. HYPOTHESIS CONSTRUCTION

Software maintenance activities are goal directed: a specific enhancement must be made to a system or a reported bug must be corrected. The exploration associated with these activities leads to the formation of hypotheses proposing one or more locations in the code and documentation at which alterations should be made[6]. The hypotheses may be expressed in a variety of formats ranging from vague suspicions expressed in natural language, to specific textual alterations (code or documentation "fixes"). They may involve several files, containing both code and documentation. Alterations may also be suggested for the user documentation or test suite.

Some hypertext systems support annotation (the ability to append informal textual comments to a node in the information space). Some systems augment the linkage process by allowing the appendage of attribute information to a link to facilitate exploration or retrieval. Finally, others support functional triggers that cause supporting activities to be initiated when a particular link is followed. For example, following a link from a section of software documentation to the related code could cause the recent revision history of the code to be displayed or a warning that another user has currently "checked out" the section of code in order to make an alteration.

4. CONFIRMATION

The next phase of the maintenance process involves confirming a hypothesis constructed by the previous phase. This may involve further exploration to determine possible ramifications of a change, formal modification procedures, or actual tests. The result may be a reformation of the hypothesis, in which case the process returns to the analysis or hypothesis formation stage. Alternatively, a confirmed hypothesis implies other procedures must be followed to formally implement the change into the system.

One role that hypertext can play in this activity relates to version control. Version history is just another form of internal relationship among the information stored in a software engineering environment. This information could be linked to the item being altered by an attributed "history" link. Exploration of the linked information could proceed interactively or be used for automatic generation of reports.

5. SUPPORT FOR "PROGRAMMING IN THE LARGE"

Most software engineering environments collect all project related information into a "project data base"[1]. It includes documents from all phases of the life cycles as well as the history of alterations to the information. This large information collection is used not only as text to be viewed, but also as the base text of various generated reports. A properly organized project data base could also serve as the underlying information structure for a hypertext software maintenance system.

A variety of data base organization methods and data models have been used to implement hypertext systems. These range from unadorned text files through special purpose databases to "off-the-shelf" commercial systems. Usually, the specific database system is largely invisible to the end user. The particular data model and data base implementation are less important than the need to organize the information to promote the "link-following" exploration provided by hypertext systems.

Hypertext systems can also be used as glorified report generators. For example, SAM allows the user to select arbitrary sections of text to comprise a report. Moreover, the resulting report need not exist (and be maintained) as a separate document but can be thought of as an automatically generated product that changes dynamically as its constituents are altered.

6. CONCLUSIONS

Hypertext technology matches well the requirements of an automated software maintenance environment. The information generated by a compiler during its flow analysis phase can be used to support hypertext tools for "programming in the small". The information captured by a project data base can serve the same purpose when larger scale issues are concerned.

REFERENCES

[1] James Bigelow.
Hypertext and CASE.
IEEE Software, March, 1988.

[2] Paul M. Cashman and Anatol W. Holt.
A Communications-Oriented Approach to Structuring the Software Maintenance Environment.
ACM SIGSOFT, Software Engineering Notes, Volume 5, Number 1, January 1980.

[3] J. Conklin.
Hypertext: An Introduction and Survey.
IEEE Computer, September, 1987, pages 17-41.

[4] John R. Foster and Malcolm Munro.
A Documentation Method Based on Cross-Referencing.
Proceedings Conference on Software Maintenance - 1987, IEEE Computer Society, 1987.

[5] R. K. Overton et al.
A Study of the Fundamental Factors Underlying Software Maintenance Problems: Final Report.
Corporation for Information Systems Research and Development, December, 1971, NTIS Numbers: AD 739479 and AD 739872.

[6] R. K. Overton, et al.
Research Toward Ways of Improving Software Maintenance: RICASM Final Report.
Corporation for Information Systems Research and Development, January, 1973, NTIS Number AD 760819.

[7] G. Perlman.
An Overview of SAM: A Hypertext Interface to Smith & Mosier's *Guidelines for Designing User Interface Software.*
Wang Institute Technical Report, WI TR-87-08, 1987.

APPENDIX: DISPLAY LINKAGES FOR A HYPERTEXT SOFTWARE MAINTENANCE TOOL

There are many types of exploration paths that might be supported by a hypertext software maintenance tool. The list below is intended to give a flavor of the possibilities. It is not intended to describe the user interface to such a system.

A. Exploration in a Single File

Within a single source file, the user can explore in the following ways.

+ Sequential presentation; forward and backward scrolling; random access by line number.

+ Search for a specified character string (the consecutive characters "COUNT"); search for a specified lexical entity (the identifier "COUNT" anywhere in the file); search for a specified syntactic construct (the definition of the procedure "COUNT"); search with semantic constraints (the INTEGER variable "COUNT").

+ Access to the declaration, references, and definitions of a variable.

B. Holophrastic Display of Information

The user can control the amount of detail visible at any time. The user can select any component of a display for further magnification. The user can also request that certain sections be "telescoped" (replaced by their outlines).

+ Display the names of the files comprising the system.

+ For a given documentation file, display the major section heading to any degree of resolution.

+ For a given source file, display the major components such as external variable and procedure declarations. Within a procedure, detail can be controlled at the control structure level. For example, a procedure can be viewed as a sequence of paragraphs; a paragraph may be a loop, conditional, or sequence of statements. Paragraphs may be expanded individually.

+ The part of a source file that relates to a specific data item can be viewed as a skeleton. That is, only the lines referring to the data items are displayed.

C. Summary Information

Various types of summary information are typically available to help document a software system. These can be readily accessed assuming a large screen bit-mapped display is available.

+ Diagrams can be viewed such as call graphs and inverse call graphs, file inclusion hierarchy, profile histograms, and statistical summaries.

+ Traditional program cross reference information can be displayed including designation of declarations, definitions, and references. Of course, the user can link from the cross reference display to any of the referenced lines.

+ Cross reference access to documentation files; access to indices, glossaries, and tables of contents.

D. Multiple File Access

All of the linkages described above should be available across multiple files.

+ In the case where program documentation refers to source language constructs, the linkage should be supported.

+ In the case where the project has supported "requirements tracing", the linkages among the requirements, design, and source documents should be supported.

E. Version Control

Part of the information associated with a large software development project relates to the various versions of it that have been produced. Sometimes a file in a new version completely replaces a file in the previous version. In other cases, conditional inclusion or line-by-line "deltas" are used to incrementally construct the new version from the old. In any case, it should be possible for the user to view any previous versions of a document and access commentary describing when, by whom, and for what reason a modification was made.

F. Annotation

There are several occasions when informal commentary can be associated with a node within a document. These may relate to changes between versions (e.g. "we should fix this if we get the time"), references to published articles, or alternatives that might be used if requirements change (portability or efficiency become more important). It is important that this information be recorded and be accessible to the maintenance programmer. It is also important that it not clutter the document unless explicitly requested. One possibility is to use a screen display mode, such as inverse video, to indicate the presence of an annotation and allow the maintenance personnel to view the commentary on demand. Of course, annotations about annotations should also be supported (e.g. "this was fixed in version 6.2").

G. Exploration Path Tracking

It is easy to imagine getting lost in this large information structure. The hypertext system must support mechanisms for reminding the user about the current point of view and how it was obtained. This is a concrete representation of the exploration path that lead to the viewpoint. This representation should be viewable by the user in summary form; for example, by a small "aerial view" in the corner of the screen. Moreover, the user should be able to save and restore such paths so that multiple explorations can be conducted.