1998 International Computer Symposium
Workshop on Software Engineering and Database Systems
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

AN INTEGRATED MEASUREMENT SYSTEM FOR DOCUMENTATION AND PROGRAMS

Timothy K. Shih, Wen C.Pai[+], Yule C. Lin, Lawrence Y. Deng, Chuan-Feng Chiu, and Wen-Hui Chang[*]

Dept. of Computer Science & Information Engineering, Tamkang University, Tamsui, Taipei 251, Taiwan
Dept. of Information Management, Kuang Wu Institute of Technology & Commerce, Taipei 112, Taiwan[+]
Chung-Shan Institute of Science & Technology, Lung-Tan, Taoyuan 325, Taiwan[*]

E-mail: tshih@cs.tku.edu.tw

ABSTRACT

Current program documentation has a number of drawbacks, such as the incompleteness, inconsistency, traceability problems, no quantitative methods to measure the quality, and unfriendly to read and write, which usually cannot achieve its goal successfully. In this paper, we propose an integrated measurement system for documentation and programs, which can force the documents and programs to be tightly coupled. No modification of programs can be proceeded outside this system, and all related documents must be updated along with a modified program. The completeness, consistency, and traceability of documentation can be assured. Multimedia can be used to enrich the expressiveness of documents. The statistical data about the documents, programs, and all activities of the team members are recorded into a database. Lots of valuable information can be acquired through various queries. Configuration management issues, especially the change control, can be achieved easily and conveniently via the Internet/Intranet.

Key Words: configuration management, documentation, software metrics, hypertext, multimedia.

1. INTRODUCTION

Software development and maintenance have long been regarded as some kind of sophisticated, unreliable, and costly work. Software experts have proposed many methods, such as structured programming, object-oriented programming, computer-aided software engineering (CASE), and re-engineering etc., to improve this situation. We believe poor documentation methodology and tools are one of the dominated factors. Program documentation is used to help maintenance or naïve programmers, sometimes even the original development programmers, (hereafter referred to as newcomers) to understand the program. Well-engineered program documentation can certainly increase the reliability of the software and the productivity of a software company, and reduce the time and cost of development and maintenance significantly. But current program documentation methods do have following drawbacks, and usually fails to achieve these goals.

- Documents are incomplete. Some parts are unwritten or lost.
- Documents and programs are inconsistent. Programmers often modified programs, but forget to or unwilling to update the corresponding documents.
- Traceability is lost. The correspondence between documents and programs is missed or confused.
- Only monotonous text and plain charts can be used to write the documents.
- Whether all requirements and designs have been done is not clear?
- Whether programmers update the program and corresponding documents simultaneously is also unclear.
- Whether maintenance programmers have read and understood all the necessary documents and programs cannot be measured quantitatively.
- Once the programs or documents are updated, there are no convenient ways to notify all the related project managers and programmers.

These drawbacks result in serious problems in project development and maintenance. Besides it make projects error-prone and unreliable. Software experts know that the maintenance of existing software can account for over 60 percent of all effort expended by a software company. And the percentage continues to rise as more software is produced [9]. Based on our years of project management and programming experience, we know current documentation methodology and Computer-Aided Software Engineering systems (CASE) cannot solve the-

se problems. We often take great pains during development and maintenance of large projects. Therefore we decide to develop an easy-to-use but effective project management system to overcome these problems.

This system can tightly integrate the documentation and the programs together. Programmers can no longer modify the programs separately. It uses the capability of a hypertext editor to edit the entire documentation and programs. Programmer cannot edit the program alone, all programs should be created and modified within our system. The interaction between documents and programs are bi-directional, reviewers can trace from bottom-level programs to top-level documents upwards, or downwards conversely. Our system then extracts all the programs, builds them into project files, compiles, and links them into an executable file. All associated documents also should be modified, the measurement database can easily detect which related documents are not updated along with the programs. With this system, the traceability problem, the incomplete problem and the inconsistent problem can be overcome effectively.

Besides our system allows programmers to use the richer multimedia to enhance the readability and expressiveness of our documentation. Multimedia data, such as video, sound, pictures, text, animation, and virtual reality (animation and virtual reality can be represented in video form), can be recorded as project documents. We know speaking is much easier and faster than writing, a picture is worth a thousand words, and video is certainly better than pictures. They can carry far more information than traditional text documents. Imagine a scene, when certain document or program segment confuses a newcomer, the original programmer appears in the video to explain why and how after a simple click. The newcomer can know not only the meaning of this program, but also many characters of this programmer.

From the viewpoint of software engineering, the manager or customer can know lots of important information in software quality control if they ask the project team to use sound or video to explain their programs. Such as programmers who ever join this project? How about their quality and working attitude? Are the code review meetings really held? Who ever attend these meetings? Are their reviews very serious? Does this project team really follow the steps of software development standards etc.? These are all missing in conventional written documents. Video and sound records are much harder to counterfeit.

Projects in this integrated system can easily and seamlessly transferred into the Internet or a company's Intranet. Project team members can easily cooperate together via Internet or Intranet. All activities are collected, analyzed, and put into the statistical database for further quantitative measurement. Such measurement serves as another kind of software metrics [6, 19]. The feature of multimedia documents will be especially useful in a high-speed Intranet. Video and sound discussions will be much better than traditional text documents.

Maintenance of a large project is undoubtedly a headache problem for all managers, because they know nothing about the maintenance programmers. Whether they have read all necessary documents and programs? Whether they have paid much attention in some critical issues? With this integrated and measurement system, project managers can easily grasp whether all necessary items (including documents and programs) are visited by maintenance programmers, whether critical issues are browsed patiently and carefully, and whether they have responded any questions and comments. Besides the maintenance programmers can send E-mails to the original development programmers and manager to ask, report, even complain something about the documentation or programs. If some document or program is updated, programmers responsible for modification can send E-mails to notify all the related people that ever visited this program about why and how they modify this program. We consider this system might be very helpful in software maintenance and configuration management.

This paper is organized as follows. Section 2 discusses how to integrate the documentation and code, and the benefits from this integration. Section 3 presents how to extract the programs from an integration set, and collect various data during the extraction process. A database management system is used to store these data and query them. In section 4, we address some problems of configuration management and maintenance of large projects, and how to use this integrated system to improve this situation. We also discuss previous researchers' related work and a couple of CASE-like systems in section 5. Section 6 concludes this work and discusses some interesting issues can be studied in the future.

2. DOCUMENTS AND PROGRAMS INTEGRATION

Our system utilize a hypertext editor to edit all the documents (including requirement design and testing

documents etc.) and programs. Although there are numerous hypertext editors around, such as the *Microsoft's FrontPage*, *Netscape's Composer*, or *Sausage Software's HotDog* etc., they cannot meet our specifications more or less. Therefore, we decide to design our proprietary hypertext editor. However this editor still adopts the HyperText Markup Language (HTML) format in order to be portable with other hypertext systems.

Fig. 1 shows how we use a hypertext editor to create project development documents and programs. The major requirement document hyperlinks to design documents, then hyperlinks to corresponding programs. If any documents, that is hyperlinks, are missed or lost, a simple hypertext utility can easily find out which are absent. This feature can facilitate project managers to control the incompleteness of documentation.

If a project adopts the bottom-up development methodology, many component programs will be coded first to test the feasibility of some bottleneck or technical problems. The project manager can put all verified components into the hypertext editor, then connect upwardly to corresponding documents. With this system, reviewers can read any interesting documents first, then trace upwards to the parent documents or downwards to the programs conveniently. Or they can watch the programs first, then trace the related documents bi-directionally.

Fig. 2 illustrates that our system can also use a data flow diagram (DFD) to describe the actual flows of input data. All external entities, processes, data objects, and data stores can also hyperlink to deeper level DFDs. Traditional software component block diagrams can also be represented as hypertext structure.

Due to the inherent capability of hypertext, the traceability problem can be easily tackled. Any non-traceable node (a document or a program) can be picked out easily. The inconsistency problem can also be handled elegantly. Any modified node will be recorded with a timestamp, our system can soon detect it and display a different color on all related hyperlinks. The inconsistent node will also be receded into the statistical database for further inspection during code extraction.

Because most commercial hypertext editors adopt the HTML format. Programs created by these kinds of editors cannot be fed into ordinary compilers directly. This is one of our design aims to prevent casual programmers to modify the programs outside the system. All programs reside in our system can only be compiled through the code extraction process.

3. CODE EXTRACTION

The integration of documents and programs forms a bi-directional graph (See Fig. 3), because more than one document might hyperlinks to a frequently used function, such as a sorting routine, or a window utility. Our system utilizes a graph search algorithm to visit each node, picks out the program nodes to form build files. These build files are temporary files and are invisible to programmers, which cannot be copied and edited outside the system. Our system will also collect and put related statistical data into a database during the graph traversal process.

3.1 Search and Extraction

We utilize the compiler-generator tools *lex* and *yacc* [1,13,15] to pick out the programs from this system. Our scanner has to tell whether a node is a document file or a program file.

If a program node is encountered, all HTML control words are skimmed to make it a standalone file, then fed into the compiler and linker to produce a executable file. The number of files might be more than that of ordinary projects, however this complexity is completely handled by our system and irrelevant to programmers. Currently this system can only process source files, that is included header files and program files, precompiled header files, object files and library routines are not permitted in this version.

If a document node is found, the connectivity to all hyperlinked files is checked first to ensure the completeness of the documentation. Then it will be analyzed and counted, all statistical data are put into a database. Fig. 4 illustrates a sample relation in the statistical database of a project using military documentation standards MIL-STD-498. Although these statistical data are rather preliminary, experienced software quality assurance specialists still can perceive some valuable information from this database.

3.2 Quantitative Analysis

Through this statistical database, we can know much information about this project. For instance, if a program node is modified, but its parent document nodes are not updated, then the consistency is destroyed. If a document node is updated, but its parent document nodes, or child program node is not updated,

the consistency corrupts too. Besides the consistency checks, this database can also answer following questions:

- The amount of multimedia used to annotate the entire project.
- The average of multimedia used to annotate each document.
- Programmers willing to write documents, and those unwilling.
- The media most programmers prefer, and the media some programmers never use.
- Documents with few multimedia data.
- The quantities of documents are above or lower than others.

Project managers can also perceive some hidden messages from this summary. For example, if the percentage of inconsistent program nodes is high, these program nodes might be modified frequently. Programmers are tired to update the corresponding documents every time. If excessive programmers are unwilling to write documents, it might indicate the project schedule is somewhat rush. Programs with few documents might reveal that the programmers know little about what they wrote. If few programmers use sound or video to annotate their programs, it might point out their office is unsuitable for recording (too small or noisy).

Based on this table, the manager can prepare a list for all programmers from the one with the most documents down to with the least, and reward those above the average, and encourage those below. Of course, programmers can examine these results before the manager. The primary purpose of this statistical database is to encourage programmers to write documents and programs more completely.

4. CONFIGURATION MANAGEMENT

Configuration management is the art of identifying, organizing, and controlling modifications to software. The goal is to maximize productivity by minimizing mistakes [2].

Changes are inevitable in the software life cycle, most project managers admits change control and maintenance are their heavy burden. According to our practical experience, configuration management is hard to be tackled well, which results in serious project delay, customers' complaints, and some dangerous bugs.

4.1 Management in the Internet/Intranet

Projects developed with our system can be put into the Internet or a company's Intranet smoothly, many configuration management tasks during development and maintenance can be achieved through lots of utility programs. Project team members can conveniently communicate one another via the network during every phase. Documents and programs can be browsed by anyone that connected the network. Multimedia data will be very helpful especially in the Intranet. Voice and video can truly reflect the realistic aspects of miscellaneous problems.

Team members can send multimedia E-mails to the authors of documents and programs. All comments and opinions are collected and browsed by related members. Our system will record whoever browsed which kind of documents and programs. If the documents or programs are updated, the members responsible for modification can send the update message E-mails to all the people that ever visit this node. Besides this, all activities in the network are collected and put into previous statistical database. Fig. 5 illustrates a relation recorded with the detailed statistical data of visiting members. Upon querying this database, project managers or customers can know lots about the interior issues of this project

Maintenance programmers can also use the editor to read the documentation and programs. All their browsing activities, suggestions, questions, and complaints are recorded. Project managers can use any kind of web site servers to monitor their behavior, and collect miscellaneous data.

4.2 Quantitative analysis

When newcomers report that they have "finished" reading the program, the project manager usually doesn't have any method to evaluate the status of their understanding, let alone do any quantitative analysis. And the manager often cannot obtain responses from them systematically. Our system provides a statistical database to record and analyze their learning status and responses. This database can report following information.

- Whoever browsed the documents and programs?
- How long does certain visitor stay at each document or program?
- The number of documents and programs have been traversed and the percentage.
- Total and average time expended in traversing documents and programs.

- Total and average quantities of multimedia responses (suggestions, complaints, and questions).

According to these queries, the manager can grasp the learning status and the efforts devote of a newcomer, and the quality of documents and programs. The manager can perceive somewhat about the quality of the project, and compare this result to other newcomers' or projects' responses to obtain a more accurate evaluation. Similarly, the manager can know which member performs best (who has most good comments), and which has least etc. The summary of suggestions, complaints, and questions reports not only the quality of the project, but also the effort the newcomer devotes. In comparison with other projects, the manager can realize the quality of documents, programs, development programmers, newcomers and the program itself in further detail.

We consider the most important meaning of this database is not to monitor the learning behavior of newcomers, but to understand the quality of documents and programs. If they are clear, complete and easy to understand, newcomers must be willing to traverse the entire program and documents thoroughly, and can finish the tour quickly with few complaints.

5. RELATED WORK

Hypertext is a special kind of active text, and has been widely used in computer on-line documents since the 80's [4,16]. There are numerous hypertext systems around. Due to hypertext's special characteristics, it has also been used in Computer-Aided Software Engineering (CASE) [3]. But conventional HyperCASE systems often don't provide statistical database to measure documents and programs quantitatively, most of all, they cannot prevent the casual programmers modify the programs outside the system. Besides most activities via the network are not collected and analyzed. Project managers often cannot acquire useful information about the project itself and the team members.

HTML is the most popular hypertext language now. There are many HTML editors and browsers, such as *Netscape Composer* and *Navigator*, *Microsoft FrontPage* and *Explorer*, and *Sausage Software HotDog* etc.

Multimedia presentation and authoring tools [7], have been used as a computer training tools for years, the effect is deeply impressed. Trainees can communicate with the system interactively, and their learning status can be measured quantitatively. The idea of the tour of a program comes from these multimedia systems.

Knuth noted that program documentation is incomplete in the early 80's, which leads to his pioneering work on *literate programming* [14]. His approach embeds documents into the program, which allows designing and describing a program from top to bottom according to the stepwise refinement methodology. This method can be represented as hypertext naturally, several researchers have implemented systems using hypertext for literate programming [17]. Knuth's method can also be applied to our system due to the hypertext nature, embedded documents can be represented not only with text, but richer multimedia. However, because our system is not designed for literate programming, the resulting literate programs cannot be compiled directly. We consider the literate programming approach still needs a quantitative method to measure the completeness of embedded documents.

The Naval Research Laboratory (NRL) has proposed a project entitled *Software Cost Reduction* (SCR) [10], which specifies the requirements of real-time, and safety-critical systems precisely. This project has a profound impact on follow-up systems about formal documentation. Recently, some researchers have designed a suite of automated tools for constructing and analyzing formal requirement specification [10]. This tool set acts as an unambiguous and concise front-end of software development. It can form a complete and useful CASE environment if these tools for requirements and designs can be combined with ours.

Parnas, Madey and Iglewski's work on precise documentation of well-structured programs [18] serves as another approach to program documentation. Their method can document a program extremely precisely, and most suitable for safety-critical systems. In fact, their method has been successfully applied to a project in Darling nuclear plant in Canada. If their work can equip with a tool to annotate the formal specifications and programs, it will help general programmers to accept this precise and elegant methodology.

ISO [12], DOD [5], and IEEE [11] have produced various software development standards. Our system can be regarded as a tool set of these standards. Its hypertext functionality can help to create documents, its multimedia capability can help improve the monotonous text with various pictures, charts, diagrams, sound, and video. Meetings, discussions and reviews can be recorded in more truth-

ful aspects. The automatic checking and analyzing utilities in our system can fulfill previous drawbacks of these standards.

6. CONCLUSIONS

The CPU performance and storage device capacity have made tremendous progress recently. Why don't we use them to improve the software documentation? Based on years of programming experience, we know programmers are unwilling to write documents. More convenient tools must be provided to facilitate them to write the documents and programs. The advantages of our system can be summarized as follows:

- It integrates the documentation and programs as a tightly coupled object. Team members can no longer modify the program separately. This feature can prevent the occurrence of incomplete and inconsistent documents.
- Multimedia can enrich the expressiveness of conventional monotonous documents. All discussions, meetings, seminars related to this project can be recorded with sound, or video, which can pass the ideas much more truthfully.
- Project managers can measure many aspects of a project quantitatively through the statistical database. They can know not only the status of documents and programs, but also the team members and newcomers.
- Configuration management will become more easy and convenient via various networks. All activities are collected, analyzed and put into the statistical database for further queries.
- Changes can be handled much more easily. All modifications to programs or documents will not only be recorded into the statistical database, but also notify all the people that ever visit them automatically.

Our future work will focus on how to make this system become a more complete CASE system. It will be much more convenient and useful if a compiler/linker environment can combine with us [8]. Any compiler/linker errors can directly jump to the program node, not the generated build files. We also hope to implement various software standards, such as ISO 12207 or MIL-STD 498, into a set of templates. If different project teams adopt different standards, they need only to change a new template. Besides many defense and aerospace projects are safety-critical, we hope to embed the precise documentation methodology into our system to ensure the reliability and robustness of projects developed and maintained with this system.

REFERENCE

[1] A. V. Aho, R. Sethi and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, Reading, MA, 1986.

[2] W. A. Babich, *Software Configuration Management*, Addison-Wesley, 1986.

[3] J. Bigelow, 'Hypertext and CASE', *IEEE Software*, Mar. 1988.

[4] J. Conklin, 'Hypertext: an introduction and survey', *IEEE Computer*, Sep. 1987.

[5] DOD, Software Development and Documentation, DOD MIL-STD-498, Dec. 1994.

[6] N. Fenton, 'Software measurement: A necessary scientific basis', *IEEE Trans. on Software Engineering*, Mar. 1994.

[7] A. Ginige and D. B. Lowe, 'Hypermedia authoring', *IEEE Multimedia*, May 1995.

[8] A. N. Habermann and D. Notkin, 'Gandalf : software development environments', *IEEE Trans. on Software Engineering*, Dec. 1986.

[9] M. Hanna, 'Maintenance burden begging for a remedy', *Datamation*, Apr. 1993.

[10] C. L. Heitmeyer, R. D. Jeffords and B. G. Labaw, 'Automated consistency checking of requirements specifications', *ACM Trans. on Software Engineering and Methodology*, Jul. 1996.

[11] IEEE Computer Society, *Software Engineering Standards*, IEEE, 1994.

[12] ISO and IEC, *Software Life Cycle Processes*, ISO/IEC 12207, Aug. 1995.

[13] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd ed., Prentice Hall, 1988.

[14] D. E. Knuth, 'Literate programming', *The Computer Journal*, Feb. 1984.

[15] J. R. Levine, T. Mason and D. Brown, *Lex & Yacc*, O'Reilly & Associates, Inc., 1992.

[16] J. Nielsen, 'The art of navigating through hypertext', *CACM*, ACM, Mar. 1990.

[17] K. Osterbye, 'Literate Smalltalk programming using hypertext', *IEEE Trans. on Software Engineering*, Feb. 1995.

[18] D. L. Parnas, J. Madey and M. Iglewski, 'Precise documentation of well-structured programs', *IEEE Trans. on Software Engineering*, Dec. 1994.

[19] J. M. Roche, 'Software metrics and measurement principles', *Software Engineering Notes*, ACM, Jan. 1997.

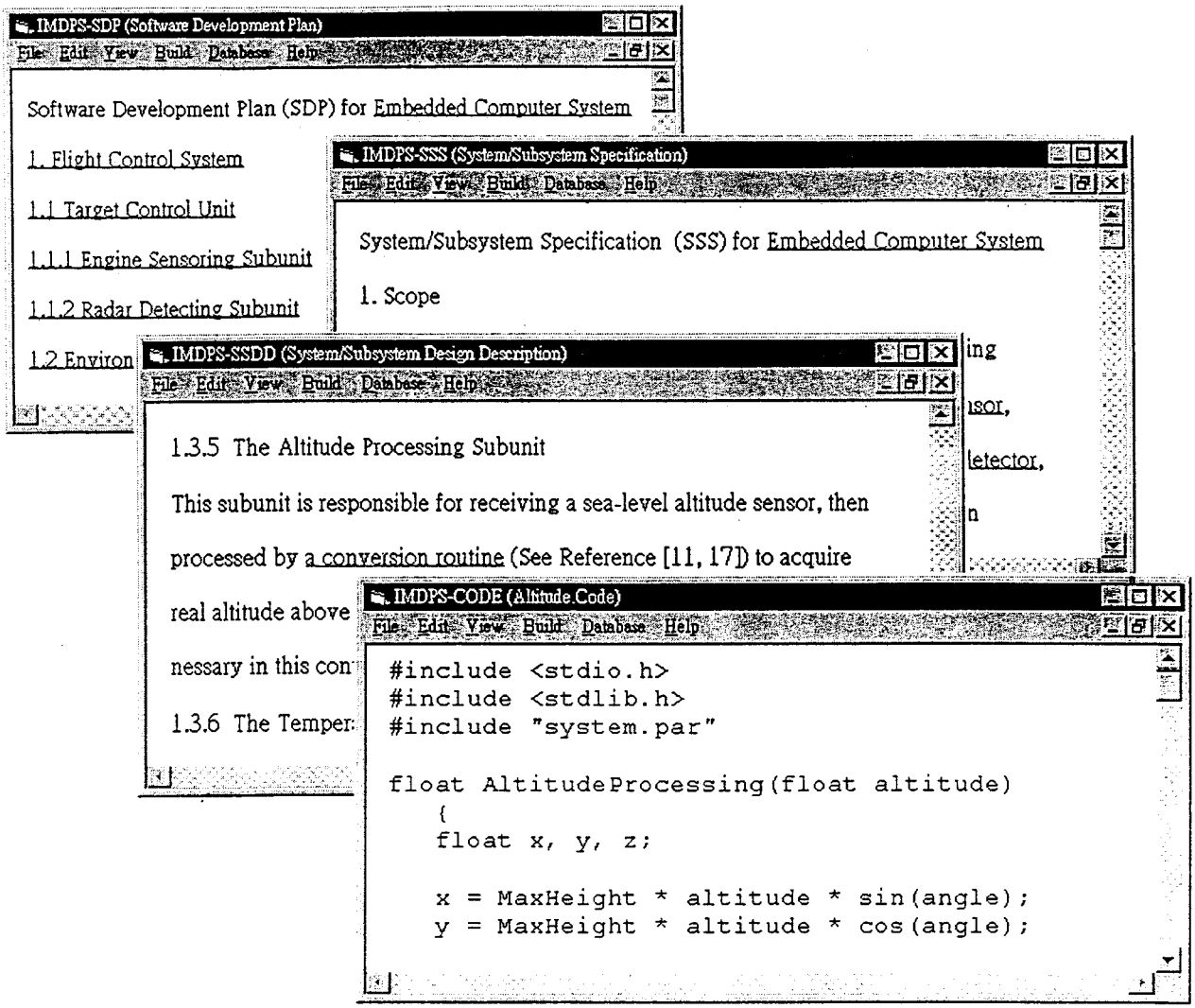

Fig. 1. Using a hypertext editor to create development documents and programs.

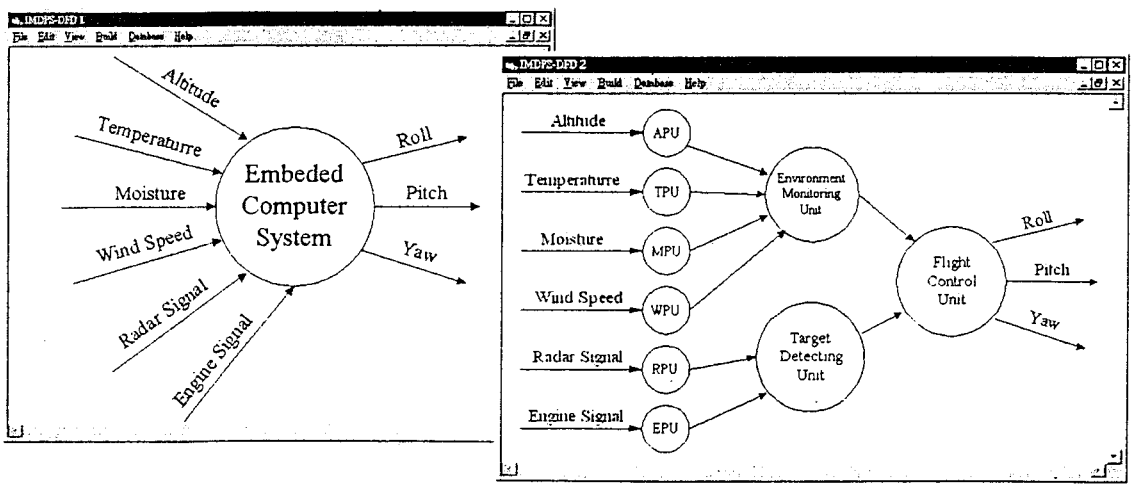

Fig. 2. Data flow diagrams (DFD) can also be represented as hypertext.

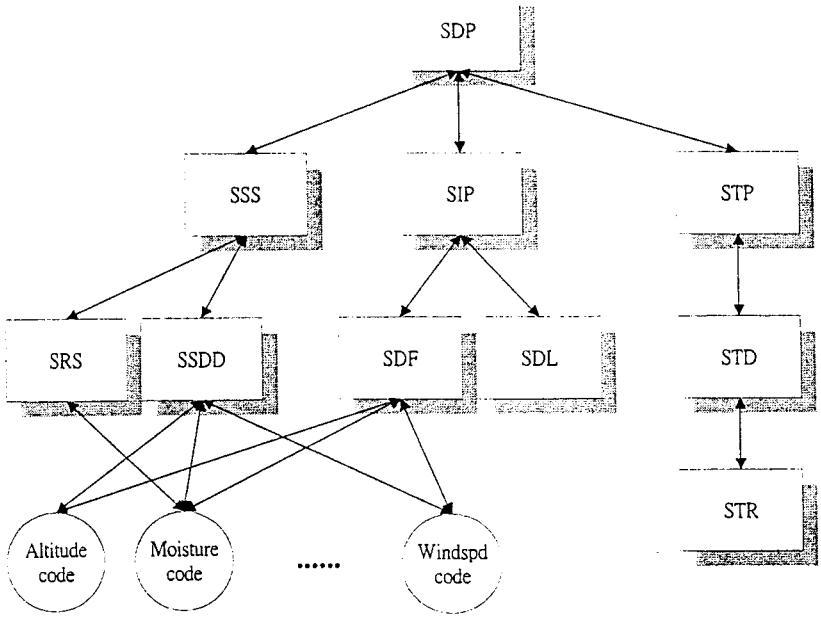

Fig. 3. The integration of documents and programs forms a bi-directional graph.

DID	Hyperlinks	Linked	Video	Sound	Charts	Text	Date	Time
SDP	38	35	01:10:17	10:26:18	38	31,937	11-Nov-97	09:03
SIP	17	10	00:18:23	07:00:26	11	7,321	22-Nov-97	11:57
STP	22	20	00:25:07	03:11:20	16	8,816	9-Dec-97	15:08
QCD	19	10	00:10:39	05:08:49	8	6,215	26-Dec-97	10:05
SSS	23	20	01:10:11	17:56:32	37	46,265	17-Jan-98	16:36
SSDD	35	27	03:08:05	18:10:29	59	58,498	29-Jan-98	10:27
SRS	41	29	02:37:25	11:08:51	67	81,654	9-Mar-98	11:08
...	...	...	...	...	...	...	...	...

Fig. 4. A relation in the statistical database for a project using MIL-STD-498.

E-mail	Visited	Sound	Text	Duration	Date	Time
yulelin@yahoo.com	SDP	00:17:08	1,932	01:02:57	12-Apr-98	09:00
yulelin@yahoo.com	SRS	00:38:25	2,312	00:58:32	17-Apr-98	11:18
yulelin@yahoo.com	SSS	00:25:39	1,817	01:57:36	28-Apr-98	17:30
yulelin@yahoo.com	SSDD	00:51:48	3,762	02:32:49	5-May-98	14:09
yulelin@yahoo.com	Altitude.code	00:04:16	213	00:35:21	11-May-98	10:05
yulelin@yahoo.com	Engine.code	00:06:52	317	00:42:55	12-May-98	08:58
yulelin@yahoo.com	Moisture.code	00:05:25	178	00:36:19	13-May-98	17:34
...	...	...	...	...	...	...

Fig. 5. A relation of visiting members' statistical data.