



US 20080065512A1

(19) **United States**

(12) **Patent Application Publication**
Schumacher et al.

(10) **Pub. No.: US 2008/0065512 A1**
(43) **Pub. Date: Mar. 13, 2008**

(54) **STRUCTURED DOCUMENT BROWSER**

08/649,271, filed on May 17, 1996, now Pat. No. 5,933,841.

(76) Inventors: **Robert M. Schumacher**, Wheaton, IL (US); **James E. Matthews**, Chicago, IL (US)

Publication Classification

Correspondence Address:
BRINKS HOFER GILSON & LIONE
P.O. BOX 10395
CHICAGO, IL 60610 (US)

(51) **Int. Cl.**
G06Q 30/00 (2006.01)
G06F 17/30 (2006.01)
G06F 3/048 (2006.01)
(52) **U.S. Cl.** **705/27; 715/810**

(21) Appl. No.: **11/710,255**

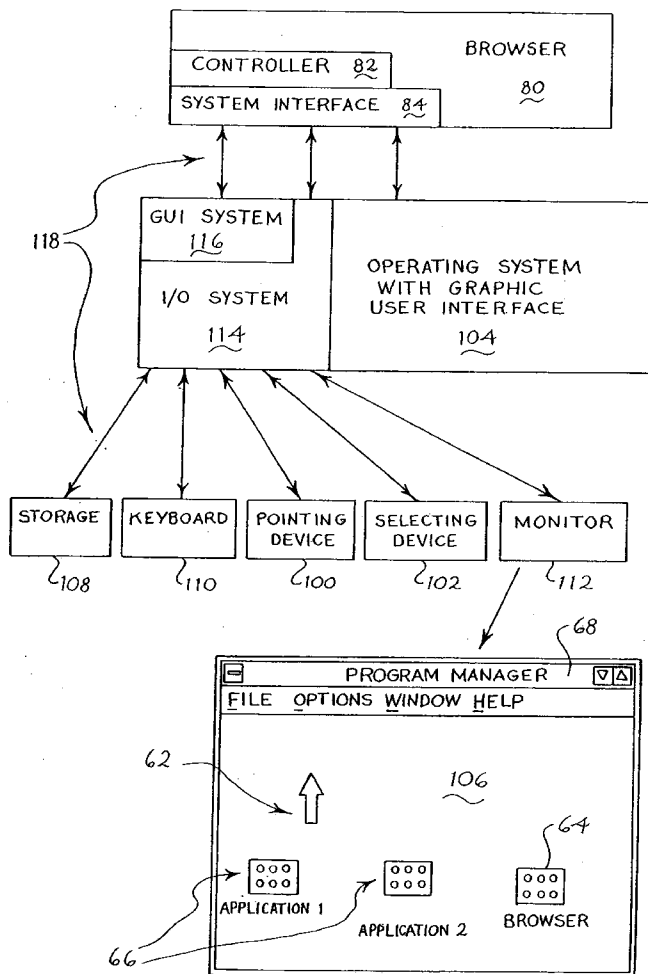
(57) **ABSTRACT**

(22) Filed: **Feb. 23, 2007**

A structured document browser includes a constant user interface for displaying and viewing sections of a document that is organized according to a pre-defined structure. The structured document browser displays documents that have been marked with embedded codes that specify the structure of the document. The tags are mapped to correspond to a set of icons. When the icon is selected while browsing a document, the browser will display the section of the structure corresponding to the icon selected, while preserving the constant user interface.

Related U.S. Application Data

(63) Continuation of application No. 10/773,370, filed on Feb. 6, 2004, which is a continuation of application No. 10/214,442, filed on Aug. 7, 2002, now abandoned, which is a continuation of application No. 09/304,425, filed on Apr. 29, 1999, now Pat. No. 6,442,574, which is a continuation of application No.



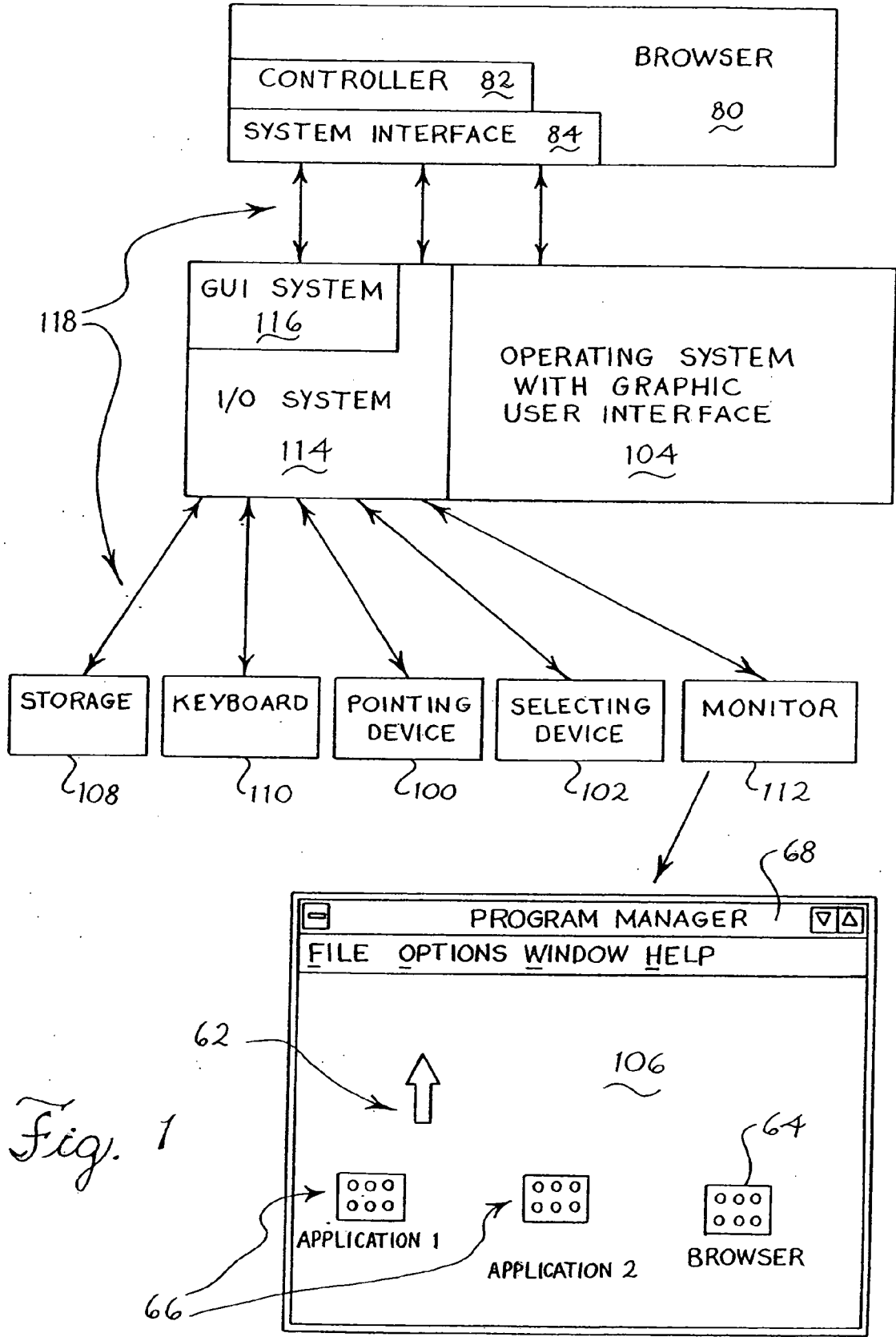
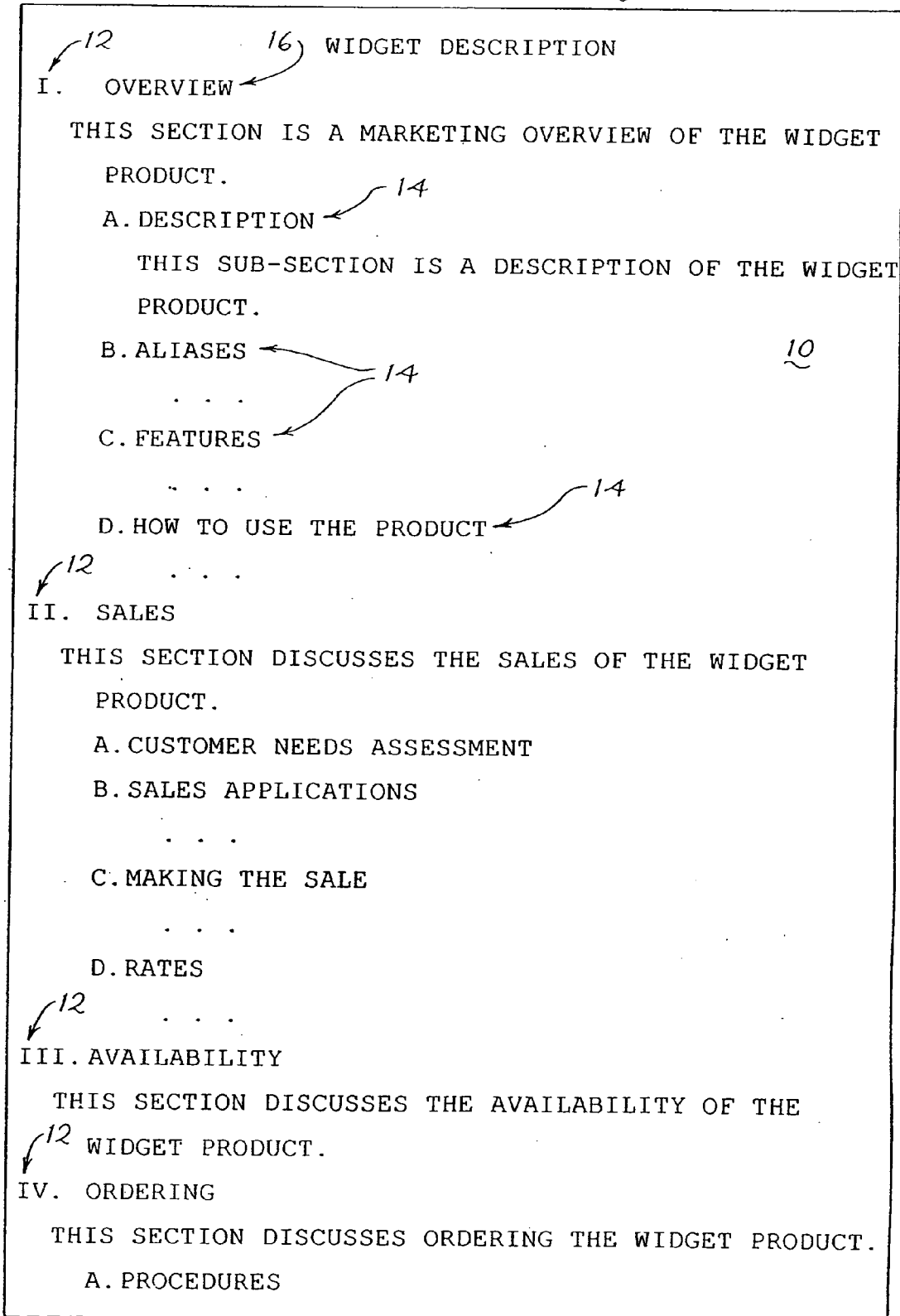


Fig. 1

Fig. 2(A)



TO FIG. 2(B)

Fig. 2(B)

-----TO FIG. 2(A)-----

. . . .
B. LISTINGS

. . . .
C. DUE DATES

12 ↙
V. BILLING

THIS SECTION DISCUSSES BILLING FOR THE WIDGET PRODUCT.

A. CHARGES

. . . .
B. BILLING REDUCTIONS

. . . .
C. PAYMENT OPTIONS/ CONTRACTS

12 ↙ ↘ *18*
VI. TROUBLESHOOTING

THIS SECTION DISCUSSES TROUBLESHOOTING WIDGET PRODUCT PROBLEMS.

A. KNOWN PROBLEMS

. . . .
B. PROBLEM ASSESSMENT

. . . .
C. DIAGNOSING CUSTOMER TROUBLE

12 ↙
VII. SUPPORT

THIS SECTION DISCUSSES WIDGET PRODUCT SUPPORT.

A. PRODUCT CONTACTS & REFERRALS

. . . .
B. RESPONSIBILITIES

-----TO FIG. 2(C)-----

Fig. 2(c)

TO FIG. 2 (B)

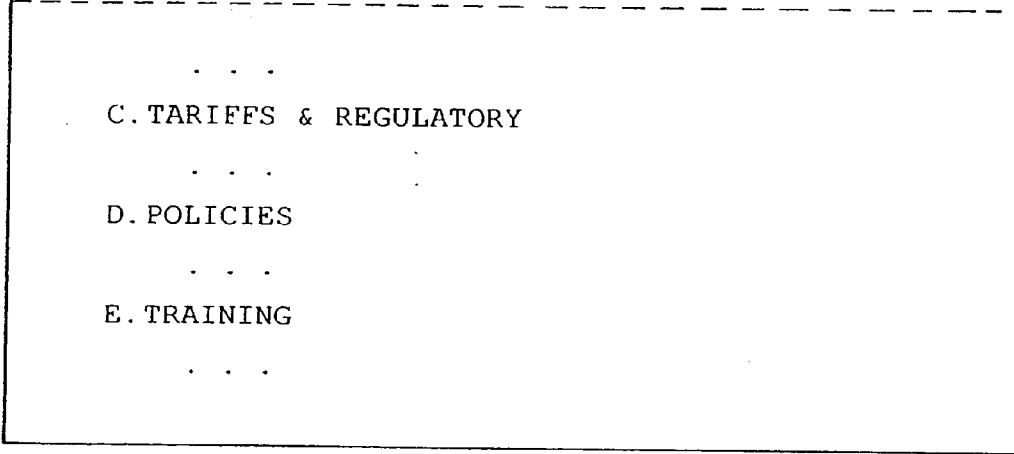
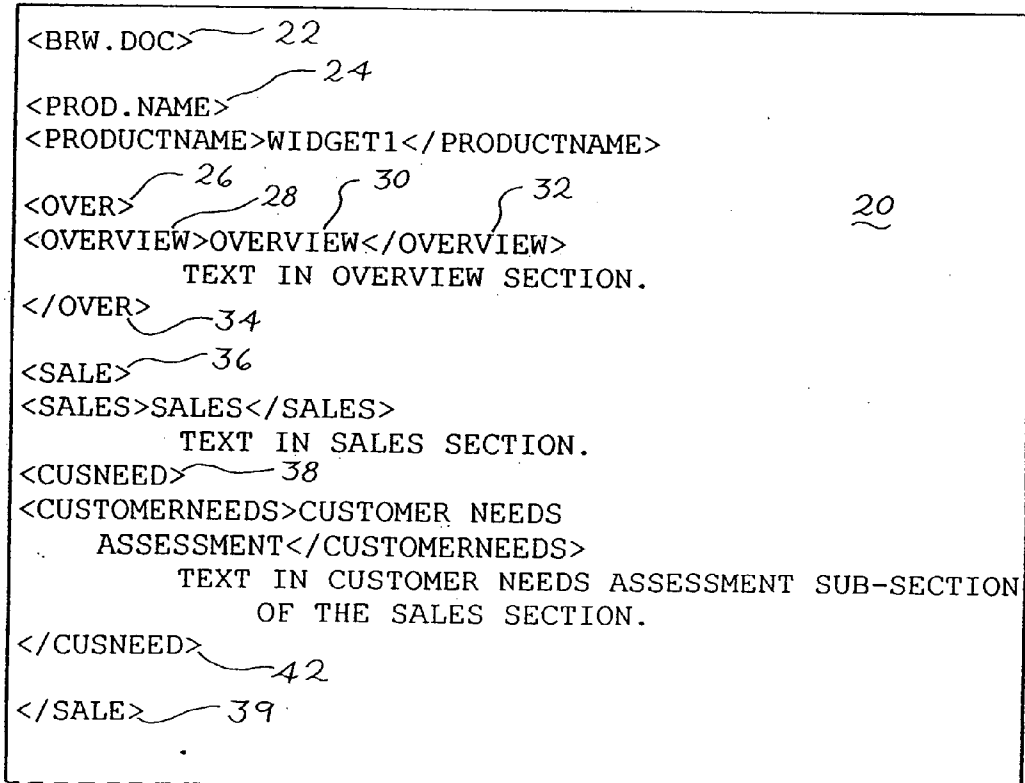


Fig. 3(A)



TO FIG. 3 (B)

Fig. 3(B)

TO FIG. 3(A)

```
<TROUBL> 25  
<TROUBLESHOOTING>TROUBLESHOOTING</TROUBLESHOOTING>  
27 TEXT IN TROUBLESHOOTING SECTION.  
<TRBL.KNOWPROB>  
<TRBLKNOWPROB>KNOWN PROBLEMS>/TRBLKNOWPROB>  
TEXT IN THE KNOWN PROBLEMS SUB-SECTION OF THE  
TROUBLESHOOTING SECTION.  
</TRBL.KNOWPROB>  
  
<TRBL.PROBASSESS>  
<TRBLPROBASSESS>PROBLEM ASSESSMENT</TRBLPROBASSESS>  
TEXT IN THE PROBLEM ASSESSMENT SUB-SECTION OF  
THE TROUBLESHOOTING SECTION.  
</TRBL.PROBASSESS>  
  
<TRBL.DIAG>  
<TRBLDIAG>DIAGNOSING CUSTOMER TROUBLE</TRBLDIAG>  
TEXT IN THE DIAGNOSING CUSTOMER TROUBLE SUB-  
SECTION OF THE TROUBLESHOOTING SECTION.  
</TRBL.DIAG>  
  
</PROD.NAME>  
  
</BRW.DOC>
```

Fig. 4(A)

```

<!ELEMENT PROD.NAME - - ( PRODUCTNAME, (%SUB-SECT; |
TROUBL |
ORDER |
AVAIL |
SALE |
BILL |
SUPP |
OVER)*) >
<!ATTLIST PROD.NAME
  ID CDATA #IMPLIED >
<!ELEMENT PRODUCTNAME - - ( (#PCDATA | %PARA-
CONTENT;)* ) >
<!ELEMENT TROUBL - - ( TROUBLESHOOTING, (%SUB-SECT; |
  TRBL.KNOWPROB |
  TRBL.PROBASSESS |
  TRBL.DIAG |
  TRBL.QUES |
  HEAD3 |
  HEAD2 |
  HEAD5 |
  HEAD4 |
  HEAD7 |
  HEAD6)*) >
<!ATTLIST TROUBL
  ID CDATA #IMPLIED
  ID CDATA #IMPLIED >
<!ELEMENT TROUBLESHOOTING - - ( (#PCDATA | %PARA-
CONTENT;)* ) >

```

42

46

47

49

44 45

40

48

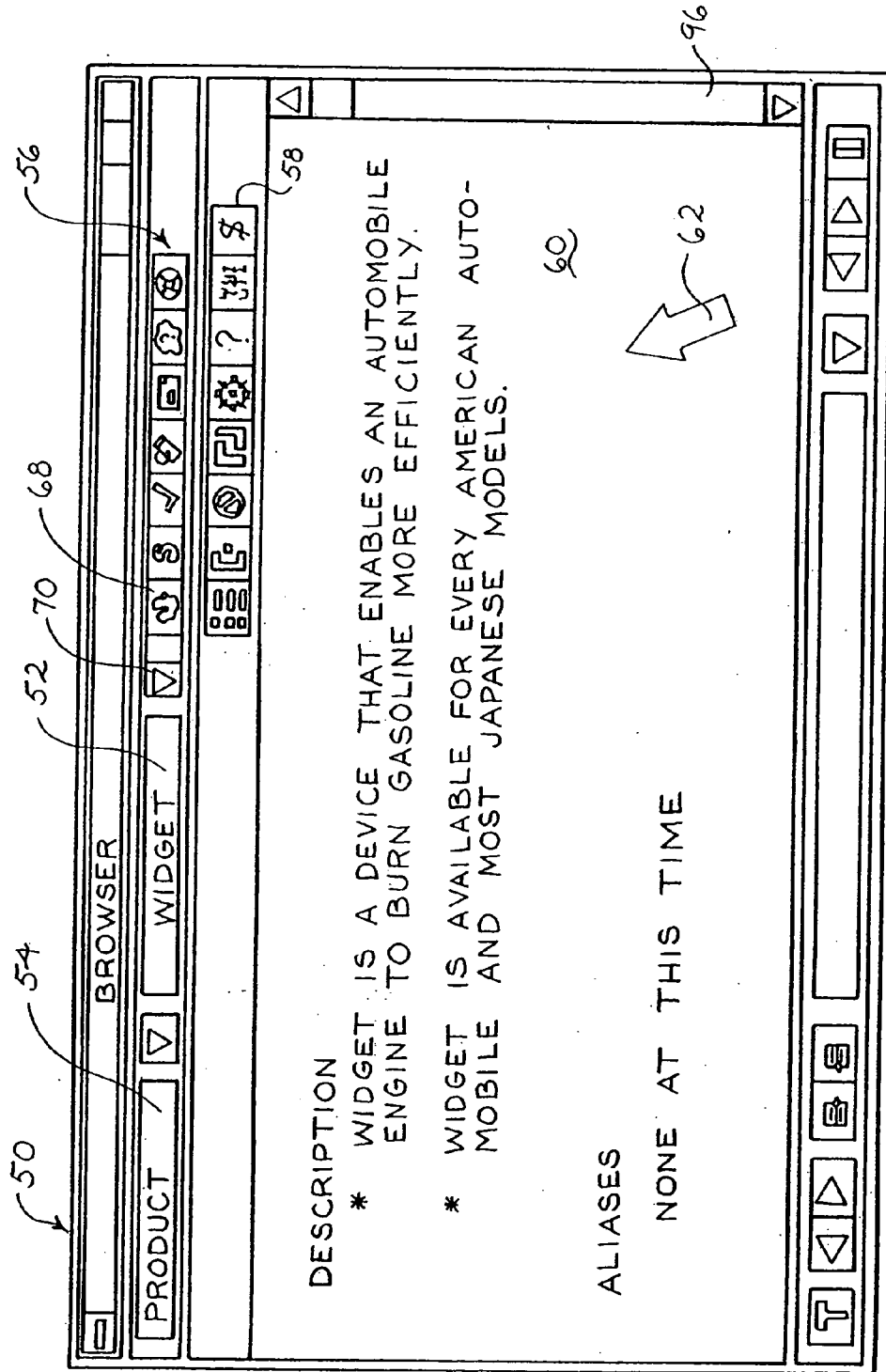
51

TO FIG. 4(B)

Fig. 4(B)

TO FIG. 4(A)

```
<!ELEMENT TRBL.KNOWPROB - - ( TRBLKNOWPROB, (%SUB-  
SECT; |  
HEAD3 |  
43 HEAD5 |  
HEAD4 |  
HEAD7 |  
HEAD6)*) >  
  
<!ATTLIST TRBL.KNOWPROB  
ID CDATA #IMPLIED  
TYPE CDATA #IMPLIED >  
  
<!ELEMENT TRBLKNOWPROB - - ( (#PCDATA | %PARA-  
CONTENT;)* ) >  
  
<!ELEMENT TRBL.PROBASSESS - - ( TRBLPROBASSESS,  
(%SUB-SECT; |  
43 HEAD3 |  
HEAD5 |  
HEAD4 |  
HEAD7 |  
HEAD6)*) >  
  
<!ATTLIST TRBL.PROBASSESS  
ID CDATA #IMPLIED  
TYPE CDATA #IMPLIED >  
  
<!ELEMENT TRBLPROBASSESS - - ( (#PCDATA | %PARA-  
CONTENT;)* ) >
```

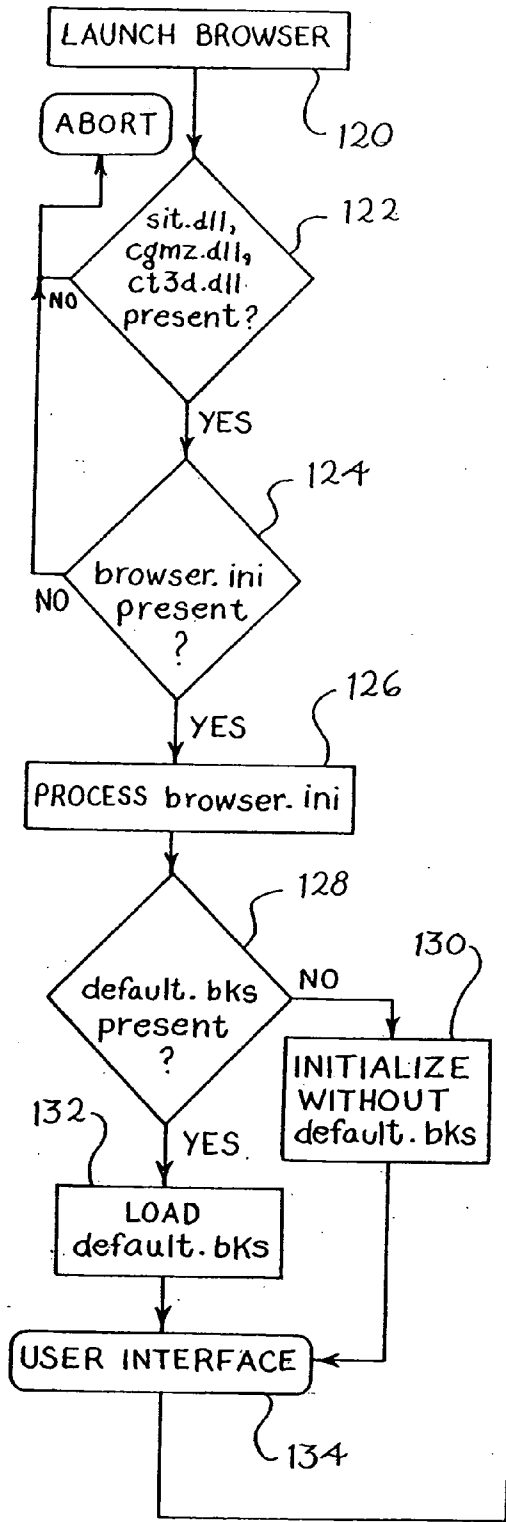
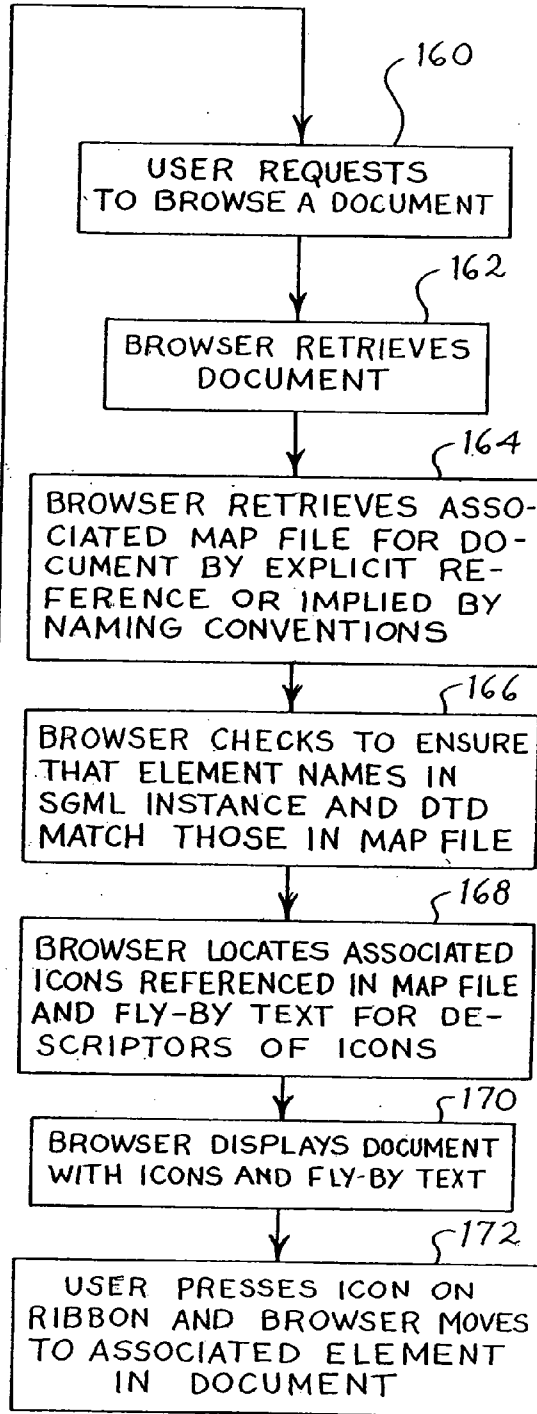


Fig. 6



[BOOK] 140
144 COLLECTION=\XYZ
146 COLLECTIONTITLE+"PRODUCTS"
148 BOOKNAME=WIDGET
150 MAP=PRD.MAP 152

Fig. 7

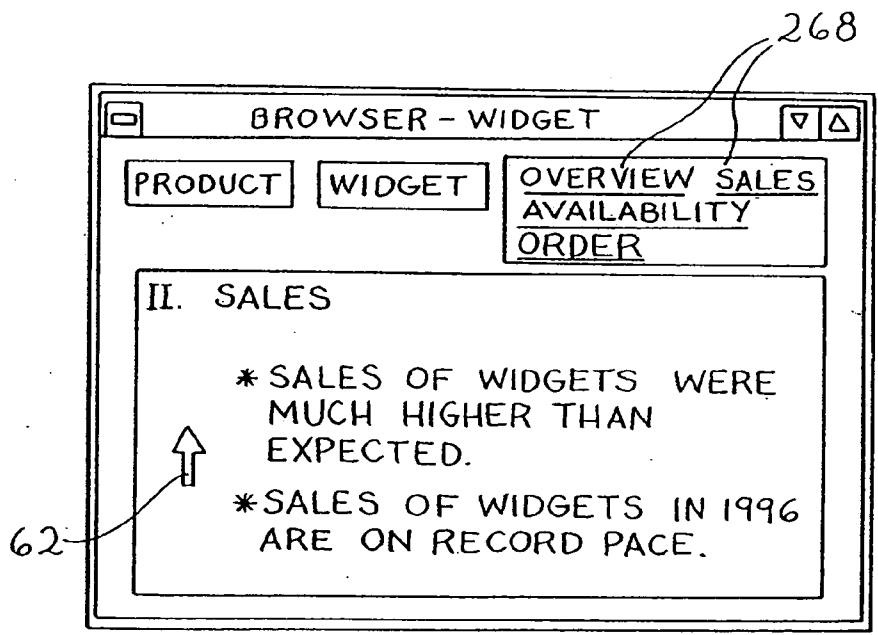


Fig. 10

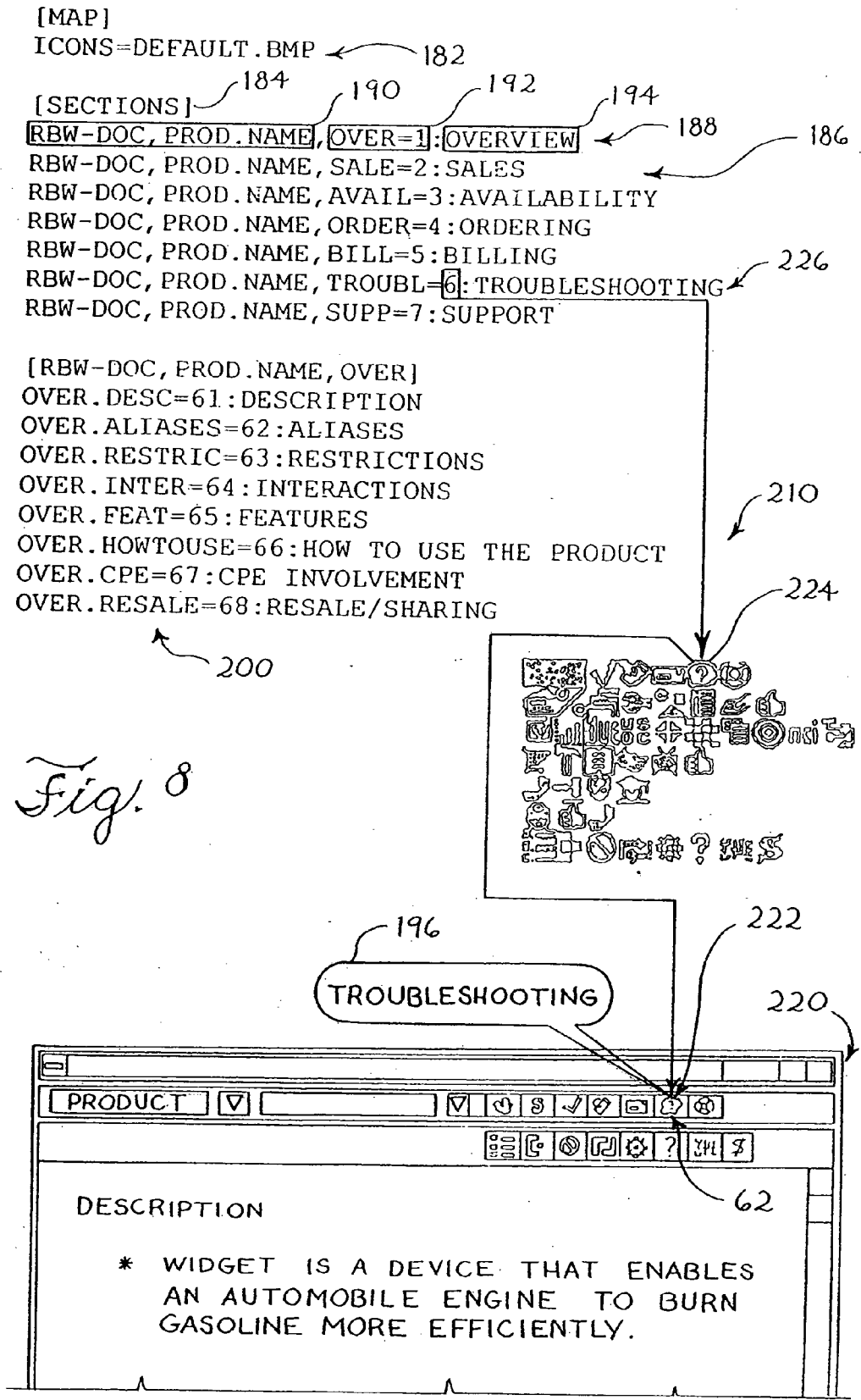
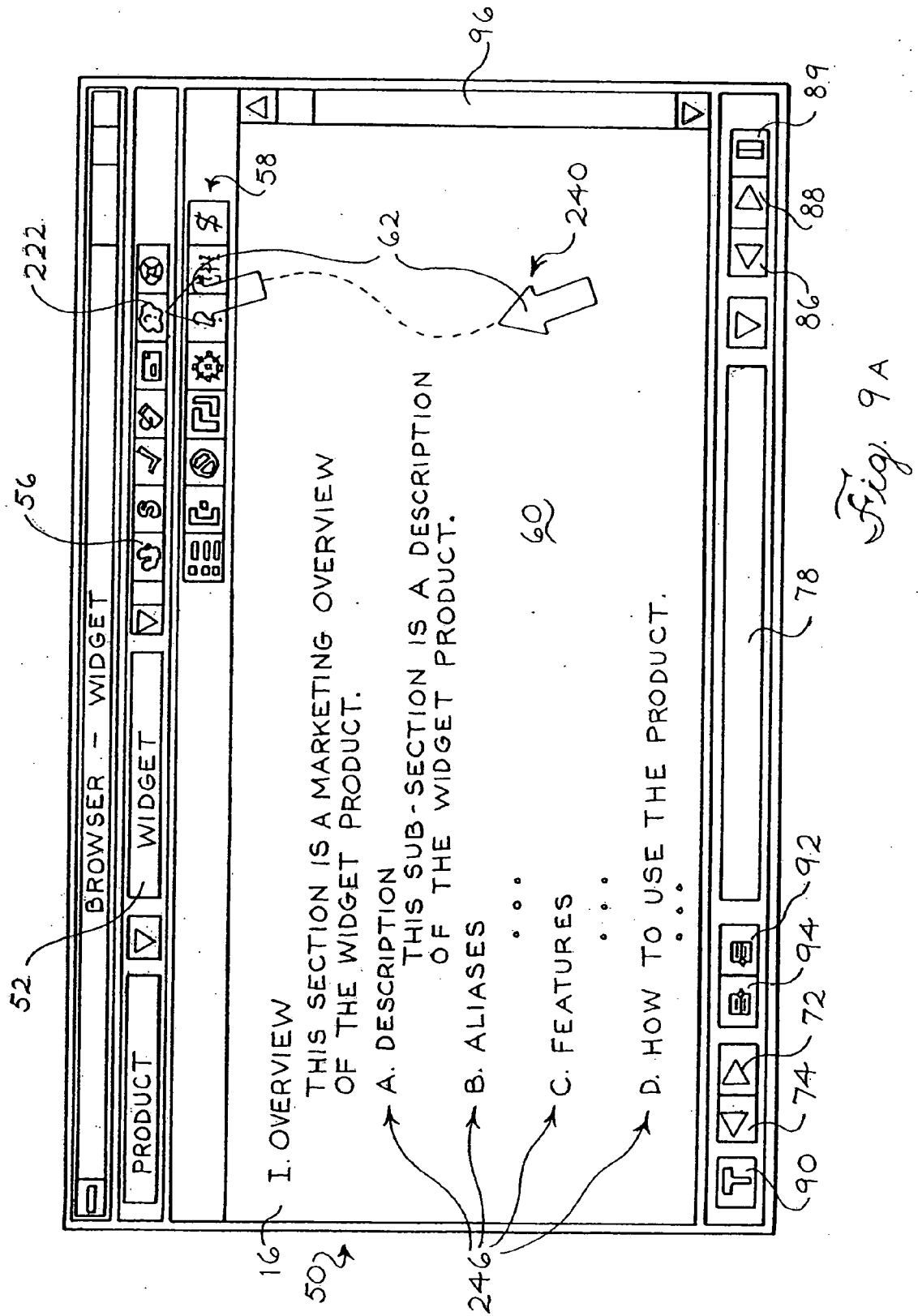
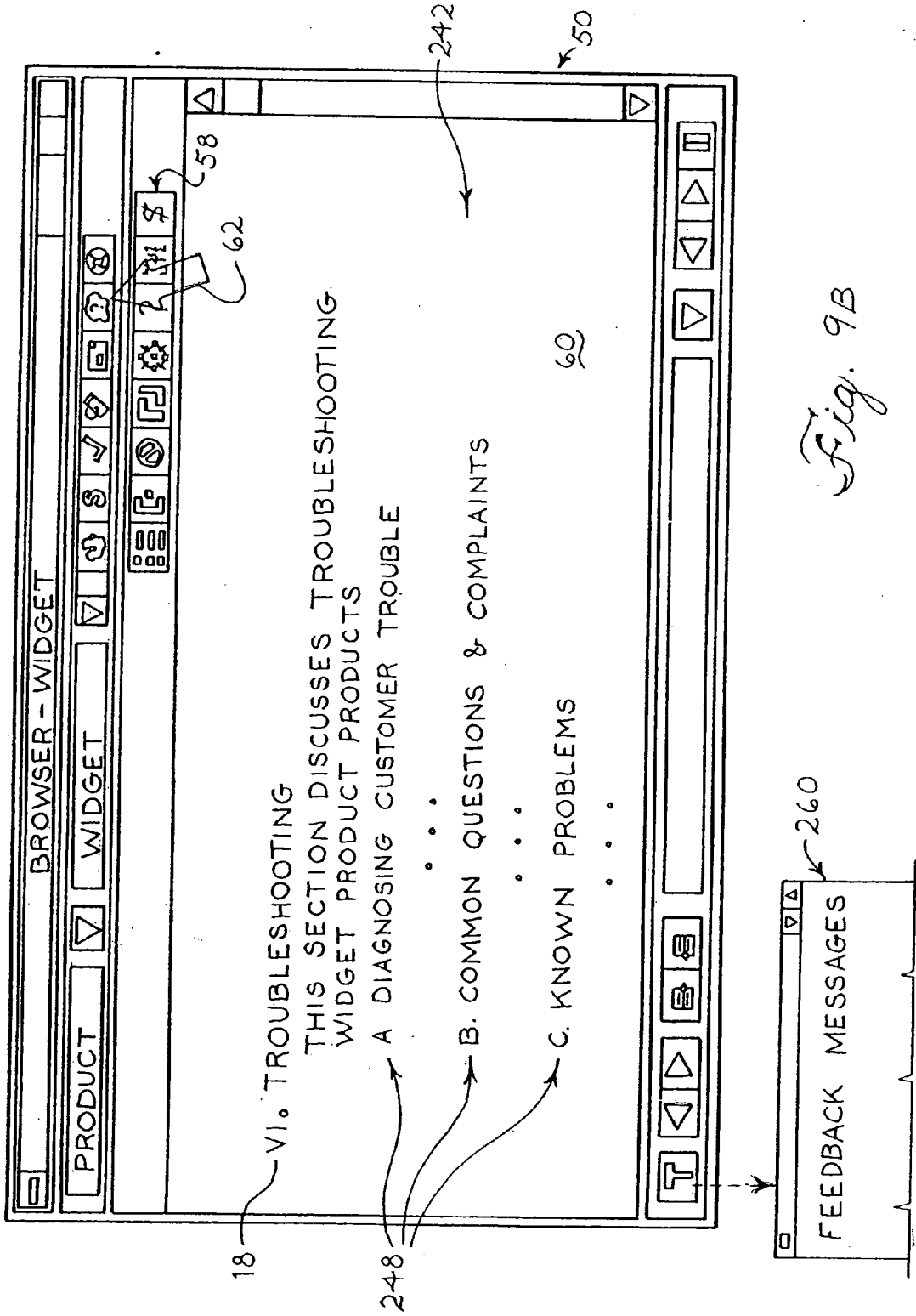


Fig. 8





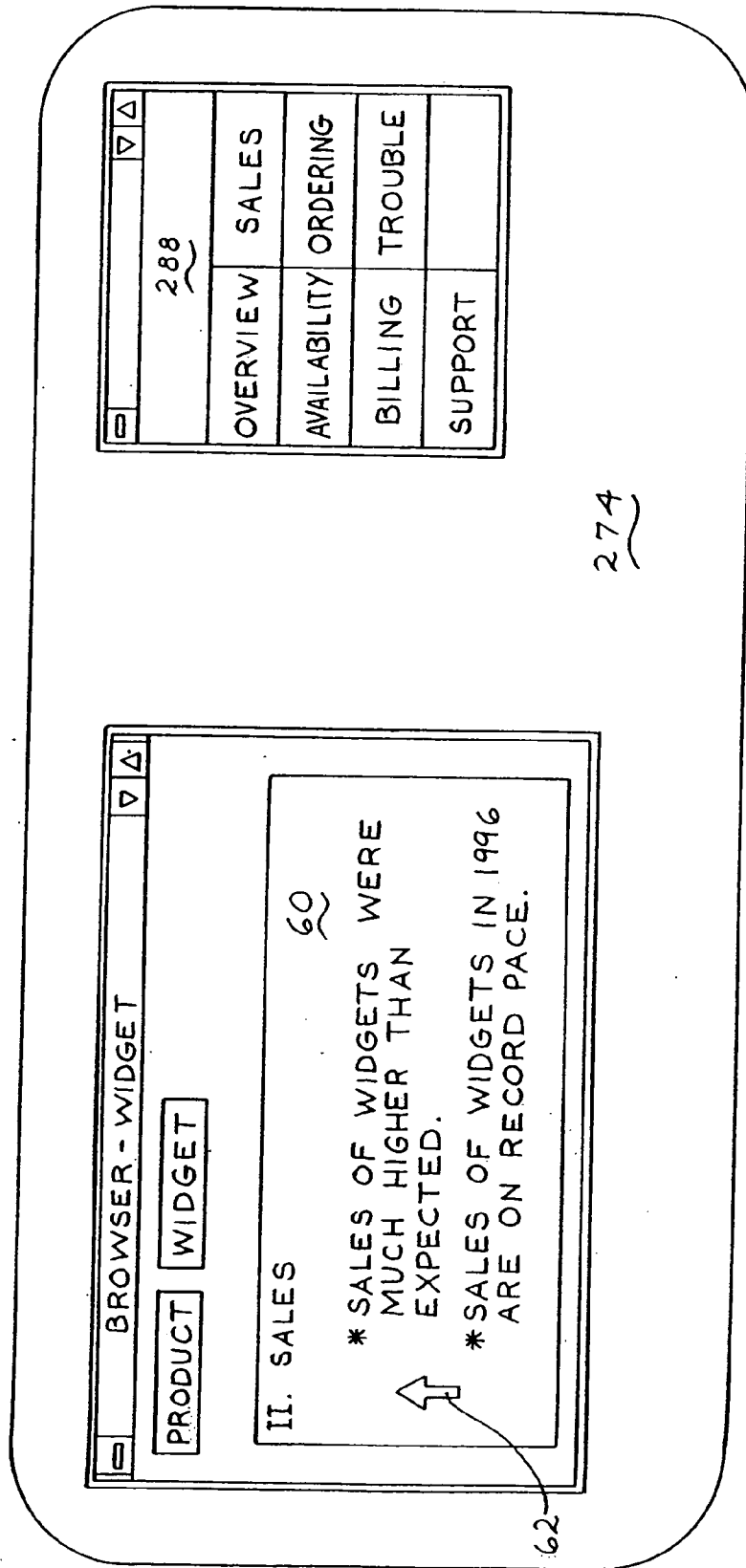
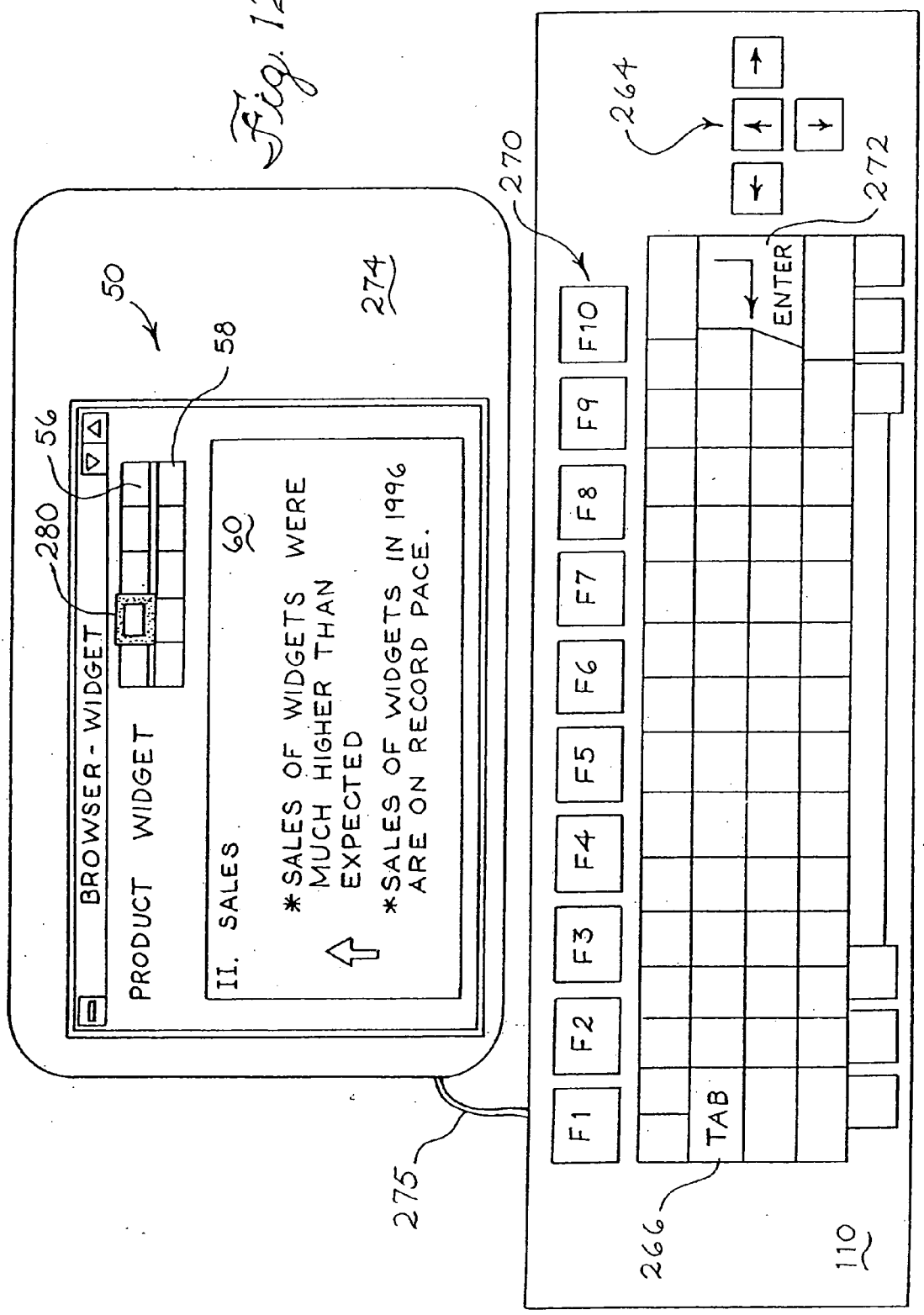


Fig. 11

Fig. 12



275

270

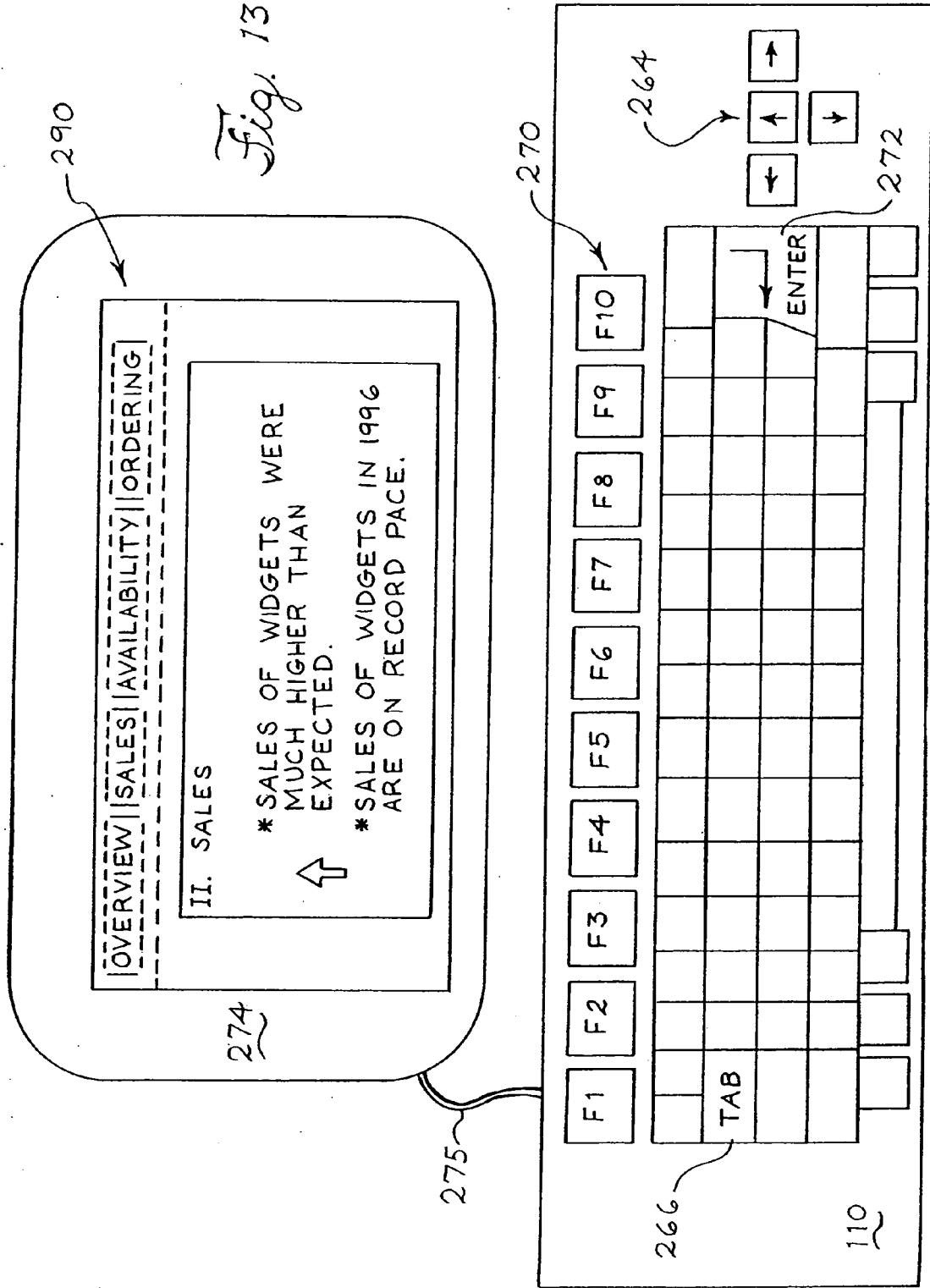
264

272

266

110

Fig. 13



STRUCTURED DOCUMENT BROWSER

[0001] A portion of the disclosure of this document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by any one of the patent disclosure as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all rights to copyright protection whatsoever.

FIELD OF THE INVENTION

[0002] This invention relates to computer applications for viewing documents, and in particular, to a computer program for viewing documents having a predefined structure.

BACKGROUND OF THE INVENTION

[0003] Current computing environments typically include a graphical user interface (GUI). IBM personal computers and IBM compatibles run either OS/2 or Windows. The Macintosh has always had GUI capabilities as part of its operating system. Even higher end computers, including those that run Unix or VAX/VMS operating systems, are available with components that provide a GUI environment.

[0004] GUI's are generated using a set of software tools that put graphical objects on the computer display. One of these graphical objects is a screen pointer that the user controls with a mouse or a trackball. The user moves the mouse to position the pointer on the screen over selected objects on the screen. The user can select an object by using a selecting device which is typically a mouse switch. By selecting an object, the user instructs the operating system or an application to execute the function associated with the object. The GUI objects can include the graphical representation of buttons, menus or any other graphic object.

[0005] GUI's are the foundation of hypertext and hypermedia applications. Such applications allow computer users to create interfaces in which graphical objects are configured to correspond, or to link to objects of information. For example, a user may create an interface having the graphical representation of a button to display a motion video by selecting the button with a screen pointer. Similarly, the button can be configured to make a sound, display an image or display a separate text file.

[0006] Today's increasing interest in the Internet is due in part to the improvements in hypermedia applications. The World Wide Web is growing dramatically due to the evolution of standard markup languages that allow users to mark documents with links to other documents, and of presenters or viewers that interpret the markup language in the documents allowing the users to view the links.

[0007] One of the first markup tools was the Standard Generalized Markup Language (SGML). SGML was developed by the International Standards Organization and has been adopted by the Department of Defense and other government agencies as a way of standardizing documentation. SGML is machine-based in a manner similar to a computer language. An SGML Document Type Definition (DTD) may be defined according to the specifications of SGML for a given document structure. The DTD defines elements to be embedded in a document. The documents are then viewed using a viewer or browser that interprets the elements in the document according to the data structure defined in the DTD.

[0008] The HyperText Markup Language (HTML) is an instance of a DTD defined by SGML. HTML elements or codes are embedded into documents for use with HTML browsers. Browsers display the documents according to stylesheets associated with the embedded codes. The stylesheets contain rules or instructions that dictate the appearance of a document as presented by a browser. Stylesheets may also contain references to other documents in different computers. These references may be used in conjunction with context-sensitive regions of the documents such that a user retrieves the document reference by selecting the region. In this manner, an author links one document to others in a meaningful way such that a viewer may provide a user with access to information in documents that are linked together in a web-like fashion.

[0009] One common characteristic of many browsers is that the links to information are presented solely within documents. The links may take the user to other documents or to locations within the same document, but typically, the elements that provide the link-control are within the documents.

[0010] Having the control to the information links within the documents themselves is adequate where the documents are short and where the purpose is to obtain information in brief, concise statements. But where a document is long, it becomes difficult to browse the document since the only potential access to other destinations are in whatever part of the document is currently being displayed.

[0011] Moreover, organizations often work with standardized documents. These documents typically have a carefully defined purpose and are usually characterized by a standard structure. These documents may be long and the main purpose for viewing the documents is often to access information found in a specific section of the well-known structure of the document.

SUMMARY OF THE INVENTION

[0012] In view of the above, a structured document browser is provided with a user interface that remains uniform and familiar as the user browses documents according to their structure instead of their contents. The browser uses codes embedded in the document to identify sections of the structure of the document. The user interface includes graphical user interface objects, such as buttons, that are configured to display a standard section of the document structure when selected by the user.

[0013] The structured document browser may also include menus that allow a user to select a document from among different documents, or a document structure type from among different structure types.

[0014] It is another object of the invention to provide a method for browsing a document by using a screen pointer to select a graphical user interface object and displaying a section of the document that corresponds to the graphical user interface object.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIG. 1 is a block diagram of a preferred embodiment of the structured document browser showing how it communicates with other components of the computer.

[0016] FIG. 2 illustrates a sample structure for a document.

[0017] FIG. 3 illustrates a portion of a document having the structure shown in FIG. 2 after it has been marked with SGML tags.

[0018] FIG. 4 illustrates an SGML document type definition (DTD) created for use by a structured document browser for a document having the structure shown in FIG. 2.

[0019] FIG. 5 is a representation of an example of a user interface of the structured document browser for FIG. 1.

[0020] FIG. 6 is a flow chart showing the process of retrieving a structured document and illustrates playing the section of a document in response to the selection of a button.

[0021] FIG. 7 illustrates an example of a bks data structure.

[0022] FIG. 8 is a diagram that shows the interaction between a button, a map file and bit map file.

[0023] FIGS. 9A & 9B illustrate the operation of the browser of FIG. 1.

[0024] FIG. 10 illustrates one example of an alternative implementation of the user interface.

[0025] FIG. 11 illustrates a second alternative implementation of the user interface.

[0026] FIG. 12 illustrates a third alternative implementation of the user interface.

[0027] FIG. 13 illustrates a fourth alternative implementation of the user interface.

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EMBODIMENTS

[0028] In the description that follows, reference is made to the drawings where like elements are identified by like numerals throughout.

[0029] A presently preferred embodiment of this invention includes an application program called browser.exe that has been developed using the 'c' programming language in the Windows environment. The browser.exe executable file is programmed to make function calls to three dynamic link libraries named sit.dll, cgrmzv.dll and ct13d.dll. These libraries are components of Dynatext Version 2.0, a user interface development system from Electronic Book Technologies, Inc. These Dynatext libraries provide functions that implement the SGML related functions and the graphic input/output functions. Further information regarding the Dynatext program may be obtained by contacting Electronic Book Technologies, Inc. at One Richmond Square, Providence, R.I. 02906.

[0030] The browser.exe program uses data structures in several support files that are in the same directory tree as browser.exe. These support files will be described in more detail in the description that follows.

[0031] A listing of the present version of browser.exe is attached as Appendix I of this specification. The listing is an octal representation of browser.exe. The presently preferred embodiment may be carried out by converting the octal to a

binary executable file using methods that are well known in the art. After conversion, the browser.exe file may be executed from a directory that includes the dynamic link libraries and the support files described in this disclosure.

[0032] It is to be understood however, that an embodiment of the present invention may be developed for any computing environment using any suitable development system.

[0033] The browser application in a presently preferred embodiment is referred to in the following detailed description as the browser 80 as shown in FIG. 1. FIG. 1 is a block diagram describing at a high level the browser 80 in the computing environment. The components of the browser 80 in its operating environment include the browser 80 itself, an operating system 104 with GUI capabilities, storage media 108, a keyboard 110, a pointing device 100, a selecting device 102 and a monitor 112. The operating system 104 further includes an I/O system 114 and a GUI system 116.

[0034] The double headed arrows 118 denote the communication between the respective components. The communication at 118 may entail communication over a network where appropriate.

[0035] The hardware devices 108, 110, 100, 102, 112 may be implemented by choosing from among many alternatives for each device. A pointing device 100 may be implemented using a mouse, a trackball or any other device that controls the position of a screen pointer. A selecting device 102 is typically implemented with mouse buttons or buttons that operate in conjunction, with a trackball. In general, any device that may be used to effect the selection of an object at the location of the screen pointer may be used as a selecting device 102. A selecting device 102 may even include a key on the keyboard 110. The storage media 108 is understood to include random access memory (RAM), the temporary storage out of which programs are executed as well as the mass storage devices in which programs are stored. The hardware devices 108, 110, 100, 102, 112 are understood to include the software drivers necessary for their operation in the computing environment.

[0036] The browser 80 includes at least a controller 82 and a system interface 84. The system interface 84 is responsible for processing the communication between the browser 80 and the operating system 104, the I/O system 114 and the GUI system 116. The controller 82 receives and interprets requests from the system interface 84 to perform a browser function. For example, the system interface 84 receives signals from; the I/O system 114 that the pointing device 100 and the selecting device 102 were used to press a button or icon, to request a display of a section of a document. The controller 82 receives the information from the system interface 84 to determine which document section to display.

[0037] In a presently preferred embodiment, the system interface 84 includes the functions provided by the Dynatext development system and any operating system or I/O system functions. The controller 82 in a preferred embodiment is the executable program browser.exe. It is to be understood that the diagram in FIG. 1 is by way of illustration and is not intended to limit the software structure chosen to carry out the invention.

[0038] The browser 80 operates with documents that have been prepared as described below. Because the browser 80 is designed to navigate documents according to their struc-

ture, the utility of the browser **80** is maximized when an organization establishes a standard structure for its key documents. A software engineering group, for example, may find it desirable to maintain a uniform structure for the software requirement specifications that the group develops. A different structure is desirable for the group's design specifications, and yet a different structure works for the group's test documentation. The group's goal for such documentation is to maintain uniformity.

[0039] Referring to FIG. 2, a marketing group might maintain product descriptions for its company's product line in documents having the predefined document structure **10**. This structure is predefined to have headings **12** that provide overview information, sales information, product availability information, ordering information, billing information, troubleshooting information and product support information. The structure also has sub-headings **14** within each heading where relevant. The overview section **16** has sub-headings **14** for sections devoted to a product description, aliases, product features and instructions on how to use the product.

[0040] A specific document of the predefined structure in FIG. 2 is marked with codes for viewing with the browser **80**. In a preferred embodiment, codes are used to mark the document as shown in FIG. 3. The codes shown in FIG. 3 have angle brackets around them. In a preferred embodiment, the codes are SGML elements. These codes may be replaced by elements of other markup tools in alternative embodiments. In the marked document **20**, the application identification code **22** indicates that the document has been marked for use by the browser **80**. The overall document structure is identified with a product name code **24**. Each standard structure component is then marked with an appropriate code or element.

[0041] The sections in a document are preferably marked according to a convention. First, the beginning of a section is marked by a code or element. For example, the <OVER> element **26** identifies the beginning of the section of the document that provides an overview of the product. The <OVER> element **26** is followed by the heading and text that constitute the Overview section of the document. A begin section name element **28** indicates that the information that follows the element **28** specifies the name that will appear in all documents of the same structure for that section. In the marked document **20** in FIG. 3, the name of the overview section is "Overview"**30**. Immediately after the name, an end section name element **32** indicates the end of that section name such that the text between the begin section name element **28** and the end section name element **32** is the text **30** that will appear in the heading. Similarly, an end section element **34** indicates the end of the section. In this case, the element </OVER> **34** marks the end of the overview section. The convention of marking the beginning and the end of parts of the document is used to mark other sections (as shown at **36**, **38**, **39** and **42**) that form the standard document structure.

[0042] Begin sub-section elements **38** and end sub-section elements **42** mark the sub-sections of the document, using the same convention, with the stipulation that the end elements **39** are in an order that keeps the sub-sections nested within the sections. Documents may have sub-sections within sub-sections.

[0043] In a preferred embodiment, the structure of documents is defined by a document type definition (DTD). The DTD **40** in FIG. 4 illustrates how a document structure is represented in SGML. Codes such as those referenced at **24**, **25**, **26** and **38**, that are used to indicate a section or structure part of the document as shown in FIG. 3, are defined by setting an element name, as shown at **42** and **44** in FIG. 4; setting a heading name, as shown at **47** and **49**; and listing sub-parts as shown at **46**. For example, the code for the product name **24** in FIG. 3 is defined in the DTD **40** as the element name **42** for the high level document structure composed of the list of sections **46**. The list of sections **46** comprises the codes defining the sections of the document. The codes used in FIG. 3 for the begin section codes **25** are defined in the DTD **40** of FIG. 4 as element names **44** for the respective section level data structures. The respective name codes, such as the begin section name element **27** in FIG. 3, is shown as part of a section, as shown at **45** in FIG. 4. The name element **45** is also defined at **51**. If a section has sub-parts, or sub-sections, the codes for the sub-sections within the section are listed in the definition of the section code as shown at **48**. Each sub-section code is then defined individually as shown at **43**. More details on the syntax and constructs of SGML may be obtained by referring to the ISO Standard for SGML ISO 8879: 1986/A1: 1988(E).

[0044] In a preferred embodiment, a document having a basic word processing format may be converted to a marked document **20**, also known as an SGML instance, using an SGML utility. For example, a utility called DYNATAG from Electronic Book Technologies, Inc., uses documents having the structure described in FIG. 2 to create a DTD **40**, illustrated in FIG. 4, as well as an SGML instance **20**, illustrated in FIG. 3. Other SGML utilities may be used to create DTD's and SGML instances. The discussion of the DTD **40** is presented to illustrate how alternative embodiments might implement a program that interprets SGML DTD's to operate with the browser.

[0045] An infinite number of DTD's may be used, including common DTD's such as the HTML DTD. In addition, other embodiments of this invention may not use SGML as a markup language. Any other suitable markup language, including a proprietary markup language, may be used as well, provided that the appropriate software components are available to translate the code. Other embodiments of this invention may be designed to support the use of more than one markup language.

[0046] Once a document has been marked and converted into a format that is appropriate for the browser **80**, the browser **80** may be started on a computer. Referring back to FIG. 1, the user invokes the browser **80** by using the pointing device **100** and the selecting device **102** to select the browser **80** in a manner dictated by the GUI of the operating system **104**. In the preferred embodiment, the operating system **104** is the Windows Operating System (Version 3.1 and later for purposes of this example), the pointing device **100** is a mouse, and the selecting device **102** includes a pair of mouse buttons (left and right buttons). The Windows user interface **106**, as it appears on the monitor **112**, includes a menu bar **68**, icons **66** representing application programs ready for initialization and a screen pointer **62** controlled by the user with the mouse pointing device **100**.

[0047] To initialize the browser **80** in the Windows environment, the mouse **100** may be used to place the screen

pointer **62** over the browser icon **64** in the operating system interface **106** and the left mouse button may be double-clicked.

[0048] Once the browser **80** has been initialized, the graphical user interface changes from the Windows user interface **106** in FIG. 1 to that of the browser user interface as shown at **50** in FIG. 5.

[0049] Referring to FIG. 5, the user interface **50** of the browser **80** gives the user the capability of selecting a particular document and then selecting different sections of the document for display. The objects used to browse through the documents remain on the user interface **50** regardless of where in a document a user is browsing. The user interface remains familiar to the user and responsive to the document structure.

[0050] The user interface **50** of the browser **80** includes a document menu **52**, a document type menu **54**, a first row of selectors **56**, a second row of selectors **58** and a display window **60**.

[0051] The user selects a document for browsing by using the document menu **52**. The document menu **52** is implemented with standard graphical user interface menu objects such that the user selects a document menu arrow **70** to display the full menu of the documents available. The user then selects the document desired from the menu using the selecting device **102** in combination with the screen pointer **62**.

[0052] The user interface **50** of the browser **80** is configured in a manner that allows the document types, or standard structures of different types, to be listed in a document type menu **54**. The document type is selected by the user in the same manner that the user selects a document. The name of the document type is the name or alias of directories designated by the browser to have documents of a consistent structure. When a different document type is selected, the names listed in the document menu are changed according to the names of the available documents having the new document type.

[0053] The selectors **56**, **58** are examples of document navigation tools that may be used for browsing documents in a presently preferred embodiment. More specifically, selectors **56**, **58** are display regions in the user interface that are configured to perform pre-defined operations when the user places the screen pointer **62** over one of the selectors **56**, **58** and then selects it with the selecting device **102**. In the presently preferred embodiment, the display regions are depicted as icons that make them look like buttons with graphic images on them. The image may be designed to convey a sense of the operation to be performed if the icon is selected.

[0054] The first row of selectors **56** is configured to correspond to the first level of sections in a document. For example, the first selector **68** in the first row **56** is configured to correspond to the Overview section **16** in the document shown in FIG. 2.

[0055] Each selector is configured so that when the user places the screen pointer **62** over the selector and then selects it with the selecting device **102**, the system interface **84** (described above with reference to FIG. 1) receives data indicating which selector was selected. The system interface

84 may then determine the document section associated with the selector and send the request to display that section to the controller **82** (described above with reference to FIG. 1). Alternatively, the system interface **84** may send to the controller **82** the identification of the chosen selector and let the controller **82** determine what section to display. The controller **82** will cause the document to be searched for the document section that matches the selector chosen. When the chosen section is found, the browser **80** displays in the display window **60** the section of the document structure that corresponds to the selector.

[0056] Once the desired section is in the display window **60**, the user may navigate within the section by selecting one of the second row of selectors **58**. Each time a selector from the first row of selectors **56** is selected, the second row of selectors **58** is configured to correspond to the sub-sections **14** within the section **12** being displayed (as shown in FIG. 2). A document structure having sub-sections within its sub-sections may also be accommodated so that three rows of selectors might be present in the user interface. The number of sub-sections within sections of a document may be further accommodated with rows of selectors as desired, or as limited by system constraints, such as the size of the display window **60**.

[0057] The user may also navigate within the section by controlling the display window scroll bar **96** with the screen pointer **62** and the selecting device **102**.

[0058] To describe the manner in which the selectors **56**, **58** in the browser user interface **50** are configured, the initialization of the browser will be described in conjunction with exemplary files or data structures that are utilized during browsing operations. It is to be understood that this is only one implementation of the preferred embodiment, and that the files may be replaced, or integrated, or revised to form different data structures without departing from the scope of the invention. Furthermore, the Dynatext development tools may be replaced by other functionally equivalent tools.

[0059] In the presently preferred embodiment, documents are converted into "books" which are actually directory trees that reside in a directory called the "\xyz\books" directory. The terms "book," "collection" and "library" are defined and used according to the specifications of the Dynatext development system.

[0060] In order to create the Dynatext books, the documents that have been coded as illustrated in FIG. 3 are supplied as input to a utility called DYNATAG which is a component of the Dynatext system. DYNATAG creates a DTD (as shown in FIG. 4) and an SGML instance (as shown in FIG. 3) of the document. The SGML instance and the DTD are used as input files to MKBOOK, another Dynatext utility. The MKBOOK utility creates a binary instance of the document, a directory tree, or a "book," and some of the support files used by the browser **80**. For example, the document for Widget having the structure for products descriptions defined in FIG. 2 is processed with MKBOOK to create the book "\xyz\books\widget," a sub-directory of "\xyz\books."

[0061] The browser **80** uses a number of data files to define how a document is found and displayed in the browser. These files and their names in the preferred

embodiment are 1) the browser executable (browser.exe); 2) an initialization file (browser.ini); 3) a set of dynamic link libraries (sit.dll, cgmzv.dll, & ct13d.dll); 4) a bks file which is an ASCII file that contains information about a book, a library or collection of documents, and names in the browser menus (named *.bks where the * represents the name of a document); 5) a bitmap containing up to 100 regions for icons (named *.bmp or default.bmp) and 6) an ASCII file that provides the linking of the document element names to the icons in the bmp file and for pop-up text in the executable (named *.map or default.map).

[0062] The execution of the browser 80 will be described with reference to FIG. 6 which is a flowchart describing the steps taken to browse a document. When the user starts the structured document browser in the manner described above in reference to FIG. 1, the operating system launches the browser.exe executable file as shown at block 120. This file is located in a directory called the “xyz” directory. When browser.exe is launched it first looks for the three required dynamic link libraries (DLL’s) in the same directory as shown at block 122. The DLL’s, supplied by Electronic Book Technologies, contain functions related to the user interface, SGML processing and access to the books.

[0063] If the DLL’s are available, the browser then checks for the browser.ini file as shown in 124. If the browser.ini file is present, the browser 80 reads its contents, as shown at block 126.

[0064] As the sample file in Table 1 shows, the browser.ini file contains objects, or data structures that include the [Files] object, the [DTEXT] object, and the [MAP] object. The [FILES] object defines an annotation file. The annotation file is a repository for feedback from users of the browser regarding the documents being reviewed. The [DTEXT] object contains file names that the executable will use to find the location of the data directory, security key, and public and private directories. The [MAP] object provides the file name of the initial map file that is to be loaded (typically, the name is “default.map”). The map file, as discussed in detail below, contains the associations between the document elements and the icons.

TABLE 1

Sample browser.ini File

[Files]
AnnotationFile=xyz\annot.txt
[DTEXT]
DATA_DIR=xyz\data
DTEXT_AUTH=@xyz\data/security
PUBLIC_DIR=xyz/tmp/public
PRIVATE_DIR=xyz/tmp/private
[MAP]
Icons=default.bmp

[0065] Referring back to FIG. 6, once the browser.ini file is processed at 126, the browser checks for a file called default.bks at block 128. The default.bks file is an ASCII file which provides the browser with the information required to display a standard initial document, such as a document that displays a message of the day.

[0066] If a default.bks file is not found, the browser initializes without a book as shown at 130. Otherwise, the

contents of default.bks are loaded as shown at 132 and the user interface is presented on the display to the user as shown at 134.

[0067] The bks files will be described by reference to FIG. 7. The bks files are ASCII files that contain information about how the book should be loaded into the menus in the browser and which map file to use. The bks file always begins with the [Book] object 140 in FIG. 7. The next line identifies the “collection.” In a preferred embodiment, a collection, also called a library, refers to the directory containing the books to which the browser 80 has access. In FIG. 7, the collection is located in the “xyz” directory.

[0068] The next line in a bks file as shown in FIG. 7 identifies the collection title 144 which defines a document type for the document type menu 54. The collection title 144 is an alias for the group of books that will be listed under a heading in the document type menu 54 called “Products.”

[0069] The name of the book 146 is on the next line of the bks file. The book name is actually the name of the directory tree that must be present under the collection path specified in line two 142 of the bks file. The book title at 148 specifies an alias for the book. The book title contains the name the user will see on the document menu 52 (at FIG. 5). The map file name 150 is optional. It designates the map file that associates the tag names with the icons in the .bmp file. If specified, the browser will use the file name 152 to access the map file. Otherwise, the browser will assume a file name based on a pre-defined naming convention.

[0070] Referring back to FIG. 6, a user requests a document 160 by using the document menu 52 or the document type menu 54 in the user interface 50 (shown in FIG. 5). When the menu is selected (before a document is selected), the bks file is read in and the menu lists document and document type names according to the contents of the bks file. When a user requests to receive a document as shown at 160, the browser retrieves the document itself as shown at 162 and the map file associated with that document as shown at 164.

[0071] The browser then verifies that the elements in the map file match the SGML DTD and the structure codes in the document as shown at 166. If there are no discrepancies, the browser 80 reads the bmp file as shown at 168 which is specified in the map file. Also, in block 168, the browser 80 locates fly-by text for description of icons. The bmp file allows the browser to display the icons as shown at 170 for the selectors 56, 58 (in FIG. 5). The browser also displays the text in the chosen document as shown at 170.

[0072] Once the selectors on the user interface match the structure requirements of the document, the user may select a section of the document to view by pressing a selector button that corresponds to that section as shown at 172. The correspondence between the selectors 56, 58 and the document structure is established in the map file and in the bmp file. This correspondence will be described with reference to FIG. 8.

[0073] FIG. 8 shows a portion of the map file 200 for the document having the structure in FIG. 2, a bmp file 210 and the location on the user interface in which the selectors are placed 220. The first object in the map file is the [MAP] object 180 which identifies the data structure. The next line in the map file is the icon line 182 which is a filename that

is used by the browser 80 to obtain the bmp file for the structure corresponding to the map file.

[0074] The next line in the map file shown in FIG. 8 contains the [SECTIONS] object 184. The [SECTIONS] object 184 marks the beginning of a set of definitions of data structures 186 that help tie the selector icons to the sections in the document. The order of these data structures indicates the order in which the sections appear in the document.

[0075] The first item under [sections] is "RBW-DOC, PROD.NAME, OVER=1:Overview"188. The RBW-DOC, PROD.NAME 190 label indicates that the section identified in this line is in the first or highest level section. The OVER label 192 in the line matches the <OVER> tag 26 used to identify the overview section in the marked structure document shown in FIG. 3. The expression OVER=1 192 specifies which icon in the bmp file corresponds to the section identified. The text following the colon at 194 is used in fly-by help or bubble help messages. For example, when the screen pointer 62 is positioned over an icon 222 as shown in FIG. 8, a Help message 196 is displayed to indicate the function of the button. The expression OVER=1 192 and the fly-by-help text 194 are read by the browser 80 during block 168 in FIG. 6.

[0076] An example of the bmp file is shown in FIG. 8 at 210. The icons are stored in the bmp file as a single rectangular bitmap. Each icon is 16 pixels wide by 15 pixels high. In the presently preferred embodiment, the icons make the selectors appear to be buttons. The icon index starts at 1 and proceeds from left to right. So the icon index might appear as follows:

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	<etc>							

[0077] An example of how the map file relates to the bmp file is given in FIG. 8. Assume the user moves the screen pointer 62 to the troubleshooting icon, which is the sixth selector or 222 in the top row of selectors 56 and then selects it. The sixth button 222 corresponds to the sixth icon 224 in the bmp file. The browser 80 refers to the map file at the line RBW-DOC, PROD.NAME,TROUBL=6:Troubleshooting 226 to determine which element name must be searched. The line at 226 defines the TROUBL element as the element that must be searched when the sixth button in the first row 222 is selected.

[0078] While every book could have a separate map file and bmp file, these files relate to all books in a library. In other words, all books in the product library should have the same document structure (i.e., document type definition). Thus, a prd.map provides the mappings for all books in the products (i.e., prd) library and, a prd.bmp provides the icons for all books in the products (i.e., prd) library. The icons should relate to the hierarchical structure of the document structure as specified in the document type definition.

[0079] FIGS. 9A & 9B demonstrate the operation of the browser 80 by illustrating the way in which the user interface 50 changes in response to the pressing of a button. FIG. 9A shows the user interface 50 in an initial state with the Overview section 16 of the widget products document from

FIG. 2 in the display window 60. The selectors 58 of the second row are represented by icons that correspond to the sub-sections 246 of the description section.

[0080] The user of the browser 80 may wish to view information that is known to exist in the troubleshooting section 18 of the widget product description. As shown in FIG. 2, the troubleshooting section 18 is towards the end of the document. Standard documents may be as short as one printed page or long enough to fill several binders. The browser 80 simplifies the retrieval of information by allowing access to a pre-defined section of a document by pushing a button. The screen pointer 62 may be moved from an initial position 240 to the position over the icon for the troubleshooting button 222. The user then selects the troubleshooting button 222 by enabling the selecting device 102. The browser 80, using the process illustrated in FIG. 8, then searches through the binary instance of the document for the troubleshooting section. The browser 80 displays the found section 242 in the display window 60 as shown in FIG. 9B. In addition to displaying the found section 242, the browser 80 updates the second row of selectors 58 to correspond to the sub-sections 248 in the found section 242.

[0081] The user interface 50 of the browser 80 may be enhanced by adding objects to give the user more tools with which to view the documents. The user interface 50, shown in FIG. 9A, includes a next section button 92, a previous section button 94, a go forward button 72, a go backward button 74 and string search tools 78, 86, 88, 89. These objects may be programmed into the browser 80 along with the software components that provide the indicated functions.

[0082] When the next section button 92 is selected with the combined action of the screen pointer 62 and the selecting device 102, a user views the next section in the document. For example, the next section after the Overview section 16 in FIG. 9A is the Sales section (See FIG. 2). Selecting the next section button 92 in FIG. 9A causes the browser 80 to display the Sales section. The previous section button 94 operates in the same manner as the next section button 92 except that the previous section is shown.

[0083] The go forward button 72 and go backward button 74 may be used to scroll text in the display window 60.

[0084] The user interface 50 as shown in FIG. 9A may also include string search tools 78, 86, 88. The string search entry box 78 may be used to input a text string that the user wishes to locate in the document. The next found and previous found buttons 86, 88 may be used to display the locations in the document in which the string was found. The clear search button 89 clears the text in the search entry box 78.

[0085] The feedback entry function gives the user the ability to provide feedback on a document for those who may browse the document at a later time. As shown in FIG. 9B, by selecting a feedback file in a menu, or by selecting a tools button 90, a text box 260 opens up to allow the user to enter a note. The text box 260 may be a compilation of messages to which users append notes, or the contents of the text box 260 may be saved into a separate repository of data periodically. In a preferred embodiment, the compilation of messages may be saved to a SGML-based file for support as a document that may be viewed by implementation of the browser 80. In addition, the textbox 260 may be replaced by a view of the messages in the display window 60.

[0086] It is to be understood that the appearance of the user interface 50 shown in FIGS. 9A & 9B is one example of the user interface in the present invention. The appearance and the choice of graphic objects may be varied to suit the needs of the intended users.

[0087] Referring to FIG. 10, one example of how the user interface 50 may be altered replaces the buttons 56, 58 with other objects. The buttons are merely display regions of the user interface configured to perform a function when selected with the screen pointer 62 and the selecting device 102. In the presently preferred embodiment of FIG. 9A, the selectors are represented by button icons. As shown in FIG. 10, these icons may be replaced with words or phrases 268 that are descriptive of the section that they are configured to display.

[0088] Another variation, shown in FIG. 11, uses a distributed user interface in which the buttons 288 are located in their own window that is detached from the display window 60. FIG. 11 illustrates the separate windows 50, 288 as they might appear on a monitor screen 274.

[0089] In another variation shown in FIG. 12, keys on the keyboard 110 may be configured as functional equivalents of the screen pointer 62 to select document section selectors 56, 58. FIG. 12 illustrates a monitor screen 274 connected to a keyboard 110 via connection 275. In one approach to using the keyboard, the selectors 56, 58 may be mapped to function keys 270 on the keyboard 110. In another approach which may be combined with the first approach, the browser 80 may first, highlight a selector in response to certain keys such as a TAB key 266, or an arrow key 264, and then select the highlighted selector 280 in response to another key such as the ENTER key 272.

[0090] The user interface 50 may also be implemented in an environment that lacks a GUI, such as a character-based system interface. In an example of such an implementation shown in FIG. 13, the selectors 56, 58 are words or phrases that have features such as a character-based border identifying them as selectors. The user then selects a selector using the keys on the keyboard as described above.

[0091] In another example of a character-based user interface 50, the selectors 56, 58 are not used at all and the entire screen is the display window. Function keys 270 on the keyboard 110 are implemented in place of the selectors. The function keys 270 may be mapped according to the labels indicated at 290.

[0092] It is to be understood that this specification is provided by way of illustration and that it is only the claims and their equivalents that define the invention.

What is claimed is:

1. A system for displaying product information, the system comprising:

a processor;

memory, in communication with the processor, that stores:

operational instructions that, when executed by the processor, cause the processor to send data to display a plurality of selectors in a row in a region of a user interface of a browser, each of the plurality of

selectors configured to correspond to a respective portion of product information;

operational instructions that, when executed by the processor, cause the processor to send data to display a menu selector in the row in the region of the user interface, wherein selection of the menu selector results in a display of a menu of selection choices, and wherein the menu selector is not one of the plurality of selectors;

operational instructions that, when executed by the processor, cause the processor to send data to display a fly-by help message indicating a function of a selector of the plurality of selectors after a screen pointer is positioned over the selector of the plurality of selectors;

operational instructions that, when executed by the processor, cause the processor to receive a selection of one of the plurality of selectors; and

operational instructions that, when executed by the processor, cause the processor to send data to display, at a same time as the plurality of selectors, a portion of the product information that corresponds to the one of the plurality of selectors, wherein the portion of the product information is displayed in a display window outside of the region of the user interface.

2. The system of claim 1, wherein the system is part of a computing environment that includes communication over a network, and wherein the processor is in communication with the network.

3. The system of claim 1, wherein the memory further stores:

operational instructions that, when executed by the processor, cause the processor to send, over a network, the data to display the plurality of selectors.

4. The system of claim 1, wherein the memory further stores:

operational instructions that, when executed by the processor, cause the processor to send data to display at least one additional selector in response to receiving the selection of the one of the plurality of selectors.

5. The system of claim 1, wherein the plurality of selectors are chosen from a larger set of selectors.

6. A method for displaying product information, the method comprising:

displaying a plurality of selectors in a row in a region of a user interface of a browser, each of the plurality of selectors configured to correspond to a respective portion of product information;

displaying a menu selector in the row in the region of the user interface, wherein selection of the menu selector results in a display of a menu of selection choices, and wherein the menu selector is not one of the plurality of selectors;

displaying a fly-by help message indicating a function of a selector of the plurality of selectors after a screen pointer is positioned over the selector of the plurality of selectors;

receiving a selection of one of the plurality of selectors;
and

displaying, at a same time as the plurality of selectors, a portion of the product information that corresponds to the one of the plurality of selectors, wherein the portion of the product information is displayed in a display window outside of the region of the user interface.

7. The method of claim 6, wherein the data to display the plurality of selectors is sent over a network.

8. The method of claim 6 further comprising:

displaying at least one additional selector in response to receiving the selection of the one of the plurality of selectors.

9. The method of claim 6, wherein the plurality of selectors are chosen from a larger set of selectors.

10. A computer-readable storage medium comprising:

operational instructions that, when executed by a processor, cause a processor to send data to display a plurality of selectors in a row in a region of a user interface of a browser, each of the plurality of selectors configured to correspond to a respective portion of product information;

operational instructions that, when executed by the processor, cause the processor to send data to display a menu selector in the row in the region of the user interface, wherein selection of the menu selector results in a display of a menu of selection choices, and wherein the menu selector is not one of the plurality of selectors;

operational instructions that, when executed by the processor, cause the processor to send data to display a

fly-by help message indicating a function of a selector of the plurality of selectors after a screen pointer is positioned over the selector of the plurality of selectors;

operational instructions that, when executed by the processor, cause the processor to receive a selection of one of the plurality of selectors; and

operational instructions that, when executed by the processor, cause the processor to send data to display, at a same time as the plurality of selectors, a portion of the product information that corresponds to the one of the plurality of selectors, wherein the portion of the product information is displayed in a display window outside of the region of the user interface.

11. The computer-readable storage medium of claim 10 further comprising:

operational instructions that, when executed by the processor, cause the processor to send, over a network, the data to display the plurality of selectors.

12. The computer-readable storage medium of claim 10 further comprising:

operational instructions that, when executed by the processor, cause the processor to send data to display at least one additional selector in response to receiving the selection of the one of the plurality of selectors.

13. The computer-readable storage medium of claim 10, wherein the plurality of selectors are chosen from a larger set of selectors.

* * * * *