AIAA-95-1026-CP

ENHANCING REUSE IN SOFTWARE DEVELOPMENT

Young S. Cho
Doris L. Carver
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803

Abstract

Software reusability is a key factor in addressing the ongoing software crisis by improving software productivity and quality. We present a multi-faceted software reusability model, a Quintet Web. The Quintet Web consists of links of five types of reuse support information among existing document blocks: semantic, horizontal, hierarchical, syntactic, and alternative relationships. The five types of reuse support information are named the Semantic Web, the Horizontal Web, the Vertical Web, the Syntactic Web, and the Alternative Web. The Semantic Web defines the operational functionality of each software block. The Horizontal Web links functionally identical blocks of all phases. The Vertical Web identifies hierarchical relationships of software blocks. The Syntactic Web forms a chain from the declaration of each variable to its uses for all variables. The Alternative Web enables software developers to select an alternative algorithm for an identical functionality. The Quintet Web, which supports both software development and maintenance, is independent of a specific software development method.

I. Introduction

Software reusability is a key factor in addressing the ongoing software crisis by improving software productivity and quality.[2] Also, software reuse is the most economical method of creating new software.[5] Although software reuse activities have been relatively successful in several organizations,[1, 3, 4, 6] the promise of software reusability has not been fulfilled thoroughly. The difficulties of software reuse in the past are due partially to the fact that software was not developed to be reused. Software must be designed to be reused[7] in order to make future software reuse more efficient and more effective.

In this paper, we present a software reuse model. Primary characteristics of the model are:

1) operates independently from a software development method,
2) enables software developers to help assure the consistency of software being developed when it is applied to the development environment,
3) supports all phases of software development processes,
4) supports maintenance process,
5) supports adaptation of existing software to a new environment,
6) requires neither an intensive initial

investment nor an excessive ongoing effort,

7) enables software to be designed to be reused, and
8) allows search for software components at varying abstraction levels.

These characteristics are accepted features that are important for software reuse.

II. A Software Reusability Model - The Quintet Web

For a block of a software to be reused, five types of relationships of the block to its environment must be thoroughly analyzed:

1) The semantics of a block has to be identified to determine its functionality. The analysis of the semantic relationship in the context of a whole system identifies what the block performs in what situation,
2) A horizontal relationship of a block with blocks in other phases provides the means to reuse all phases,
3) A vertical relationship of a block in its operational environment gives the concept of the inter-relationship of the block in the system in which it operates,
4) A syntactic relationship of variables used in a block to the variables in the same block or out of the block supports the adaptation and the modification of the block to a new environment, and
5) An alternative relationship of a block enables choices of algorithms for the same operation.

All five distinct relationships must be interwoven to support integrated reuse. We name each of these five relationships a Semantic Web, a Horizontal Web, a Syntactic Web, a Vertical Web, and an Alternative Web, respectively. We call these five webs collectively a Quintet Web. Software is decomposed according to the operations at different levels at all phases. Each decomposed part is defined as a block and is assigned to keywords that completely describe the functionality of the block. Each type of relationship between blocks is defined as an individual Web. All five distinct relationships are interwoven to support integrated reuse.

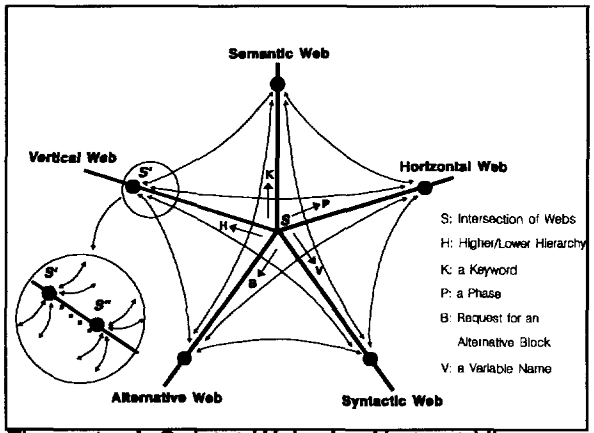


Figure 1. A Quintet Web: An Abstract View

The Quintet Web is conceptually illustrated in Figure 1. Thick radial lines represent the five Webs. The center of the radial lines, labeled *S*, is an intersection of the five Webs. A user of the Quintet Web begins at *S*. From *S*, he can expand his search to any direction. The five thick arrows pointing outward from the center *S* symbolize the search activities of reusable blocks by different strategies. Solid circles on each Web are arbitrary locations in each Web. A user can access any Web at any location in the Quintet Web through the links. So, if a user moves from *S* to *S'*, *S'* becomes the center of the Quintet Web. *S''* is another possible center of the Quintet Web.

The Semantic Web consists of links from a keyword to the blocks of user specification, requirement specification, pseudo-code, and code phases so that the operation of each block is described by the keyword. The Semantic Web may consist of several layers when the keyword can be decomposed to a set of keywords with narrower semantic meanings.

W_1	$\uparrow U_1$	$\uparrow R_1$	$\uparrow D_1$	$\uparrow P_1$	$\uparrow C_1$
W_2	$\uparrow U_2$	$\uparrow R_2$	$\uparrow D_2$	$\uparrow P_2$	$\uparrow C_2$
W_3	$\uparrow U_3$	$\uparrow R_3$	$\uparrow D_3$	$\uparrow P_3$	$\uparrow C_3$
W_4	$\uparrow U_4$	$\uparrow R_4$	$\uparrow D_4$	$\uparrow P_4$	$\uparrow C_4$
W_5	$\uparrow U_5$	$\uparrow R_5$	$\uparrow D_5$	$\uparrow P_5$	$\uparrow C_5$
W_6	$\uparrow U_6$	$\uparrow R_6$	$\uparrow D_6$	$\uparrow P_6$	$\uparrow C_6$

Figure 2 Semantic Web

Figure 2 is an example of the Semantic Web. W_1 through W_6 are keywords. U_i's, R_i's, D_i's, P_i's, and C_i's, are the ith block of user specification, requirement specification, design, pseudo-code, and code, respectively. Upright arrows(↑) are pointers to the blocks.

The Horizontal Web links blocks of software from different phases. If a block of one phase cannot be linked to its previous phase, we consider the previous phase incomplete. A modification or update is needed to complete the previous phase. When the Quintet Web is applied to the development process, this feature of the Horizontal Web enables software developers to help ensure the traceability in the software being developed.

Figure 3 is an example of a Horizontal Web. The boundary drawn by a thick curved line represents the complete set of documents in the design phase. Each phase (except the user specification and test data which are linked only to the

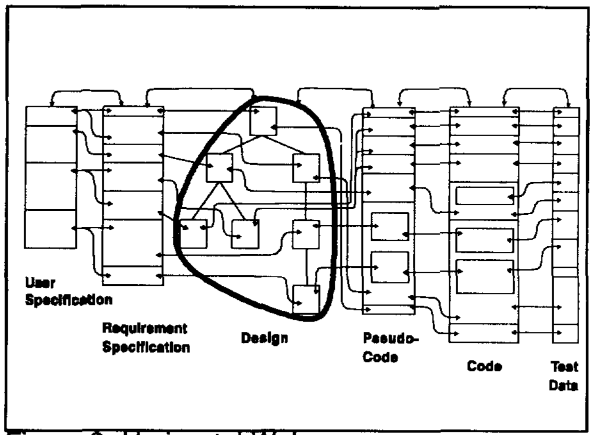


Figure 3 Horizontal Web

right or left, respectively) is connected by two links to its right and left adjacent phases. These links are the top level links. When a designer wishes to retrieve other phase documents of the corresponding phase in which he is currently investigating, the top level links are used. However, if the designer wants to retrieve blocks of other phase documents that correspond to the blocks he is dealing with, links that connect blocks, i.e., lower links, are used.

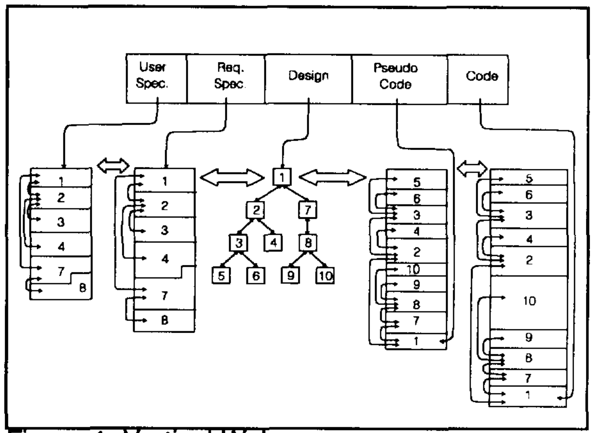
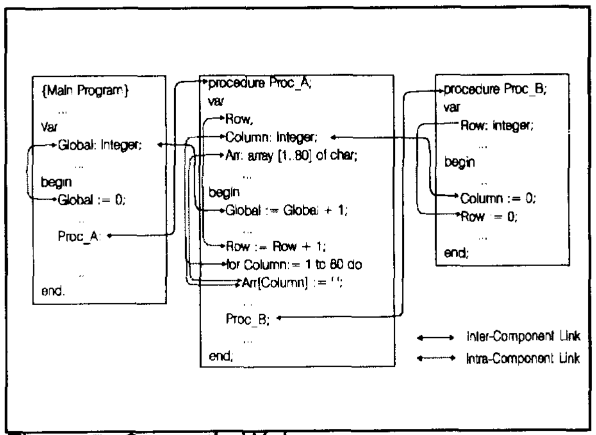


Figure 4 Vertical Web

The Vertical Web identifies the dependency of a block of software to another block. This Web applies not only for a design that involves hierarchical structure, but also for object-oriented design or other design methods where a dependency-like relationship exists, for example, caller-callee. Attributes of each block are inherited to its sub-blocks unless attributes of a sub-block are modified by its own attributes.

The diagram in Figure 4 is a Vertical Web. The array on the top consists of five pointers to corresponding documents: the first pointer points to the user specification, the second points to the requirement specification, the third points to the design, the fourth points to the pseudo-code, and the fifth pointer points to the code. More precisely, each pointer points to the highest block in the hierarchy of each document. This link is unidirectional. From/to the highest block from/to the next highest blocks, until the lowest blocks are linked, bidirectional links develop a complete Web of each document. These bidirectional links enable a software developer to investigate the hierarchical relationship between blocks easily. Note that blocks 5, 6, 9, and 10 in design, pseudo-code, and code are realized after the requirement specification phase.



```
{Main Program}
   ...
Var
   Global: Integer;
   ...
begin
   Global := 0;
   ...
   Proc_A;
   ...
end.
```

```
procedure Proc_A;
var
   Row,
   Column: Integer;
   Arr: array [1..80] of char;
   ...
begin
   Global := Global + 1;
   ...
   Row := Row + 1;
   for Column:= 1 to 80 do
      Arr[Column] := ' ';
   ...
   Proc_B;
   ...
end;
```

```
procedure Proc_B;
var
   Row: integer;
   ...
begin
   ...
   Column := 0;
   Row := 0;
   ...
end;
```

Figure 5 Syntactic Web

The Syntactic Web shows the syntactic chain of variables within and beyond a block. The concept of modification changes the perception of *a reusability system from a static library of* building blocks to a living system of components that spawn, change, and evolve new components with the changing requirements of their environment. [2] The Syntactic Web is helpful especially when a block of software is to be adapted to a new environment, or to be edited to comply with different documentation standards.

The Syntactic Web shows the syntactic chain of variables within and beyond a block. We call these two types of chains the intra-component link and the inter-component link. An intra-component link is a link for every variable that occurs in the block, starting from its declaration (or the first occurrence of each variable when implicit declaration is allowed and used) to every location it occurs. All the variables in a block can be traced through the intra-component link. An inter-component link is a link of every variable whose scope is beyond the scope of the block in which it resides, starting from the declaration of the variable to the places it is used. The use of global variables or variables whose scope is beyond the block can be traced through the inter-component link. Figure 5 is a simple example of a Syntactic Web. Dashed arrows represent the intra-component links and solid arrows represent the inter-component links.

The Alternative Web is links of a block to a block of another system(s) if the blocks perform the identical operation. If there is more than one block that performs the same operation, but in different ways, then all the blocks are linked together to provide a user with a choice of methods to attain the same operation.

The Alternative Web is shown in Figure 6.. There are two independent systems, *S* and *T*, and an independent block in the reusability system. Solid arrows connect blocks in each system as alternatives of each other. So, they are called bridge alternatives of each system. However, dashed arrows connect blocks in a system to the blocks which are not a part of any system, i.e., they are independent blocks. The purpose of the independent blocks is to provide alternatives to other blocks. They are called orphan alternatives. The dotted arrows are links that connect blocks and their attribute tags, or blocks to other blocks. When a software developer investigates an alternative of the block found, the information included in the attribute tags of all candidate blocks is given to the software developer in order to provide the description of each alternative. This link is also bidirectional to provide a mechanism to locate the actual document that is chosen.

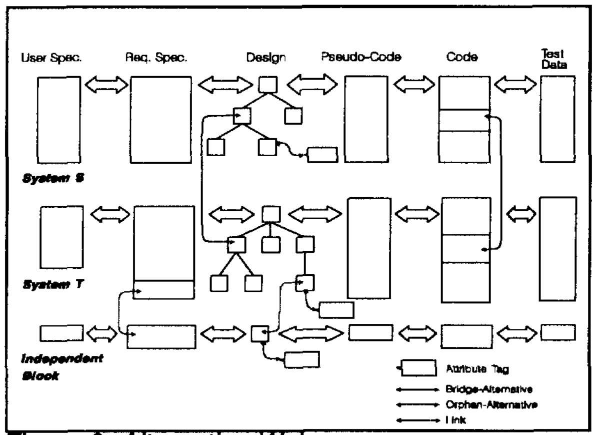


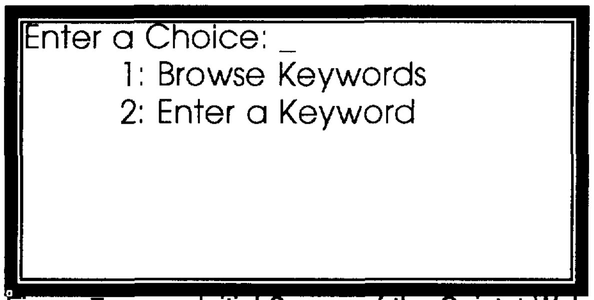
Figure 6 Alternative Web

III. Implementation of the Quintet Web

The Quintet Web has been implemented using the object-oriented paradigm because the characteristics of the Quintet Web are basically hierarchical.

The Quintet Web does not require that users have knowledge about each Web. Instead, a menu driven method is applied. The first menu prompts users to input (or browse) a keyword to search. A list of attributes of each candidate software component is provided before a user examines software. Choices of phases are given to the user. During the examination of the software components, users can change Webs by selecting a choice. If a user wishes to retrieve a software component, the component is copied to files with its other phase documents.

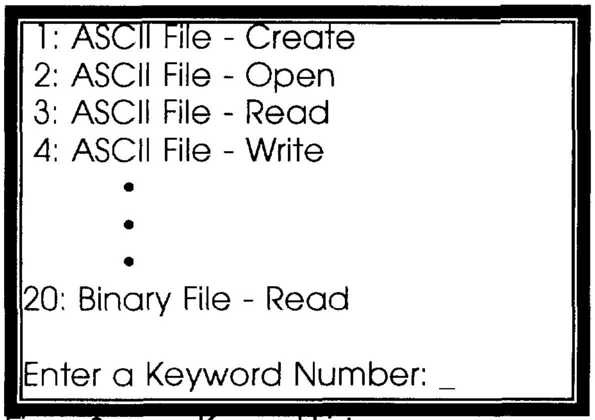
We demonstrate the execution of the prototype of the Quintet Web by introducing the sequence of screens. In implementation, the five webs in the Quintet Web are hidden to users. In other words, users are not required to have knowledge about the internal implementations of the Quintet Web. Each screen is shown in simplified form.



```
Enter a Choice: _
        1: Browse Keywords
        2: Enter a Keyword
```

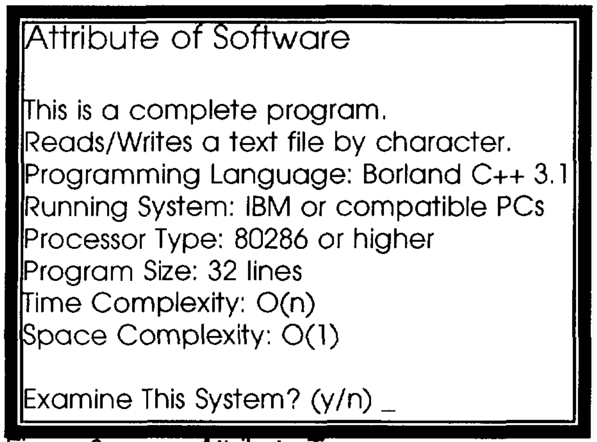
Figure 7 Initial Screen of the Quintet Web Prototype

The initial screen is shown in Figure 7. If a user chooses 1, a keyword list is given to the user. A sample keyword list is shown in Figure 8. By entering a keyword number, a software component is selected and its attribute tag is shown. An example attribute tag is shown in Figure 9. If the attributes are appropriate for the software to be developed, a user answers yes('y') to the prompt. Then, the choice of the phase to be browsed is prompted. This screen is shown in Figure 10.



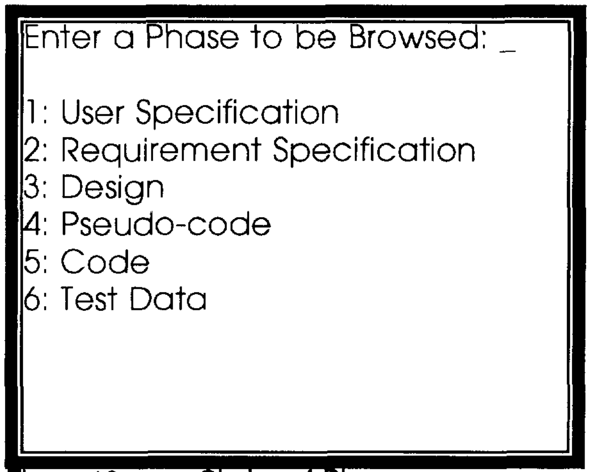
```
1: ASCII File - Create
2: ASCII File - Open
3: ASCII File - Read
4: ASCII File - Write
        •
        •
        •
20: Binary File - Read

Enter a Keyword Number: _
```

Figure 8 Keyword List



```
Attribute of Software

This is a complete program.
Reads/Writes a text file by character.
Programming Language: Borland C++ 3.1
Running System: IBM or compatible PCs
Processor Type: 80286 or higher
Program Size: 32 lines
Time Complexity: O(n)
Space Complexity: O(1)

Examine This System? (y/n) _
```

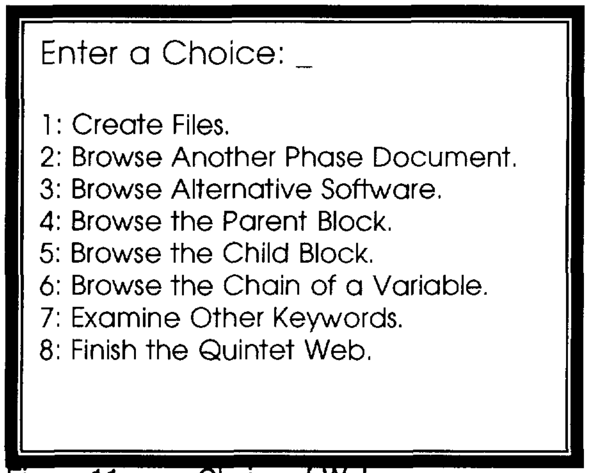
Figure 9 Attribute Tag



```
Enter a Phase to be Browsed: _

1: User Specification
2: Requirement Specification
3: Design
4: Pseudo-code
5: Code
6: Test Data
```

Figure 10 Choice of Phases

So far, a user has selected a keyword, and examined the attributes of a software component through the Semantic Web. By selecting a phase from the screen shown in Figure 10, the Horizontal Web is invoked. By each choice, a block of the appropriate document is shown on the screen. The actual screen is omitted because it is simply a copy of the actual document. The next screen, shown in Figure 11, guides the user in the selection of the web to choose for browsing.



```
Enter a Choice: _

1: Create Files.
2: Browse Another Phase Document.
3: Browse Alternative Software.
4: Browse the Parent Block.
5: Browse the Child Block.
6: Browse the Chain of a Variable.
7: Examine Other Keywords.
8: Finish the Quintet Web.
```

Figure 11 Choice of Webs

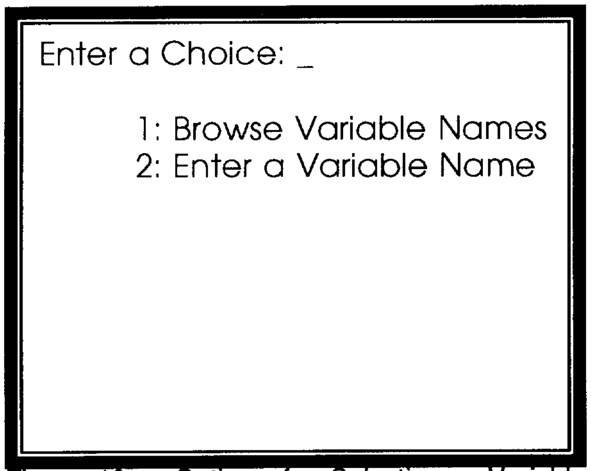
By selecting 1, a copy of a software block is created for all phases as text files. When these files are created, special codes are eliminated so that they can be used as they are. A user can specify file names; otherwise, default file names are used. The screen to input the file name is not shown here because it is a simple list of default file names for each phase and prompt to the user a alternate file names.

Choice 2 invokes the Horizontal Web again. The screen shown in Figure 10 will appear and process explained above is repeated.

Choice 3 invokes the Alternative Web. The alternative software is browsed on the screen. In this process, the phase a user selected on the screen shown in Figure 10 is used as the default phase. After an alternative software is examined, the screen shown in Figure 11 appears again.

Choices 4 and 5 invoke the Vertical Web. Callers/callees, or superclasses/subclasses of the current software block are browsed. The phase a user selected in Figure 10 is used as the default phase. If no callers/callees or superclasses/subclasses are available, appropriate messages are issued and goes back to the screen in Figure 11.

Choice 6 invokes the Syntactic Web. Users are prompted to enter or select a variable name. This screen is shown in Figure 12. Figure 13 is the screen for a variable name list. By either method, a variable is selected. The chain of a variable *fp* is shown in Figure 14. In our implementation, each statement where the selected variable occurs is marked in order to show the location of the variable.



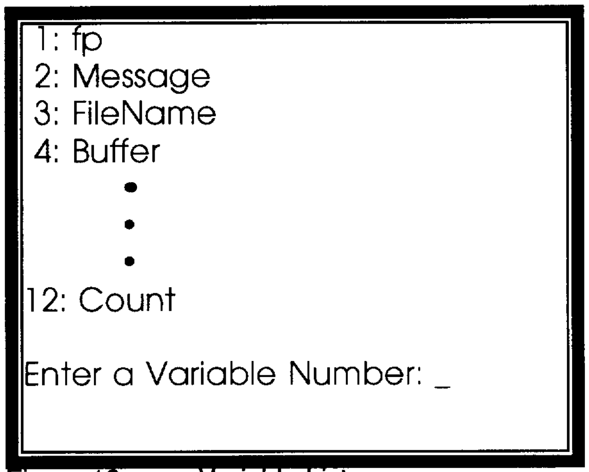
```
Enter a Choice: _

        1: Browse Variable Names
        2: Enter a Variable Name
```

Figure 12 Options for Selecting a Variable Name

Returning to the initial screen in Figure 7, choice 7 is to be selected. A search for other software components starts again.

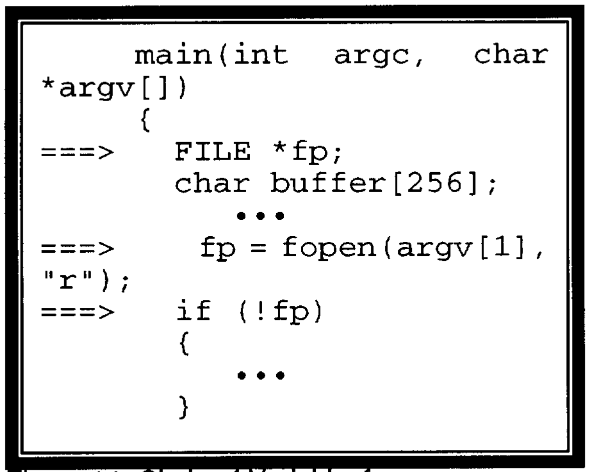
Choice 8 ends the Quintet Web.

Two types of information are required in the implementation: global information and information local to each Web. Global information includes attributes of each software, keywords list, and software system lists. Local information consists of links of blocks (or variables for the Syntactic Web) in each Web.



```
1: fp
2: Message
3: FileName
4: Buffer
      •
      •
      •
12: Count

Enter a Variable Number: _
```

Figure 13 Variable List



```
      main(int  argc,  char
*argv[])
      {
===>    FILE *fp;
        char buffer[256];
            •••
===>     fp = fopen(argv[1],
"r");
===>    if (!fp)
        {
            •••
        }
```

Figure 14 Chain of Variable *fp*

IV. Summary

A software reusability model, the Quintet Web, has been presented with discussion of each of the five Webs. The Quintet Web can be applied to the software development processe as well as to software reuse activities. This work represents a model that exemplifies the characteristics as stated in Section I. The implementation provides support that the Quintet Web model is feasible and can be used to support traceability during software development and reuse.

References

1. Banker, Rajiv D., Kauffman, Robert J., and Zweig, Dani, "Repository Evaluation of Software Reuse," *IEEE Transactions on Software Engineering*, Vol. 19, No. 4, April 1993, pp.379-389

2. Biggerstaff, Ted J., and Richter, Charles, "Reusability Framework, Assessment, and Directions," *IEEE Software*, March 1987, pp.41-49

3. Incorvaia, Angelo J., Davis, Alan M., and Fairley, Richard E., "Case Studies in Software Reuse," *Proceedings: The Fourteenth Annual International Computer Software and Applications Conference*, 1990, pp.301-306

4. Isoda, Sadahiro, "Experience Report on Software Reuse Project: Its Structure, Activities, and Statistical Result," *Proceedings: 14th International Conference on Software Engineering*, 1992, pp.320-326

5. Jones, Gregory W., *Software Engineering*, John Wiley & Sons, New York, New York, 1990

6. Prieto-Díaz, Rubén, "Making Software Reuse Work: An Implementation Model," *ACM SIGSOFT Software Engineering Notes*, Vol.16, No.3, July 1991, pp.61-68

7. Tracz, Will, "Where Does Reuse Start?," *ACM SIGSOFT Software Engineering Notes*, Vol.15, No.2, April 1990, pp.42-46