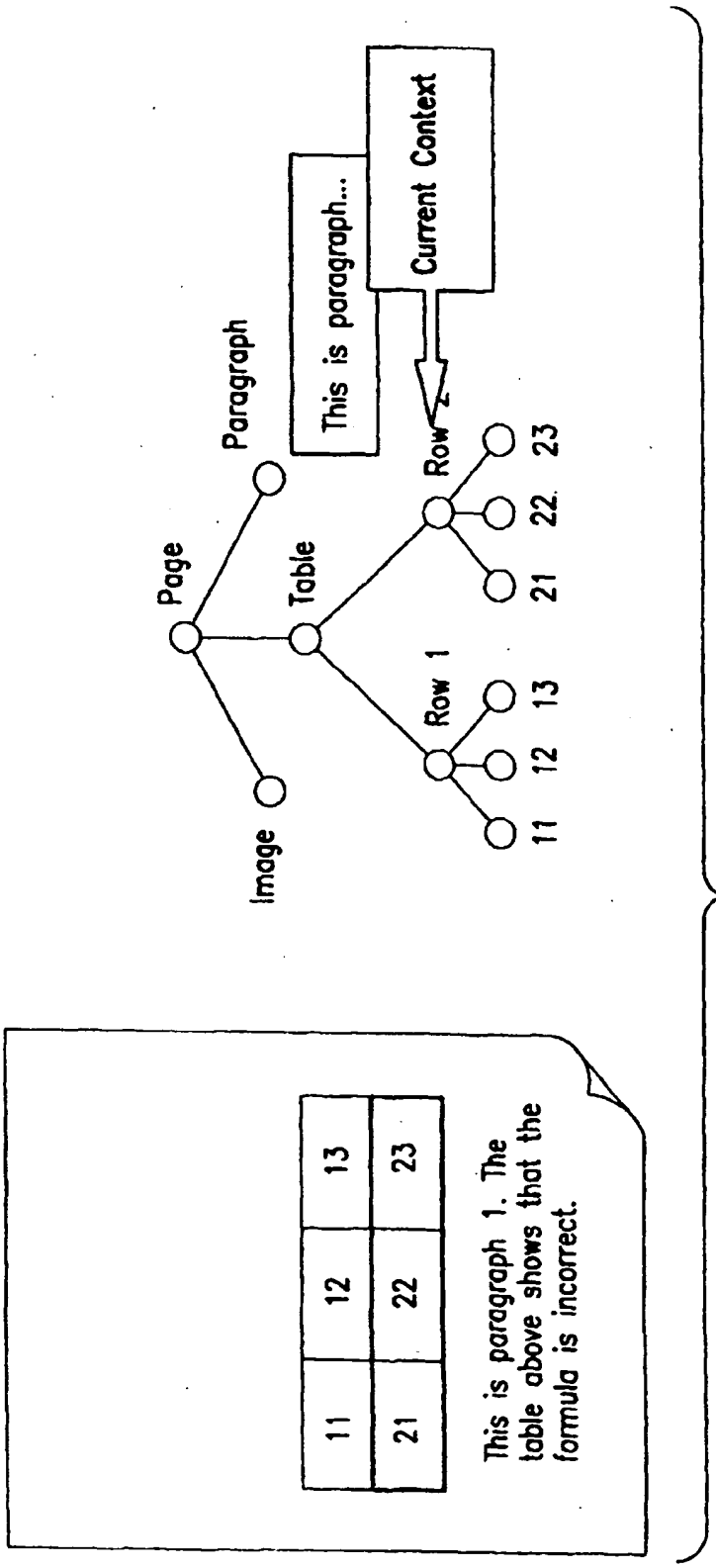


FIG.17

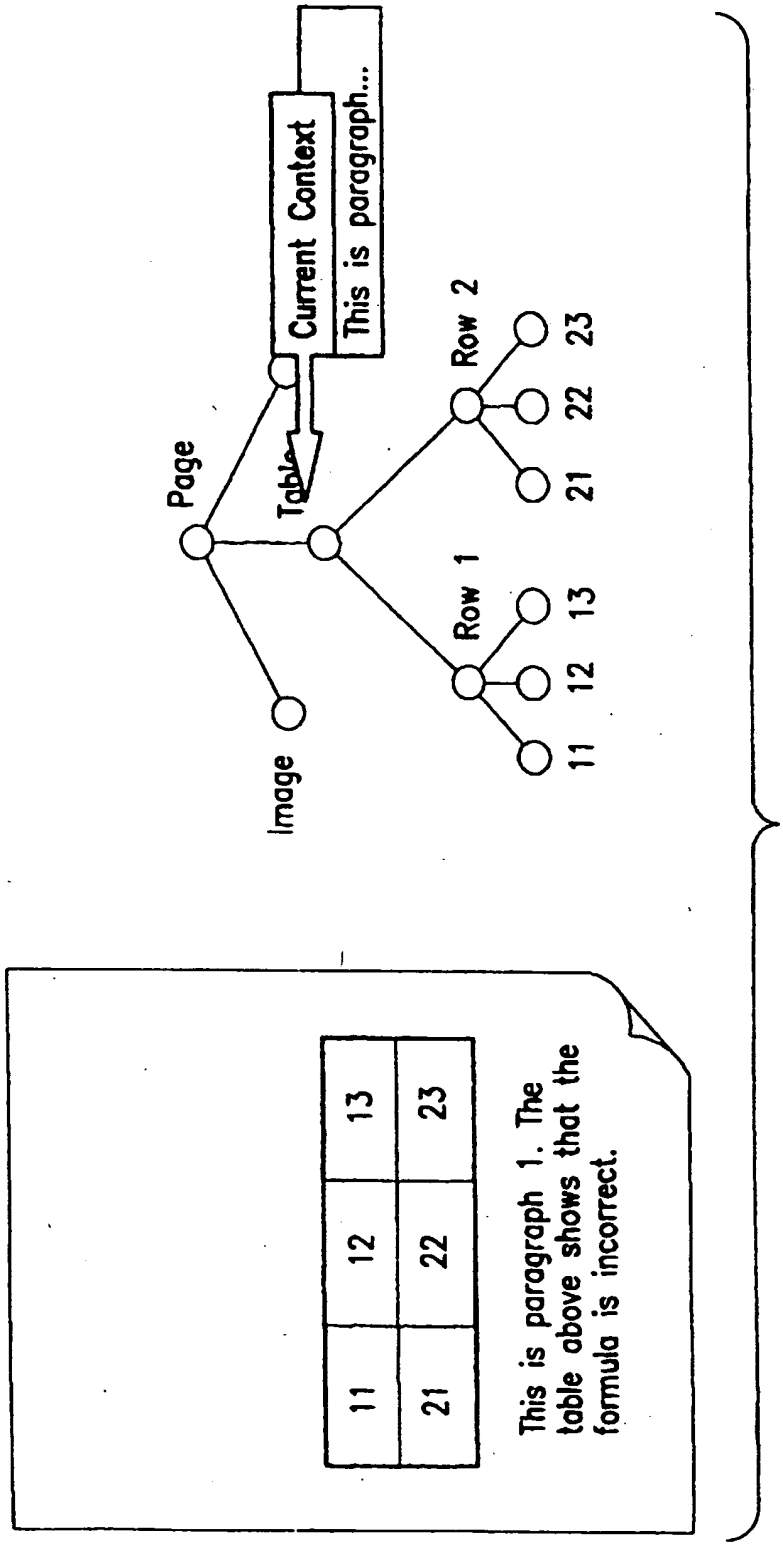


FIG.18

EP0926607

Publication Title:

Object-oriented system for mapping structered information to different structured information

Abstract:

Abstract of EP0926607

An object-oriented system and computer program product for mapping structured information to different structured information, which allows a user to interactively define the mapping. The present invention operates as an object-oriented user tool by accepting interactive input from a user of a source input, by processing the input to display the source input in a format for accepting and processing user commands to create or edit a transformation map of source components to target components. Interactive user input is then accepted and processed for selection of an input file to be transformed and selection of a transformation map to be used for the requested transformation. Interactive user input is accepted and processed for selection of individual components of the first structured information format for mapping, and for selection of options for the target components. Exemplary options for the target components are a null value, the source component itself, a single selected target component, or plural selected target components. Interactive user input is accepted for processing to assign attribute values to components of the second structured information format. Exemplary options for the sources of attribute values are attribute values o 103a btained from the source components, system attribute values, no value, attribute values input interactively by the user, and content of element. Interactive user input is then accepted and processed to initiate processing of a transformation of the source input file in the first structured information format to a target output file in the second structured information format.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 926 607 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
30.06.1999 Bulletin 1999/26

(51) Int. Cl.[6]: **G06F 17/30**

(21) Application number: **98124276.1**

(22) Date of filing: **18.12.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **23.12.1997 US 997482**
23.12.1997 US 997705

(71) Applicant: **Ricoh Company**
Tokyo 143-8555 (JP)

(72) Inventors:
- **Motoyama, Tetsuro,**
c/o RICOH CORPORATION,
Systems
San Jose, CA 95134-2088 (US)
- **Fong, Avery,**
c/o RICOH CORPORATION,
Systems
San Jose, CA 95134-2088 (US)
- **Bhatnagar,Anurag,**
c/o RICOH CORPORATION,
Systems
San Jose, CA 95134-2088 (US)

(74) Representative:
Schwabe - Sandmair - Marx
Stuntzstrasse 16
81677 München (DE)

(54) Object-oriented system for mapping structered information to different structured information

(57) An object-oriented system and computer program product for mapping structured information to different structured information, which allows a user to interactively define the mapping. The present invention operates as an object-oriented user tool by accepting interactive input from a user of a source input, by processing the input to display the source input in a format for accepting and processing user commands to create or edit a transformation map of source components to target components. Interactive user input is then accepted and processed for selection of an input file to be transformed and selection of a transformation map to be used for the requested transformation. Interactive user input is accepted and processed for selection of individual components of the first structured information format for mapping, and for selection of options for the target components. Exemplary options for the target components are a null value, the source component itself, a single selected target component, or plural selected target components. Interactive user input is accepted for processing to assign attribute values to components of the second structured information format. Exemplary options for the sources of attribute values are attribute values obtained from the source components, system attribute values, no value, attribute values input interactively by the user, and content of element. Interactive user input is then accepted and processed to initiate processing of a transformation of the source input file in the first structured information format to a target output file in the second structured information format.

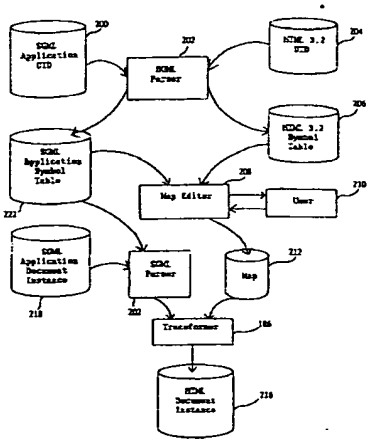


Fig. 5

EP 0 926 607 A2

Description

CROSS-REFERENCES TO RELATED APPLICATION

[0001] This application is related to and being concurrently filed with another patent application: U.S. Patent Application S/N 08/XXX,XXX, Attorney Docket No. 5244-0063-2X, entitled "Method and Apparatus For Mapping Structured Information to Different Structured Information" filed on _____, 1997, and incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] This invention relates generally to mapping structured information to different structured information in an object-oriented framework. The present invention relates more specifically to processing a document encoded in a markup language format, a database information format, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, or a DOS file name scheme, transforming it into another markup language format, another database information format, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, or a DOS file name scheme, in an object-oriented framework. The invention is more specifically related to a system and computer program product for mapping in which a user interactively defines the mapping for the transformation in an object-oriented framework.

[0003] This invention also relates generally to providing a user interface for mapping structured information to different structured information. The present invention relates more specifically to providing a user interface for processing a document encoded in a markup language format, a database information format, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, or a DOS file name scheme, transforming it into another markup language format, another database information format, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, or a DOS file name scheme. The invention is more specifically related to a method and apparatus for providing a user interface for mapping in which a user interactively defines the mapping for the transformation.

Discussion of the Background

[0004] Standard Generalized Markup Language ("SGML") is an information management standard adopted by the International Organization for Standardization ("ISO"), as ISO 8879:1986, as a means for providing platform-independent and application-independent documents that retain content, indexing, and linked information. SGML provides a grammarlike mechanism for users to define the structure of their documents and the tags they will use to denote the structure in individual documents. A complete description of SGML is provided in Goldfarb, C. F., *The SGML Handbook*, Oxford University Press, Oxford, 1990, and McGrath, S., *Parseme.1st: SGML for Software Developers*, Prentice Hall PTR, New Jersey, 1998, which are incorporated herein by reference.

[0005] HyperText Markup Language ("HTML") is an application of SGML that uses tags to mark elements, such as text or graphics, in a document to indicate how Web browsers should display these elements to the user and should respond to user actions such as activation of a link by means of a key press or mouse click. HTML is used for documents on the World Wide Web. HTML 2.0, defined by the Internet Engineering Task Force ("IETF"), includes features of HTML common to all Web browsers as of 1995, and was the first version of HTML widely used on the World Wide Web. Future HTML development will be carried out by the World Wide Web Consortium ("W3C"). HTML 3.2, the latest proposed standard, incorporates features widely implemented as of early 1996. A description of SGML and HTML features is given in Bradley, N., *The Concise (SGML) Companion*, Addison Wesley Longman, New York, 1997, which is incorporated herein by reference.

[0006] Object Oriented Programming ("OOP") is a programming methodology in which a program is viewed as a collection of discrete objects that are self-contained collections of data structures and routines that interact with other objects. Many high-level languages, including C++, support the declaration of a class. The class is typically a template, or detailed description, of the objects, or instances of objects, which will be created, or instantiated, by a constructor function during program execution and destroyed by a destructor function when the object is no longer needed. A conversational reference to a class includes all of the objects currently in existence as a result of constructor calls. A class is made up of data items, structures, and methods. Data items correspond to variables of prior programming art. Structures are named groupings of related data items and other structures. Methods correspond to functions and subroutines of prior programming art.

[0007] An object-oriented framework is a reusable basic design structure, consisting of abstract and concrete classes, that assists in building applications.

[0008] Pointers, used for accessing specific objects, data items, and methods, are data items which contain system equivalents of absolute addresses in computer memory. Null pointers, or zero pointers, are pointer variables or literals

which have been assigned a system value, for example, zero, denoting that a specific pointer is currently pointing to a null, or non-existent item. References and reference variables are generally data items which contain system equivalents of absolute addresses in computer memory.

[0009] A string variable or a string literal is a data structure composed of a sequence of characters of the character set of a particular application. A null string, a nil string, or an empty string is a string which contains no characters.

[0010] The three main features of object oriented programming are inheritance, encapsulation, and polymorphism. Inheritance allows a programmer to establish a general class with features which are desirable for a wide range of objects. For example, if a programmer designs a class polygon having certain features such as a closed convex shape made up of plural straight lines joined pairwise at vertices, it is then possible to construct polygon subclasses such as triangles, quadrilaterals, pentagons, and hexagons, all having the shared properties of the parent class polygon, with additional constraints on the number of sides to be allowed for the objects generated. It is also possible, for example, to have subclasses of class quadrilateral such as rectangle and rhombus. A class square inherits all features of the class rectangle and additionally has all of its sides equal in length. The class polygon is considered an abstract class, in that instantiations of actual objects is performed only in its subclasses. However, the class polygon establishes certain properties inherent to all of the non-abstract, or concrete subclasses for inheritance purposes.

[0011] Encapsulation and polymorphism have already been described, and are already well known, in patents relating to object oriented systems. A comprehensive discussion of OOP is provided in Coad, P. and Yourdon, E., *Object-Oriented Analysis, Second Edition*, Prentice-Hall, Inc., New Jersey, 1991, and in Booch, G., *Object-Oriented Analysis and Design with Applications, Second Edition*, Addison Wesley Longman, California, 1994, which are incorporated herein by reference.

[0012] A Graphical User Interface ("GUI") is an environment that represents programs, files, and options by means of icons, menus, and dialog boxes on a screen. An icon is an image, displayed on a screen or other output device, that can be manipulated by a user. By serving as a visual pictorial representation of a function that is available, an icon generates a user-friendly interface by freeing the user of the burden of having to remember commands or type them on a keyboard. A menu is a list of options from which the user can make a selection to perform a desired action. A dialog box is a special window, or area, displayed on a screen or other output device, to solicit a response from the user. In a GUI, the user can select and activate options by pointing and clicking with a mouse or by keystrokes on the keyboard. The preceding descriptions were derived from definitions given in the *Computer Dictionary, Third Edition*, Microsoft Press, Washington, 1997.

[0013] ISO and International Electrotechnical Commission ("IEC") form a specialized system for worldwide standardization. ISO/IEC 9070:1991(E) is an international standard which is applied to an assignment or unique owner prefixes to owners of public text conforming to ISO 8879. The standard describes the procedures for making an assignment and the method for constructing registered owner names from them. Procedures for self-assignment of owner prefixes by standards bodies and other organizations are also specified. ISO/IEC 9070:1991(E) is incorporated herein by reference.

[0014] UNIX and DOS are well-known operating systems for computers. Both UNIX and DOS support a file naming scheme which involve a path from a root directory, through descendant directories, to leaf nodes which are non-directory file names.

[0015] Processing systems are known in which a data processor converts a document encoded in a markup language automatically to another format. For example, Balise software from Computing Art, Inc. processes documents encoded in SGML to convert them to a formatted output for user viewing. However, this software does not allow the user to interactively define the mapping of SGML tags to another format.

SUMMARY OF THE INVENTION

[0016] Accordingly, one object of this invention is to provide a novel object-oriented system and computer program product which can process information encoded in a structured information format to transform the information into another structured information format, and which allows a user to interactively define the mapping for the transformation. Exemplary structured information formats include markup language formats, database information formats, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, and a DOS file name scheme. Exemplary users include human users, software methods, and software objects.

[0017] It is a further object of this invention to provide a novel object-oriented system and computer program product for conversion of Standard Generalized Markup Language ("SGML") documents into HyperText Markup Language ("HTML") documents, allowing a user to interactively define the mapping for the transformation.

[0018] It is a further object of this invention to provide a novel object-oriented system and computer program product for conversion of information in a database format into information in a different database format, which allows a user to interactively define the mapping for the transformation.

[0019] It is a further object of this invention to provide a novel object-oriented system and computer program product

for conversion of information in an ISO/IEC 9070 naming scheme into a UNIX file name scheme, which allows a user to interactively define the mapping for the transformation.

[0020] It is a further object of this invention to provide a novel object-oriented system and computer program product for conversion of information from an ISO/IEC 9070 naming scheme into a DOS file name scheme, which allows a user to interactively define the mapping for the transformation.

[0021] These and other objects are accomplished by an object-oriented system and computer program product for processing information encoded in a structured information format, to transform the information into another structured information format, which allows a user to interactively define the mapping for the transformation.

[0022] An exemplary transformation for the present invention is conversion of SGML documents into HTML documents. For explanation of this example, the present invention has been developed as an object-oriented tool to allow a user to define the transformation of an SGML document into an HTML document or other structured format, for example, a database information format. The user tool for this example is currently implemented in the format of a Graphical User Interface ("GUI") using Object Oriented Programming ("OOP") technology. For this example, the current invention is designed to provide a user with an object-oriented graphic tool to transform documents written in a cryptic SGML format into another structured format for greater viewing ease and for greater portability of documents and information. A user interface object and a map creator object allow the user to select an option of performing a default or conditional mapping. The user is allowed to select an input SGML Document Type Definition ("DTD") object, or a currently existing map object. If the user selects an input SGML DTD object, the user interface object requests that a ParserService object process elements of the SGML DTD into component parts to produce an SGML symbol table object. The user interface object displays individual source component objects of the input for the user to input a selection. A user input object is utilized for accepting the selection. The user interface object and the map creator object provide the user with options for transformation of the individual source component objects such as a mapping of a source component object to a target null value, a mapping of a source component object to itself, a mapping of a source component object to a single target component object, or a mapping of a single source component object to plural target component objects. If the user selects a conditional mapping, then the map creator object checks for special cases, such as a history of an element being referenced previously, and processes special cases using further interactive input from the user by the user interface object.

[0023] The user interface object and map creator object also provide the user with options for assigning attribute values for the target components. Exemplary options are attribute values obtained from the source components, system attribute values, no value, and attribute values input interactively by the user using the user input object.

[0024] The user interface object and map creator object allow the user to interactively select options for transformation, and options for assigning attribute values for the target components, and the selected options are input to objects for properties and objects for attributes. These objects are processed by the map creator object to create a transformation rule object for the source component object.

[0025] The invention accepts and processes interactive user input, using the user input object, for making plural changes to any of the component mapping values the user desires until the user inputs a command to cease the interactive input and create a transformation map. The map creator object initiates processing of the transformation rules to create a transformation map object.

[0026] The user interface object accepts user input into a user input object for selecting an input source file for transformation to a target output file using an already existing map object specified interactively by the user. The user input is then processed, and the requested input file and map are then processed to transform the input file into the requested output file format. The created output file is then sent to the user specified destination.

[0027] Another object of this invention is to provide a novel method, apparatus, and computer program product which provides a graphical user interface for processing information encoded in a structured information format to transform the information into another structured information format, and which allows a user to interactively define the mapping for the transformation. Exemplary structured information formats include markup language formats, database information formats, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, and a DOS file name scheme.

[0028] It is a further object of this invention to provide a novel method, apparatus, and computer program product which provides a graphical user interface for defining conversion of Standard Generalized Markup Language ("SGML") documents into HyperText Markup Language ("HTML") documents, which allows a user to interactively define the mapping for the transformation.

[0029] It is a further object of this invention to provide a novel method, apparatus, and computer program product which provides a graphical user interface for defining conversion of information in a database format into information in a different database format, which allows a user to interactively define the mapping for the transformation.

[0030] It is a further object of this invention to provide a novel method, apparatus, and computer program product which provides a graphical user interface for defining conversion of information from an ISO/IEC 9070 naming scheme into a UNIX file name scheme, which allows a user to interactively define the mapping for the transformation.

[0031] It is a further object of this invention to provide a novel method, apparatus, and computer program product

which provides a graphical user interface for defining conversion or information from an ISO/IEC 9070 naming scheme into a DOS file name scheme, which allows a user to interactively define the mapping for the transformation.

[0032] These and other objects are accomplished by a method, apparatus, and computer program product which provides a graphical user interface for processing information encoded in a structured information format, such as a markup language format, or such as a database information format, to transform the information into another structured information format, such as a markup language format, or such as another database information format, which allows a user to interactively define the mapping for the transformation.

[0033] An exemplary transformation for the present invention is conversion of SGML documents into HTML documents. For explanation of this example, the present invention has been developed as a tool to allow a user to define the transformation of an SGML document into an HTML document or other structured format, for example, a database information format. The user tool for this example is currently implemented in the format of a Graphical User Interface ("GUI") using Object Oriented Programming ("OOP") technology.

[0034] For this example, the current invention is designed to provide a user with a graphic tool to transform documents written in a cryptic SGML format into another structured format for greater viewing ease and far greater portability of documents and information. The user interface provides the user with selectable options of performing a default or conditional mapping. The user interface provides the user with selectable options of selecting an input SGML Document Type Definition ("DTD") or a currently existing map. The user interface displays the input for the user to select individual source components of the input. The user interface provides the user with selectable options for transformation of the individual source components such as a mapping of a source component to a target null value, a mapping of a source component to itself, a mapping of a source component to a single target component, or a mapping of a single source component to plural target components. If the user selects a conditional mapping, then special cases, such as a history of an element being referenced previously, are checked and processed using further interactive input from the user using the user interface.

[0035] The user interface also provides selectable options to the user for assigning attribute values for the target components. Exemplary options are attribute values obtained from the source components, system attribute values, no value, and attribute values input interactively by the user using the user interface.

[0036] The user interface allows the user to interactively select options for transformation, and options for assigning attribute values for the target components, and the selected options are processed to create a transformation rule for the source component.

[0037] The invention accepts interactive user input, to be processed by a map creator, for making plural changes to any of the component mapping values the user desires until the user inputs a command to cease the interactive input and create a transformation map. The transformation rules are processed by a map creator to create the transformation map.

[0038] The invention accepts user input for selecting an input source file for transformation to a target output file using an already existing map specified interactively by the user. The user input is then processed, and the requested input file and map are then processed to transform the input file into the requested output file format. The created output file is then sent to the user specified destination.

BRIEF DESCRIPTION OF THE DRAWINGS

[0039] A more complete appreciation of the invention and many of the attendant advantages thereof will be readily obtained as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, wherein:

Fig. 1A illustrates an exemplary Standard Generalized Markup Language ("SGML") Document Type Definition ("DTD");
Fig. 1B illustrates an exemplary mapping of SGML to HyperText Markup Language ("HTML");
Fig. 1C illustrates an exemplary SGML document;
Fig. 1D illustrates an exemplary HTML document output from a transformation of the SGML document;
Fig. 2 illustrates an exemplary browser output generated using the HTML document shown in Fig. 1D;
Fig. 3A illustrates, in tree format, the hierarchical nature of an SGML document and Fig. 3B illustrates the more "flat" structure of an HTML document;
Fig. 4 illustrates a design of the major components for the SGML to HTML mapping and transformation;
Fig. 5 illustrates, in a data flow diagram format, the flow of data through the SGML to HTML mapping and transformation;
Fig. 6A illustrates the flow of data and interaction of files through the mapping and transformation of information in one structured format to information in another structured format;
Fig. 6B illustrates the flow of data and interaction of files through the SGML to HTML mapping and transformation;

Fig. 7 illustrates a file organization of a map class object for the SGML to HTML mapping and transformation;
Fig. 8A illustrates a map class structure for the map module of the SGML to HTML mapping and transformation;
Fig. 8B illustrates the major classes within the map module of Fig. 8A;
Fig. 8C(1) illustrates a map class structure for a source SGML tag attribute class of the SGML to HTML mapping and transformation;
Fig. 8C(2) illustrates a map class structure for a source SGML content class of the SGML to HTML mapping and transformation;
Fig. 8C(3) illustrates a map class structure far a map service class of the SGML to HTML mapping and transformation;
Fig. 8C(4) illustrates a map class structure for a map create and edit service class of the SGML to HTML mapping and transformation;
Fig. 9 illustrates the hierarchical interaction among major modules of the SGML to HTML mapping and transformation;
Fig. 10 illustrates an exemplary main application window for the SGML to HTML mapping and transformation;
Fig. 11 illustrates exemplary dialog boxes for opening and saving a file;
Fig. 12A illustrates an exemplary window for the Map Processing Option of the SGML to HTML mapping and transformation;
Fig. 12B illustrates an exemplary window for the SGML to HTML Map Editor;
Fig. 12C illustrates an exemplary window for the SGML to HTML Map Editor with sample data displayed in exemplary dialog windows;
Fig. 13 illustrates a class diagram for the Menu Manager for the SGML to HTML mapping and transformation;
Fig. 14 illustrates an object message diagram for startup of the system of the SGML to HTML mapping and transformation;
Fig. 15 illustrates an object message diagram for opening an SGML document for the first time;
Fig. 16 illustrates an object message diagram for opening a new SGML document;
Fig. 17 illustrates the design of the GUI (Graphical User Interface) for the SGML to HTML mapping and transformation;
Figs. 18A(1)-18A(3) illustrate, in object message diagram format, the behavior among the objects of the classes for editing a map for the SGML to HTML mapping and transformation;
Figs. 18B(1)-18C(3) illustrate, in object message diagram format, the behavior of the objects of the classes for assigning values to HTML attributes;
Fig. 19 illustrates a hardware configuration for implementation of the SGML to HTML mapping and transformation;
Fig. 20A illustrates an exemplary public identifier in ISO/IEC 9070 format;
Fig. 20B illustrates an exemplary mapping of ISO/IEC 9070 to a UNIX file name format;
Fig. 20C illustrates an exemplary UNIX file name resulting from mapping the public identifier of Fig. 20A using the map of Fig. 20B;
Fig. 20D illustrates an exemplary user interface display for mapping a public identifier in ISO/IEC 9070 format to a UNIX file name format;
Fig. 20E illustrates an exemplary user interface display for mapping a registered owner field in ISO/IEC 9070 format to a UNIX file name format;
Fig. 20F illustrates an exemplary user interface for selections for a character mapping of a prefix, owner-name component separator in ISO/IEC 9070 format to the UNIX file name format;
Fig. 20G illustrates an exemplary user interface for mapping an owner name character in ISO/IEC 9070 format to valid characters of the UNIX file name format; and
Fig. 20H illustrates an exemplary user interface for a user to map a registered owner component in ISO/IEC 9070 format to a UNIX file name format.

BRIEF DESCRIPTION OF THE APPENDICES

[0040]

Appendix A is an exemplary Standard Generalized Markup Language ("SGML") Document Type Definition ("DTD") corresponding to the tree structure or Fig. 3A;
Appendix B is an exemplary map of SGML elements from the SGML DTD of Appendix A to HTML elements to produce documents which correspond to the tree structure of Fig. 3B;
Appendix C is an exemplary SGML document which conforms to the SGML DTD of Appendix A;
Appendix D is an HTML document which is generated by using the map of Appendix B to transform the SGML document of Appendix C into HTML elements.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0041] Referring now to the drawings, wherein like reference numerals designate identical or corresponding parts throughout the several views, and more particularly to Fig. 1A thereof, there is illustrated an exemplary Standard Generalized Markup Language ("SGML") Document Type Definition ("DTD"). Figs. 1A-1D are presented to illustrate sample inputs and outputs of the SGML to HTML mapping and transformation. The functions performed during the transformation which generate the outputs are described in detail below, and with respect to Figs. 4-6B and the object message diagrams of Figs. 18A(1)-18C(3).

[0042] Figs. 1A-1D and Appendices A-D show exemplary SGML DTDs, SGML documents, maps, and HTML documents produced from transforming the SGML documents using the maps. Figs. 3A-3B show exemplary tree structures for the SGML DTD of Appendix A and the HTML structure resulting from transforming the SGML DTD using the map of Appendix B. Figs. 1A-1D, Appendices A-D, and Figs. 3A-3B illustrate mapping SGML DTDs to HTML DTDs. The problem of DTD to DTD mapping is the default mapping from an SGML instance to an HTML instance based upon the DTDs. SGML tags are either mapped to zero or more HTML tags, and the sources of HTML attributes must be specified in the mapping. A more concise mathematical expression of the problem is given below.

[0043] Let SS be the space generated by the SGML DTD where SS $=\{S_i \mid i=0,...,m\}$, $S_i=$⟨Tag Name$_i$, Attribute Set$_i$⟩, and Attribute Set$_i$=(Attribute$_j$ of Tag Name$_i$ $\mid$ j=0,...,n_i}+ϕ. Similarly, let HH be the space generated by the HTML DTD specified by W3C (world Wide Web Consortium), that is, HH=$\{H_i \mid i=0,...,k\}$ where H_i corresponds to S_i above. Further, let HG be the space generated by HH consisting of the set of ordered members of HH. Then, HG={null, ⟨H_0⟩, ⟨H_1⟩,..., ⟨H_0, H_1⟩, ⟨H_0, H_2⟩,..., ⟨H_1, H_0⟩,...}. The sequence of legal HTML tags to be mapped are likely to be found in HG. Then the SGML tag to HTML tag mapping is equivalent to the function F : SS → HG + {not-assigned}.

[0044] For purposes of this discussion, {} denotes a set, ⟨...⟩ denotes an ordered set, and + denotes union.

[0045] Let HGG be a set generated from SS, F(SS), and HH. HGG consists of the ordered set of triplets or null. A triplet consists of S_i, F(S_i) and ⟨HTMLTagName, an Attribute⟩ where HTMLTagName belongs to one of H_j in F(S_i). Assume H_0 has $Attr_0$ and $Attr_1$, H_6 has $Attr_0$, and S_0 and S_1 are mapped to ⟨H_0⟩ and ⟨H_3, H_6⟩, respectively. Then HGG={null, ⟨S_0, ⟨H_0⟩, ⟨H_0Tag Name, $Attr_0$⟩⟩, ⟨S_0, ⟨H_0⟩, ⟨H_0Tag Name, $Attr_1$⟩⟩,...,⟨S_1, ⟨H_3, H_6⟩, ⟨H_6 Tag Name, $Attr_0$⟩⟩,...}. Also, let SAtr be the source of the HTML Attribute value. Then Satr = AS+AC+{user inputs}+Null, where AS=the set of ordered pairs of tag name and one attribute name, and AC=the set of tag names with character data content. Then the identification of the attribute source is a function G mapping from HGG to Satr, denoted G: HGG→SAtr.

[0046] A complete description of SGML is provided in Goldfarb, C. F., *The SGML Handbook*, Oxford University Press, Oxford, 1990, and McGrath, S., *Parseme.1st: SGML for Software Developers*, Prentice Hall PTR, New Jersey, 1998, which are incorporated herein by reference.

[0047] The exemplary SGML document of Fig. 1C, together with the exemplary SGML DTD of Fig. 1A, and the exemplary mapping of Fig. 1B are utilized in a transformation process to generate the HTML document of Fig. 1D. The SGML DTD of Fig. 1A and the SGML document of Fig. 1C are together parsed to produce the structural components of the SGML document of Fig. 1C. These components are then utilized in conjunction with the map of Fig. 1B to transform the SGML document of Fig. 1C into the HTML document of Fig. 1D. Further details of the parsing and transformation are explained below, and with respect to Figs. 4-6B and Figs. 18A(1)-18C(3).

[0048] In Fig. 1A, line 22 is a comment line containing the name of the SGML DTD file. Line 24 is a declaration of an element t containing a model group including elements t1, t2, t3, and t4. The '?' of line 24 indicates that an element is optional. The '*' of line 24 indicates that the model group may occur any number of times in a valid element t, and may also be absent. For this example, the SGML document of Fig. 1C illustrates a valid element t containing elements t1, t2, t3, and t4 in a group in lines 64-70.

[0049] Line 26 of Fig. 1A is a declaration of an element t1 having content type CDATA. The type CDATA means that the element may have a value that consists of general characters. For this example, the SGML document of Fig. 1C includes an element t1 on line 64 having as content the string of general characters '1. Hi Larry'. Usually, content of an element is delimited by a start tag for the element before the content, and an end tag for the element after the content. A start tag typically includes the character '(' followed by the name of the element, followed by optional element information such as attribute information, followed by ')'. An end tag then includes the characters '(/' followed by the name of the element, followed by ')'. In SGML, the delimiters '(' and ')' can, by definition, be replaced by other characters.

[0050] Line 28 of Fig. 1A is a declaration for an attribute list for the element t1. An attribute is a property of an element that takes on different values for different instances of elements. For example, an element 'person' typically has an attribute list of attributes 'name', 'age', and 'haircolor'. A particular first person has name="Joe Smith", age ="27", and haircolor="brown", while a second person has name="Sally Jones", age="45", and haircolor="red". If the second person desires, her haircolor attribute is easily changed to haircolor="blond" by an assignment of a different value. For the example of Fig. 1A, on line 28, the attribute list includes an attribute 'name', of type CDATA. The character string '#REQUIRED' is an attribute value indicating that the attribute must be specified. For this example, in the SGML document of Fig. 1C the element t1 on line 64 has a general character content value of "hilarry" assigned to the name

attribute of this element t1.

[0051] Line 30 of Fig. 1A is a declaration for an element t2, of type CDATA. Line 32 is a declaration of an element t3, of type CDATA. Line 34 is a declaration of an element t4, of type CDATA.

[0052] In the SGML to HTML mapping of Fig. 1B, line 42 illustrates a mapping rule of the element t of line 24 to a string of HTML tags and text including '⟨html⟩⟨title⟩Title⟨/title⟩'. Line 44 illustrates a mapping rule of the element t1 of line 26 to a string of HTML tags '⟨H3⟩⟨A NAME="the source is t1's name"⟩'. The sentence between the double quotes of line 44 is a rule rather than an attribute value. Line 46 illustrates a mapping rule of the element t2 of line 30 to an HTML tag '⟨P⟩'. Line 48 illustrates a mapping rule of the element t3 of line 32 to an HTML tag '⟨P⟩'. Line 50 illustrates a mapping rule of the element t4 of line 34 to a string of HTML tags '⟨P⟩⟨A HREF="# the source is t1's name"⟩'. The sentence between the double quotes of line 50 is a rule rather than an attribute value.

[0053] Referring to the exemplary SGML document of Fig. 1C, line 60 shows the document type of the SGML document to be 't', with the DTD corresponding to the SGML document found in the system file "sample1.dtd". Line 62 contains the document start tag '⟨t⟩', which indicates that the lines following line 62 are assumed to follow the format defined in the DTD of Fig. 1A for the element t of line 24. Lines 64, 66, 68, and 70 are exemplary constituent parts of the element t shown on line 24 of Fig. 1A, defined for the exemplary SGML document of Fig. 1C. Line 72 contains the document end tag '⟨/t⟩', signifying the end of the document.

[0054] The HTML document of Fig. 1D is the output of the transformation process utilizing the SGML DTD file of Fig. 1A, the mapping of Fig. 1B, and the SGML document of Fig. 1C. Lines 82, 84, 86, 88, 90, 92, and 94 of Fig. 1D are generated from the information contained in each of Figs. 1A-1C.

[0055] The processing of the exemplary input files illustrated in Figs. 1A-1C to produce the output document illustrated in Fig. 1D will now be described. First, the SGML document line 60 of Fig. 1C is analyzed to determine the document type of the input SGML document and the name of the system file where the SGML documents DTD is stored. This causes the transformer to output the DOCTYPE HTML tag illustrated in line 80 of Fig. 1D, and to open the referenced system file for accessing the DTD for the current SGML document. A check is performed to determine that the DTD does correspond with the current SGML document. Next, the SGML tag of line 62 is parsed so that the current SGML tag becomes '⟨t⟩'. The map of Fig. 1B is referenced to determine that the current SGML tag is the start tag for the current SGML document, and maps to the HTML tag string '⟨html⟩⟨title⟩Title⟨/title⟩'. An HTML tag '⟨html⟩' is saved for the current SGML tag '⟨t⟩' and is output to line 82 of Fig. 1D. The HTML tag substring '⟨title⟩Title⟨/title⟩' is output to line 84 of Fig. 1D.

[0056] The first SGML tag in the string of line 64 of Fig. 1C is now obtained. The current SGML tag becomes '⟨t1⟩'. The transformer obtains the current attribute name and attribute value for the current SGML tag, obtaining an attribute called 'name' with a value "hilarry". The DTD of Fig. 1A is examined to determine that the SGML tag corresponds to the SGML element defined in line 26 of Fig. 1A, with its corresponding attribute list established in line 28 of Fig. 1A. The map of Fig. 1B is analyzed to determine that the current SGML tag's rule is line 44. The mapped HTML string '⟨H3⟩⟨A NAME=' followed by the current value of the name attribute for the SGML element t1, which is currently "hilarry", is then output to the HTML document on line 86 of Fig. 1D. An '⟩' is then output to terminate the tag. The HTML tags '⟨H3⟩' and '⟨A⟩' are saved for the current SGML tag. Next, the parser recognizes text that will be output to the HTML file on line 86 of Fig. 1D. The parser then recognizes the SGML end tag '⟨/t1⟩', at which point end tags for all the HTML tags currently saved for '⟨t1⟩' are output to the HTML document on line 86 of Fig. 1D in reverse order from which the tags were saved.

[0057] Next, the parser recognizes SGML tag '⟨t2⟩' from line 66 of Fig. 1C. The transformer utilizes the map rule line 46 of Fig. 1B to output HTML tag '⟨P⟩', shown on line 88 of Fig. 1D, and to save the HTML tag for the current SGML tag '⟨t2⟩'. The parser then recognizes text which is output to the HTML file, as shown on line 88 of Fig. 1D. The parser now recognizes SGML end tag '⟨/t2⟩' as terminating the text, at which point an end tag '⟨/P⟩' for the HTML tag currently saved for '⟨t2⟩' is output to the HTML document on line 88 of Fig. 1D.

[0058] The parser now recognizes SGML tag '⟨t3⟩' from line 68 of Fig. 1C. The transformer utilizes the map rule of line 48 of Fig. 1B to output HTML tag '⟨P⟩', shown on line 90 of Fig. 1D, and to save the HTML tag for the current SGML tag '⟨t3⟩'. The parser now recognizes text which is output to the HTML file, as shown on line 90 of Fig. 1D. Next, the parser recognizes SGML end tag '⟨/t3⟩' as terminating the text, at which point the end tag '⟨/P⟩' for the HTML tag currently saved for SGML tag '⟨t3⟩' is output to the HTML document on line 90 of Fig. 1D.

[0059] Next, the first SGML tag in the string of line 70 of Fig. 1C is obtained. The current SGML tag is now '⟨t4⟩'. The transformer obtains the current attribute name and attribute value for an SGML tag, obtaining an attribute called 'name' with a value "hilarry". The map of Fig. 1B is analyzed to determine that the current SGML tag's rule is line 50. The mapped HTML string '⟨P⟩⟨A HREF="#' followed by the current value of the name attribute, which is currently "hilarry", is then output to the HTML document on line 92 of Fig. 1D. An '"⟩' is then output to terminate the tag. The HTML tags '⟨P⟩' and '⟨A⟩' are saved for the current SGML tag '⟨t4⟩'. Next, the parser recognizes text that will be output to the HTML file on line 92 of Fig. 1D. The parser now recognizes the SGML end tag '⟨/t4⟩', terminating the text, at which point end tags for all the HTML tags currently saved for '⟨t4⟩' are output to the HTML document on line 92 of Fig. 1D in reverse order from which the tags were saved for '⟨t4⟩'.

[0060] Next, the parser recognizes the end of the SGML document by recognizing a '⟨/t⟩' tag of line 72 of Fig. 1C. This is interpreted to indicate an end tag for the SGML tag '⟨t⟩'. The HTML tags saved for the SGML tag '⟨t⟩' are then obtained and an end tag for the HTML tag '⟨/html⟩', the only tag saved for '⟨t⟩', is output to the HTML document as shown on line 94 of Fig. 1D. This terminates the current processing of the documents.

[0061] Fig. 2 shows an output 100 resulting from opening the HTML output document of Fig. 1C as an exemplary What You See Is What You Get ("WYSIWYG") output of a Web browser on a user's computer screen. This output is in a format typically preferred by users of the World Wide Web on the Internet. Users employ Web browser programs to request HTML files, and other file types, from servers on the Internet. The browser downloads a requested HTML file and opens it to display formatted text and images on the user's computer screen. A browser does not have to download a file but is also capable of displaying an HTML file stored local to the computer running the browser or a local area network connected thereto.

[0062] Referring to Fig. 2, the 'Title' line is generated by a browser utilizing line 84 of Fig. 1D. Line 84 includes a start tag '⟨title⟩' and an end tag '⟨/title⟩' to delimit the text of the title to be displayed by the browser, usually in a title bar at the top of the user's computer screen. The '⟨H3⟩' of line 86 of Fig. 1D instructs a Web browser to output non-tag text in a larger, more bold format than normal text output. As the only text appearing after the HTML start tag '⟨H3⟩' is ' 1. Hi Larry', and this is the only text appearing before the HTML end tag '⟨/H3⟩', the text appears on a computer screen enlarged and bold in comparison with the surrounding text. The '⟨P⟩' start tag of line 88 instructs a Web browser to display non-tag text delimited by the start tag and its corresponding end tag in a new paragraph. New paragraphs start on a now line in the output. The end tag '⟨/P⟩' of line 88 instructs a Web browser that this is the end of the current paragraph, and that any non-tag text following this tag will start on a new line on screen.

[0063] On line 86 of Fig. 1D, the tag containing '⟨A name=' is an anchor tag. Anchor tags are used to set place markers in text, and to establish highlighted text that can be clicked on with a mouse to cause a jump to the text containing the place marker. For this tag, a place marker is established for the browser in the non-tag text following the next '⟩' until the '⟨/A⟩' end tag is encountered. Line 92 contains another type of anchor tag containing '⟨A HREF ='. The text appearing on line 92 between the anchor tag's '⟩' and its corresponding end tag '⟨/A⟩' appears on the screen of Fig. 2 as underlined text '⟨Back to the Hi Larry greeting.⟩'. This text is typically displayed in a different color from the surrounding text on the screen. When a user clicks a mouse on this underlined text, the text marked by the reference anchor tag of line 86 is pulled in for the user's viewing, surrounded by its neighboring text.

[0064] Fig. 3A and Fig. 3B illustrate the transformation of a hierarchical SGML document tree structure to the more "flat" tree structure of an HTML document. Fig. 3A illustrates a hierarchical SGML document tree structure, whereas Fig. 3B illustrates the corresponding "more flat" tree structure of the HTML document corresponding to the SGML document graphically. displayed in Fig. 3A.

[0065] The trees of Fig 3A and Fig 3B are derived from documents illustrated in Appendices A-D. Appendix A shows an exemplary SGML DTD. The tree structure of Fig 3A is derived from the SGML DTD of Appendix A. The tree of Fig 3A has a root node test 110 which has children nodes front 112 and section 114. The node front 112 has children nodes title 116, author 118, and keywords 120. The node section 114 has children nodes number 122, title 124, para 126, and subsec 1 128. The node author 118 has children nodes fname 130, surname 132, and title 134. The node subsec 1 128 has children nodes number 136, title 138, para 140, and subsec 2 142. The node subsec 2 142 has children nodes number 144, title 146, and para 148. The tree structure of Fig. 3A which corresponds to the SGML DTD of Appendix A has five levels and twenty nodes.

[0066] The tree structure of Fig. 3B corresponds to a generalized HTML document that results from utilizing the SGML DTD of Appendix A and a mapping exemplified in Appendix B. Appendix C shows an exemplary SGML document to be processed through the mapping shown in Appendix B to give an HTML document exemplified in Appendix D. The tree structure of Fig. 3B has a root node html 150 having two children nodes, head 152 and body 154. The head node 152 has a child node title 156. The body node 154 has children h3 158 and p 160. The node p 160 has a child strong 162. In contrast to the tree structure of Fig 3A which has five levels and twenty nodes, the tree structure corresponding to the resulting HTML document has only four levels and seven nodes.

[0067] Fig. 4 illustrates an overview of major modules of the SGML to HTML mapping and transformation. A Map Module 184 interacts with a Parser 182 and a GUI 180 to create the actual mapping from an SGML document to an HTML document. A Transformer 186 interacts with the Map Module 184, the Parser 182, and the GUI 180 to transform the SGML document into the HTML document. A Service module 188 contains utility objects which can be utilized by all the modules for utility processing such as file handling. The GUI 180 handles interaction between a user and the system. The Parser 182 analyzes and breaks down input documents into recognizable component parts to be passed to other modules of the system. For example, in processing the exemplary SGML document of Fig. 1C, Parser 182 analyzes the input SGML document recognizing a DTD to generate a symbol table which can be passed to other modules of the system for processing documents. The Parser 182 recognizes line 60 of Fig. 1C as a 'DOCTYPE' tag and processes a DTD specified by a system file "sample.dtd" shown in Fig. 1A to generate the symbol table. The Parser 182 would then recognize line 62 of Fig. 1C contents as a start tag for the element t, and would transmit the tag and other

tag information to Transformer 186. Transformer 186 controls the processing of the SGML to HTML mapping and transformation, requesting information and data from the Map Module 184, the Parser 182, and the Service 188 modules when needed.

[0068] Fig. 5 illustrates a data flow diagram showing the flow of data through the SGML to HTML mapping and transformation. An SGML Application DTD 200 and HTML 3.2 DTD 204 are input to an SGML Parser 202. This SGML Parser 202 corresponds to the Parser 182 of Fig. 4. An SGML Application Symbol Table 222 and an HTML 3.2 Symbol Table 206 are output from the SGML Parser 202 to be utilized as input to a Map Editor 208, along with an interactive User 210 input. The Map Editor 208 is contained within the GUI 180 of Fig. 4. The Map Editor 208 outputs a Map 212. The SGML Application Symbol Table 222 and an SGML Application Document Instance 218 are together input to the SGML Parser 202 to give output to be used as input, along with the Map 212, to the Transformer 186. Transformer 186 corresponds to the Transformer 186 of Fig. 4. Transformer 186 then outputs an HTML Document Instance 216. The SGML Application DTD 200 and SGML Application Document Instance 218 are exemplified in Fig. 1A and Fig. 1C, respectively. The HTML Document Instance 216 is exemplified in Fig. 1D. The Map 212 is exemplified in Fig. 1B.

[0069] Fig. 6A is a more generalized data flow diagram showing exemplary paths taken by data flowing through the generalized mapping and transformation of information in one structured format to information in another structured format. A Structural Description of System A 230, together with a Structural Description of System B 232 and interactive User 210 input, are input to a Map Editor 208 to output the Map 212. The Map 212 and an Instance of System A 238 are then utilized by the Transformer 186 to output an Instance of System B 244.

[0070] Fig. 6B is a more generalized data flow diagram showing exemplary paths taken by data flowing through the SGML to HTML mapping and transformation. An SGML DTD 200, together with an SGML Document 218 and an HTML DTD 260, are input to the Mapping Editor 208 to output the Map 212. The Map 212 and the SGML Document 218 are then utilized by the Transformer 186 to output an HTML Document 216. The HTML Document 216 corresponds to the HTML Document Instance 216 of Fig. 5. The HTML Document 216 is then input to a Browser 262 for user viewing.

[0071] The SGML DTD 200, input to an SGML Editor 256, yields output to the SGML Document 218. A Database Design 250, input to the Mapping Editor 208, along with the SGML DTD 200 and the SGML Document 218, yield output to the Map 212 and a Data Base 254. The Map 212 and the SGML Document 218 are input to the Transformer 186 to yield output, which, together with the output from the Mapping Editor 208, are input to the Data Base 254. The Map 212 corresponds to the Map 212 of Fig. 5. The SGML Document 218 corresponds to the SGML Application Document Instance 218 of Fig. 5. The SGML DTD 200 corresponds to the SGML Application DTD 200 of Fig. 5. The Transformer 186 corresponds to the Transformer 186 of Fig. 5. The Mapping Editor 208 corresponds to the Map Editor 208 of Fig. 5. Arrows illustrate different paths the documents and data files take for different requests of a user.

[0072] Fig. 7 shows a hierarchical view of the DTD Map class object that can be stored in a file. The invention has been implemented using object oriented techniques, although any programming technique and/or hardware may be used to implement the invention. For purposes of this description, a class is a description of the structure and behavior of an object, while an object is an instance of the item described by a class. Objects typically communicate by passing objects and messages to each other. In structure, objects contain other objects or structures as components, as well as variables and methods.

[0073] Interpreting the horizontal lines of Fig. 7 from left to right, begin and end delimiters delimit each object in the file. List Begin and List End delimiters delimit lists from left to right. When a DTD Map Object exists, one or more SGML tag objects are placed between the SGML Tag List Begin 363 and SGML Tag List End 365.

[0074] The file begins with a Header 360 followed by a DTD Map 361. The DTD Map 361 includes, first, a DTD Map Begin 362 followed by an SGML Tag List Begin 363, followed by at least one SGML Tag 364-1 through an SGML Tag 364-n. The sequence of one or more SGML tags is followed by an SGML Tag List End 365, followed by a DTD Map End 366. Each SGML Tag 364-1 through 364-n includes an SGML Tag Begin 367, an SGML Tag Name 368, followed by an SGML Tag Empty State 369, followed by an SGML Tag Assignment Type 370, followed by an HTML Tag List 371, followed by an SGML Tag End 372.

[0075] Each HTML Tag List 371 is delimited by an HTML Tag List Begin 373 at the beginning and an HTML Tag List End 375 at the end with the list including at least one HTML Tag 374-1 through HTML Tag 374-m following the HTML Tag List Begin 373.

[0076] Each HTML Tag 374-1 through 374-m is delimited by an HTML Tag Begin 376 at the beginning and an HTML Tag End 380 at the end. Following the HTML Tag Begin 376 is an HTML Tag Name 377, followed by an HTML Tag Empty State 378, followed by an HTML Attribute List 379, followed by a delimiter HTML Tag End 380.

[0077] Each HTML Attribute List 379 is delimited by an HTML Attribute List Begin 381 at the beginning and an HTML Attribute List End 383 at the end. Following the delimiter HTML Attribute List Begin 381 is at least one HTML Attribute 382-1 through an HTML Attribute 382-P, followed by an ending delimiter HTML Attribute List End 383.

[0078] Each HTML Attribute 382-1 through 382-p is delimited by an HTML Attribute Begin 384 at the beginning and an HTML Attribute End 389 at the end. Following the HTML Attribute Begin 384 is an HTML Attribute Name 385, followed by an HTML Attribute Source Type 386, followed by an HTML Attribute Source 1 387, followed by an HTML

Attribute Source 2 388, if one exists. The delimiter HTML Attribute End 389 terminates the listing of contents.

[0079] Fig. 8A shows major class dependencies 424 for the DTD Map object. For purposes of explanation of the figures that follow, arrows show class dependencies, meaning that an object having an arrow pointing to it is contained within the object originating the arrow. A Map object 400 includes a pointer to an object DTDMap 402. A pointer is a value that represents an absolute address of an item in computer memory. A pointer to an object is used to access the information stored for the implementation of a particular object by, minimally, referencing the pointer name and the field or function name within the object. Viewing Fig. 7 and Fig. 8A together, the DTDMap 402 of Fig. 8A, corresponding to the DTD Map 361 of Fig. 7, includes, via pointers, an SGMLTagList 404 corresponding to the SGML Tag 364-1 through 364-n of Fig. 7. The SGMLTagList 404 of Fig. 8A includes, via pointers, a class SGMLTag 406 corresponding to each of the SGML Tags 364-1 through 364-n of Fig. 7. The SGMLTag class 406 of Fig. 8A includes, via pointers, HTMLTagList 408, which corresponds to the HTML Tag List 371 of Fig. 7. The HTMLTagList 408 of Fig. 8A includes, via pointers, an HTMLTag 410 which corresponds to each of the HTML Tag 374-1 through HTML Tag 374-m of Fig. 7. The HTMLTag 410 of Fig. 8A includes, via pointers, an HTMLAttrList 412 class which corresponds to the HTML Attribute List 379 of Fig. 7. The HTMLAttrList 412 of Fig. 8A includes, via pointers, an HTMLAttr 414 class which corresponds to the HTML Attributes 382-1 through 382-p of Fig. 7. The HTMLAttr 414 class of Fig. 8A includes, via pointers, a class derived from an abstract class, denoted by an 'A' inside a triangle, HTMLAttrSource 416 which corresponds to the HTML Attribute Source Type 386 of Fig. 7. An abstract class is typically defined as a model to be used for defining other closely related classes which may, for example, need to exhibit similar behavior in a system. By defining an abstract class, the other classes are defined as inheriting the structure and methods of the parent abstract class. The hollow arrows of Fig. 8A denote inheritance of classes. HTML attributes may be obtained from different sources. Therefore, a UserInput 418 class is shown to inherit from the abstract class HTMLAttrSource 416. Also, an SGMLContent 422 class inherits from HTMLAttrSource 416, as does an SGMLTagAttr 420 class.

[0080] Fig. 8B shows major classes within the Map Module 276, illustrating the major dependencies among the classes. The classes 424 illustrated inside the dashed line rectangle are the classes 424 of Fig. 8A. A class MapService 452 includes a class Transformer 450, a class DTDMapTransformerService 456, a class SrcSGMLTagAttr 462, a class SrcSGMLContent 464, a class Map 400, a class MapEdit 460, and a class MapCreateEditService 454. The class MapEdit 460 includes the class DTDMapEdit 466. The class DTDMapEdit 466 has dependencies with the class DTDMap 402, the class SGMLTagList 404, SGMLTag 405, HTMLTagList 408, HTMLAttrList 412, HTMLTag 410, HTMLAttr 414, SGMLTagAttr 420, SGMLContent 422, and UserInput 418. The class HTMLAttrSource 416 has dependencies with the class SrcSGMLTagAttr 462 and the class SrcSGMLContent 464. The class DTDMapTransformerService 456 has dependencies with the class SrcSGMLTagAttr 462 and the class SrcSGMLContent 464. The class SrcSGMLTagAttr 462 has a dependency with the class SGMLTagAttr 420 and the class SrcSGMLContent 464 has a dependency with SGMLContent 422. The class MapCreateEditService 454 has dependencies with the class DTDMapEdit 466 and the class MapEdit 460. The class Map 400 contains DTDMap 402, DTDMapTransformerService 456, SrcSGMLTagAttr 462, and the class SrcSGMLContent 464. The functionalities of these classes and their objects are explained with regard to Figs. 18A(1)-18C(3).

[0081] Data items and objects in software generally involve dynamic allocation of computer storage resources at some stage in a request for execution of program code. Pointer variables, containing addresses of data items, methods, or objects, are available to be passed among objects during execution of code. As objects and data items are constructed and destructed dynamically, an object using a pointer or reference to a data item, for example, may reference the item after it has been destructed, possibly causing a system failure. A facility for registering objects and data items as they are created and requested gives objects a means to verify the current existence and usage of objects and data items before reference. A destructor verifies the current usage of an object or data item before destruction so that other objects using the object or data item may successfully complete their usage before destruction. An exemplary use of registering objects and data items is assignment of attribute values to HTML attributes, as discussed below with regard to Figs. 8C(1)-8C(4) and Figs. 18A(1)-18C(3).

[0082] Fig. 8C(1) illustrates a class structure for a SrcSGMLTagAttr 462 class of the SGML to HTML mapping and transformation. Fig. 8C(2) illustrates a class structure for a SrcSGMLContent 464 class of the SGML to HTML mapping and transformation. Fig. 8C(3) illustrates a class structure for the MapService 452 class of the SGML to HTML mapping and transformation. As described previously with regard to Fig. 8B, the class HTMLAttrSource 416 includes references to the class SrcSGMLTagAttr 462 and the class SrcSGMLContent 464. The HTMLAttrSource 416 contains an AtrSrcSGMLTagAttr, which is a reference to a SrcSGMLTagAttr 462, and an AtrSrcSGMLContent, which is a reference to a SrcSGMLContent 464. SrcSGMLTagAttr 462 contains a method registerSGMLTagNameAndAttributeName, used for registering SGML tag attributes in the MapService 452, so that attributes that have already been registered are available to be unregistered from the HTMLAttrSource 416 by using a method unregisterTagAttrKeyAndMapEntry in SrcSGMLTagAttr 462 through a virtual function unregisterFromSourceMap. SrcSGMLTagAttr 462 also contains a method setValueForAttributeOfTag to be used at document instance processing time to transform an SGML document to an SGML document.

[0083] SrcSGMLContent 464 contains a method registerSGMLTagName, used for registering SGML tag content in the SrcSGMLContent 464, so that attributes that have already been registered are available to be unregistered from the HTMLAttrSource 416 by using a method unregisterSGMLTagName in SrcSGMLContent 464. SrcSGMLContent 464 also contains a method setValueForTag to be used at document instance processing time to transform an SGML document to an HTML document.

[0084] Fig. 8C(4) illustrates a class structure for the MapCreateEditService 454 class of the SGML to HTML mapping and transformation. MapCreateEditService 454 was discussed previously with regard to Fig. 8B, and functionalities of the class and object are explained with regard to Figs. 18A(1)-18C(3). A method setSelectedSGMLTagToBeNullAssigned in MapCreateEditService 454 sets a selected SGML tag to be mapped to a null value. A method setSelectedSGMLTagToBeNotAssigned in MapCreateEditService 454 sets a selected SGML tag to be kept in the mapping. A method getAttributeAssignmentInformationForHTMLAttribute in MapCreateEditService 454 gets assignment information for assigned values to HTML attributes. Methods assignHTMLAttributeWithSGMLAttribute, assignHTMLAttributeWithSGMLContent, assignHTMLAttributeWithSystem, assignHTMLAttributeWithNoValue, assignHTMLAttributeWithUserInput in MapCreateEditService 454 assign values to the HTML attributes.

[0085] Fig. 9 illustrates a hierarchical view of major modules for implementation of the GUI for the SGML to HTML mapping and transformation. An Application Window 470 initiates execution of a Menu Manager 472. The Menu Manager 472 initiates execution of a File Service 474, Editor for Map 476, View 478, Map 480 and Message Dialog Service 482. The Map 480 module corresponds to the Map class 400 of Fig. 8A. The View 478 module is included in the GUI 270 of Fig. 4. The functionalities of these modules are explained with regard to Figs. 18A(1)-18C(3).

[0086] Fig. 10 shows an exemplary computer screen output of a Main Application Window 510. The window includes a title bar 512, a menu bar 514, a tool bar 516, and a viewing window work space 518. In order for a user to perform any menu operation, the user must select one of the items in the menu bar 514. When the user makes a selection, a sub-menu (or pull-down menu) displays all operations available for selection from the main menu. For example, if the user selected the File option, a sub-menu appears to display the options Open SGML, Open DTD, Open Map, Save Map, Map Save As, Save HTML, HTML Save As, Close SGML, and Exit. The Open SGML option allows the user to select an SGML document to open. The Open DTD option allows the user to select a DTD to open. The Open Map option allows the user to select a map to open. The Save Map Option allows the user to save a map onto disk. The Map Save As option allows the user to save the map onto the disk with the option of selecting a new file name for the saved map. The Save HTML option allows the user to save an HTML document onto disk. The HTML Save As option allows the user to save an HTML document onto disk with the option of selecting a new file name for the HTML file. The Close SGML option allows the user to close an SGML document and Exit option allows the user to exit the Application Window processing of the conversion.

[0087] If the user selects an Edit option from the menu bar 514, a sub-menu appears to display options Create Map and Edit Map. The Create Map option allows the user to create a map that can be used, to transform an SGML document to an HTML document. The Edit Map option allows the user to modify an existing map. A sub-menu for a View option displays the options to View SGML and to View HTML. These options are designed to display an SGML document or an HTML document when selected by a user. A View SGML option allows the user to display an SGML document in the work space of the main application window. A View HTML option allows the user to display an HTML document in the work space of the main application window. If the user selects the Map option, a sub-menu appears to display an option Run Map. Selecting Run Map initiates the transformation to transform an input SGML document to an HTML document. If the user selects the Option button, a sub-menu appears to display options of Incremental and Reference. An Incremental option allows the user to perform an incremental mapping of an SGML document to an HTML document. After an SGML tag is mapped into HTML tags the SGML document is then transformed into its corresponding HTML document. This occurs after each SGML tag is mapped. A Reference option allows the user to display the reference in transforming an SGML document to an HTML document.

[0088] Referring to Fig. 11, dialog boxes for opening an SGML file and saving an HTML file are shown. Fig. 11 shows an exemplary file open dialog box 600 which would be displayed responding to an Open SGML option from the sub-menu displayed after the user selected the File option in Fig. 10. A Filter text edit box 602 is displayed allowing the user to choose a type of file to be opened. Candidate directories for files are displayed in a Directories list box 604. Candidate files for opening will be displayed in a Files list box 606. A Selection text edit box 608 displays the file name that the user selects for opening. An OK button 610 allows the user to approve the selection shown in the Selection text edit box 608. A Filter button 612 allows the user to request a display of all files of a given type, in the Files list box 606, as described in the Filter text edit box 602. A Cancel button 614 allows the user to choose termination and cancellation of the current request to open a file.

[0089] Fig. 11 also shows an exemplary Save HTML file dialog box 616 that is displayed as a result of the user selecting the Save HTML option from the sub-menu displayed after the user selected the File button in Fig. 10. A Filter text edit box 618 allows the user to select a type of file for saving the current HTML file. Candidate directories for files are displayed in a Directories list box 620. Candidate files of a given type are displayed in a Files list box 622. A Selection

text edit box 624 allows the user to select a file name for saving the file. An OK button 626 allows the user to approve an operation and complete the operation of saving a file. A Filter button 628 allows the user to request a display of all files of a given type, in the Files list box 622, as described in the Filter text edit box 618. A Cancel button 630 allows the user to terminate and cancel the current request to save the current HTML file.

[0090] Fig. 12A and Fig. 12B display exemplary Map Edit Option and Map Edit Dialog boxes utilized for creation and editing of a map. The user is allowed to create a new map or edit an already existing map. If the user selects the Edit button from the Main Application Window 510 of Fig. 10, and either the Edit Map or Create Map option is selected, the Map Edit Option 690 dialog box of Fig. 12A is displayed to allow the user to select whether the Default mapping of the SGML document or a Conditional mapping of the SGML document should be used to create or edit the map. The Default mapping, selected by clicking on a Default button 692, is the user defined tag mapping set up by the user interaction with the SGML to HTML Map Edit dialog box 700 of Fig. 12B. The Conditional mapping, selected by clicking on a Conditional button 694, involves defining the conditional or special mappings. After the user selects one of the options from the Map Edit Option 690, then the Map Edit dialog box 700 of Fig. 12B is displayed to allow the user to interact with the system in defining a map. If the Create Map option is selected, the user is allowed to create a new map. Both the Map Edit Option dialog box 690 and the Map Edit dialog box 700 are used for creating a map and for editing an existing map.

[0091] Referring to Fig. 12B, a display of the Map Edit dialog box 700 shows a display of a list of SGML Tags 702, the current HTML Tag list 704 that an SGML tag selected from the SGML Tag list 702 maps to, and a list of Legal HTML Tags 706 that can be added into the current HTML Tag list 704. For a given SGML Tag 702, the user selects the Legal HTML Tag 706 by double clicking a mouse on an HTML tag from the Legal HTML Tag list 706. This will add the HTML Tag to the Current HTML Tag list 704. If the Current HTML Tag list 704 contains a list of HTML Tags that the SGML Tag 702 maps into, a new HTML Tag will be added below the HTML Tag that is selected in the Current HTML Tag 704 list. If an HTML Tag is inserted into the current HTML Tag 704 list, the HTML Tag(s) following the inserted Tag must be deleted, as they may no longer be legal. A Clear HTML Tag 708 button will clear the current HTML Tag 704 list. A Delete HTML Tag 710 button will delete the HTML Tag selected in the Current HTML Tag 704 list. The HTML Tags following the deleted HTML Tag in the Current HTML Tag 704 list must be deleted since they may no longer be legal. An Undo 712 button will undo the last clear, delete or insert operation. These buttons are easily modified to a menu operation format by one skilled in the art of computing. A Map SGML Tag 714 button allows the user to map the SGML Tag 702 to the HTML Tag list in the Current HTML Tag 704 list and then allows the user to select the next SGML tag to map. If a Done 716 button is selected, the remaining SGML Tags 702 will not be mapped. If a Cancel 718 button is selected, all previous SGML to HTML map information will be disregarded. Two possible selections in the Legal HTML Tag list 706 are Null Assigned and Not Assigned. Null Assigned deletes the SGML Tag 702 so that the SGML tag 702 will not be mapped and will not be displayed after transformation. Not Assigned leaves the SGML Tag 702 as is, so that the SGML Tag 702 will not be mapped to HTML but will be displayed as is after transformation.

[0092] An explanation of tag attribute assignment is provided with regard to Figs. 18A(1)-18C(3).

[0093] Fig. 12C shows exemplary data in the tag list boxes of the Map Edit dialog box 700 previously discussed for Fig. 12B. An explanation of the processing of the data is provided with regard to Figs. 18A(1)-18C(3).

[0094] Fig. 13 illustrates an exemplary class diagram displaying relationships among the classes of the SGML to HTML mapping and transformation for the GUI. An Application Window 772 manages the handling of the display of the application window of the GUI. A Menu Manager 778 handles all the tasks and objects associated with menu operations. A File Service 782 handles open and save operations associated with files. An ntEntity 790 is a general system representation of an SGML document, an SGML DTD, an HTML document, or an HTML DTD. A Symbol Table 770 is the system representation of an input document after it has been processed by a Parser Service 774. A MessageDialogService 776 handles the output of messages to the system used. A View Document 786 class handles the display of SGML or HTML documents upon user request. A Map Service 780 handles the creation and editing, through a MapCreateEditService 792, of a Map 788, which is the system representation of the rules to be utilized in the transformation of an SGML document to an HTML document. A GUIEditorForMap 784 handles the GUI interface for the user to dynamically create or edit the Map 788.

[0095] Fig. 14 shows an object message diagram displaying the behavior of the system among objects of the classes for the startup of the system. The object diagram illustrates major software objects enclosed in cloud shapes. Object method calls are illustrated with an Arabic numeral preceding a colon, followed by the name of the object method. The numeric ordering illustrated by the Arabic numerals followed by colons illustrate a stepwise logical flow of execution, and a flow of object data and messages, through the diagram. For a more detailed description of object diagrams and design, see Booch, G., *Object-Oriented Analysis and Design with Applications, Second Edition*, Addison Wesley Longman, California, 1994, which is incorporated herein by reference.

[0096] An Application Window 772 is the object which generates the main application window of Fig. 10. It is the first object created when the system starts execution. It contains all the necessary information and functions to display the main application window of Fig. 10. An HTMLSymbolTable 800 is an object, which is the system representation of the

HTML DTD, created by the Application Window 772, in a call Create (new) 802 through ParserService 774. The HTML Symbol Table 800 exists throughout the lifetime of the system. A MenuManager 778 is an object created by the Application Window 772, in a call Create (new) 804, to handle all menu operations associated with the menu of the main Application Window 772. The Application Window 772 passes the HTMLSymbolTable 800 object it created to the MenuManager 778 so that the MenuManager 778 can pass it to any other objects which require it. The MenuManager 778 creates and manages all objects necessary carry to out menu operations.

[0097] A MessageDialogService 776 is an object created by the MenuManager 778, in a call Create (new) 806, to allow message dialog boxes to display any kind of message to the user from anywhere in the system. Exemplary messages are error messages, warnings or instructions to the user, or any other type of message required. The MenuManager 778 passes the MessageDialogService 776 to other objects which may need to display messages to the user.

[0098] A MapService 780 is an object created by the MenuManager 778, in a call Create (new) 808, to handle map related objects. The MenuManager 778 initiates a call MapServiceInit 810 to initialize the state of the newly created MapService 780. For this example, a map is an object which describes how the SGML document will be transformed into an HTML document. The MenuManager 778 passes the HTMLSymbolTable 800 and the MessageDialogService 776 to the MapService 780 so that it has information about the HTML DTD and can send messages to the user.

[0099] A MapCreateEditService 792 is an object created by the MapService 780, in a call Create (new) 812, to handle the creation of a map or the modification of an existing map. The MapService 780 passes the HTMLSymbolTable 800 to the MapCreateEditService 792 so that it has information about an HTML DTD. The MenuManager 778 receives the MapCreateEditService object 792 from Map Service 780, in a call getMapCreateEditServiceObject 814, so that it may create or edit a map at any time.

[0100] A FileService 782 is an object created by the MenuManager 778, in a call Create (new) 816, to handle the tasks of opening and saving a file. The file corresponds to an SGML document, an SGML DTD, a map, or an HTML document. The user requests actions for files by selecting the File button exemplified in the menu bar 514 in Fig. 10. The File Service 782 creates an Open or Save dialog box to allow the user to choose the file the user wants to open or save, as exemplified in Fig. 11. MenuManager 778 passes MessageDialogService 776 to File Service 782 so that it may display messages to the user. The MenuManager 778 also passes the MapCreateEditService 792 to the File Service 782.

[0101] A GUIEditorForMap 784 is an object created by the MenuManager 778, in a call Create (new) 818, to handle the task of allowing the user to create a map or modify an existing map through a dialog box, as exemplified in Figs. 12B-12C. The MenuManager 778 passes the MessageDialogService 776 to the GUIEditorForMap 784 for displaying messages to a user. The MenuManager 778 passes the MapCreateEditService 792 to GUIEditorForMap 784 to create or modify a map.

[0102] A View Document 786 is an object created by the MenuManager 778, in a call Create (new) 820, to handle the task of displaying an SGML or HTML document through a display window. The user requests document viewing by selecting the View button from the menu bar 514 exemplified in Fig. 10. The Menu Manager 778 passes the MessageDialogService 776 to the ViewDocument 786 so that it may display messages to the user.

[0103] Fig. 15 shows an object message diagram to display the dynamic relationships that exist among the objects of the invention when an SGML document is being opened for the first time. A User 830 requests, from an Application Window 772, opening an SGML document file using a call OpenSGML 840. This is accomplished by the user's selection of the File button in the menu bar 514 of Fig. 10, followed by selection of the Open SGML option in the resulting sub-menu. The Application Window 772 sends a call OpenSGML 842 to a MenuManager 778 to process the user request. The MenuManager 778 then sends a call OpenSGML 844 to a FileService 782. The FileService 782 initiates a call getFileName 846 to request a file name for the SGML document from the User 830. A User 830 response, a FileName 846, is returned to the FileService 782. The FileService 782 sends a request isFound 848, accompanied by a FileName 848, to an IOService 832 to determine, by a response YES 848, that the FileName 846 returned by the User 830 exists. The FileService 782 then sends a request isReadable 850, accompanied by a FileName 850, to the IOService 832 to determine, by a response YES 850, that the file is also readable by the system. The FileService 782 then sends a request getEntityObject 852 to IOService 832, accompanied by the FileName 852, so that IOService 832 may obtain and return an ntEntity 852, which is associated with an external entity such as an SGML document, and its corresponding DTD, requested by the User 830.

[0104] The MenuManager 778 sends a request getSGMLNtEntity 854 to the FileService 782 to receive the ntEntity 854 from the FileService 782. The MenuManager 778 then sends the ntEntity 856 to a Parser Service 774 with a call getSymbolTable 856. The Parser Service 774 then generates a SymbolTable 856 for the SGML document, checks that it is a valid symbol table for the SGML document, and returns the SymbolTable 856 to the MenuManager 778. The MenuManager 778 then sends the ntEntity 858 to a MapCreateEditService 792 using a call useThisSGMLDoc 858, and sends the SymbolTable 860 to the MapCreateEditService 792 using a call UseThisSymbolTable 860. The MapCreateEditService 792 then handles further processing of the requested document.

[0105] Fig. 16 shows an object message diagram to display the dynamic relationships that exist among the objects

of the invention as an SGML document as being opened, with an existing SGML document already opened. A User 830 requests, from an Application Window 772, opening an SGML document file by a call OpenSGML 890. This is accomplished by the user's selection of the File button in the menu bar 514 of Fig. 10, followed by selection of the Open SGML option in the resulting sub-menu. The Application Window 772 sends a call OpenSGML 892 to a MenuManager 778 to process the user request. The MenuManager 778 then sends a call closeHTMLWindow 894 to a ViewDocument 786 so that if there is an open window displaying an HTML document, it will be closed. If there is an HTML document that has not been saved, the MenuManager 778 sends a request SaveHTML 896 to a MessageDialogService 776. The MessageDialogService 776 then sends a request SaveHTML 898 to the User 830 as a message asking if the user wishes to save the currently displayed HTML file. A User 830 response of NO 898 is returned to the MessageDialogService 776, which then returns a NO 896 to the MenuManager 778. The MenuManager 778 then sends a call Destroy(delete) 900 to an HTMLntEntity 880, representing the HTML file. The MenuManager 778 then sends a call closeSGMLWindow 902 to the ViewDocument 786 for the ViewDocument 786 to process closing of the display window of an opened SGML document. The MenuManager 778 then sends a call Destroy(delete) 904 to an SGML ntEntity 882, representing an opened SGML file. The MenuManager then sends a call OpenSGML 906 to a FileService 782. The FileService 782 initiates a call getFileName 908 to request a file name for the SGML document from the User 830. The User 830 response, a FileName 908, is returned to the FileService 782. The FileService 782 sends a request isFound 910, accompanied by a FileName 910, to an IOService 832 to determine, via a response YES 910, that the FileName 908 returned by the User 830 exists. The FileService 782 then sends a request isReadable 912, accompanied by a FileName 912, to the IOService 832 to determine, by a response YES 912, that the file is also readable by the system. The FileService 782 then sends a request getEntityObject 914, accompanied by a FileName 914, to the IOService 832, so that the IOService 832 may obtain and return an ntEntity 914. The ntEntity 906 is then returned to the MenuManager 778 in response to the request OpenSGML 906.

[0106] The MenuManager 778 then sends an ntEntity 916 to a Parser Service 774 with a call getSymbolTable 916. The Parser Service 774 then generates a SymbolTable 916 for the SGML document and DTD and returns the SymbolTable 916 to the MenuManager 778. The MenuManager 778 then sends an ntEntity 918 and a Map 918 to a MapService 780 using a call areThisMapAndTheSGMLDocConsistent 918, to determine if the existing Map 918 can be used with the new SGML document, ntEntity 918. A response NO 918 is returned to the MenuManager 778 if the Map 918 cannot be used. The MenuManager 778 then sends a request isMapSaved 920 to the MapService 780, to receive a response of YES 920, if the Map 918 is saved. The MenuManager 778 then sends a call destroy(erase & delete) 922 to an SGMLSymbolTable 884, to destroy the system's current SGML file represented in the format of a symbol table. The MenuManager 778 then sends a call Destroy (delete) 924 to a Map 788 to destroy the system's current map for a previous SGML document. The MenuManager 778 then sends a call resetMap 926 to the MapService 780 to initialize a new map for the new SGML document to be processed. The MenuManager 778 then sends a call useThisSGMLDoc 928 to a MapCreateEditService 792, sending an ntEntity 928 obtained from IOService 832. The MenuManager 778 then sends a call UseThisSymbolTable 930, along with a SymbolTable 930 obtained from ParserService 774, to the MapCreateEditService 792. The MapCreateEditService 792 then handles further processing of the requested SGML document.

[0107] Fig. 17 illustrates, in a class diagram format, the design of the Map Editor GUI. The Application Window 772, the MenuManager 778, the GUIEditorForMap 784, the SymbolTable 770, and the MapCreateEditService 792 have been described previously with regard to Figs. 13-15. In Fig. 17, a MapEditDialog 950 is contained by the GUIEditorForMap 784. The MapEditDialog 950 manages the handling of the display of the Map Edit dialog box. This includes determining what type of operation or requests the user can perform at a given point in the mapping of an SGML element. A MapTag 952 interacts with MapCreateEditService 792 and SymbolTable 770 to handle tasks and objects associated with mapping an SGML element to an HTML element. An AssignAttribute 954 interacts with MapCreateEditService 792 and SymbolTable 770 to handle tasks and objects associated with assigning values to the attributes of HTML elements.

[0108] Figs. 18A(1)-18C(3) are object message diagrams showing the behavior of the SGML to HTML mapping for assigning values to HTML attributes and for mapping SGML tags to HTML tags. As the object diagram of Figs. 18A(1)-18C(3) was large, it was divided over a plurality of drawing sheets. However, these drawing sheets, when taken together, constitute one object diagram. The MapEditDialog 950, the MapTag 952, the AssignAttribute 954, and the MapCreateEditService 792 have been described previously with regard to Fig. 17. The GUIEditorForMap 784 has been described previously with regard to Fig. 13. The HTMLSymbolTable 800 has been described previously with regard to Fig. 14. The SGMLSymbolTable 884 has been described previously with regard to Fig. 16.

[0109] As an exemplary manner of operating, the GUIEditorForMap 784 of Fig. 18A(1) creates the Map Edit dialog box of Fig. 12B by a function call EditMap 960 to call a constructor of the MapEditDialog 950.

[0110] To fill the SGML Tag list box 702 of Fig. 12B, MapEditDialog 950 gets a first SGMLTag 962, with a YES message 962, from MapTag 952 through a function getNextSGMLTag 962, and displays the SGML tag in the SGML Tag list box 702 of Fig. 12B. MapTag 952 gets a first SGMLTag 964 of Fig. 18A(2), with a YES message 964, from the SGML-

SymbolTable 884 through a function call getFirstElement 964. An alternative GUI design is to keep a list of all SGML tags already processed in the display.

[0111] The user selects an SGML tag to map by double clicking the mouse on the SGML tag in the SGML Tag list box 702 of Fig. 12B and the SGMLTag 966 of Fig. 18A(1) is sent to MapTag 952 through a function call selectedSGMLTag 966 of Fig. 18A(1). MapTag 952 of Fig. 18A(2) sends the SGMLTag 967 to MapCreateEditService 792 of Fig. 18A(2) through a function call selectedSGMLTag 967 of Fig. 18A(2).

[0112] To fill the Current HTML Tag list box 704 of Fig. 12B, the MapEditDialog 950 box of Fig. 18A(1) then gets an HTMLTag 968, with a YES message 968, to which the selected SGML tag maps, from MapTag 952 through a function call getNextCurrentHTMLTag 968. The function will get one HTMLTag 968 at a time and display it in the Current HTML Tag list box 704 of Fig. 12B. MapTag 952 of Fig. 18A(2) gets a next HTMLTag 970, along with a YES message 970, from the MapCreateEditService 792 using a function call getExistingHTMLTagsinMap 970. If an SGML tag is being assigned for the first time, NOT ASSIGNED will be displayed in the Current HTML Tag list box 704 of Fig. 12B.

[0113] To fill the Legal HTML Tag list box 706 of Fig. 12B, the MapEditDialog 950 box of Fig. 18A(1) sends a last HTML tag 972 in the Current HTML Tag list box 704 of Fig. 12B to MapTag 952 through a function call setLastCurrentHTMLTagInList 972. Then the MapEditDialog 950 box gets legal HTMLTags 974, along with a YES message 974, through a function call getLegalHTMLTag 974 from MapTag 952. The function will get one HTMLTag 974 at a time and display it in the Legal HTML Tag list box 706 of Fig. 12B. The legal HTML tags are those which can follow the last HTML tag in the current HTML Tag list box 704 of Fig. 12B. In Fig. 18A(2), the MapTag 952 gets legal HTMLTags 976, along with a YES message 976, one at a time, from HTMLSymbolTable 800, through a function call getFirstLegalElement 976, or MapTag 952 will get legal HTMLTags 978, along with a YES message 978, from HTMLSymbolTable 800 through a function call GetNextLegalElement 978.

[0114] To add an HTML tag from the Legal HTML Tag list box 706 of Fig. 12B to the Current HTML Tag list box 704 of Fig. 12B, the user selects an HTML tag from the Legal HTML Tag list box 706 of Fig. 12B by double clicking the mouse on the HTML tag. The HTML tag is added to the Current HTML Tag list box 704. The selected HTMLTag 980 of Fig. 18A(1) is sent to MapTag 952 through a function call addSelectedHTMLTag 980. In Fig. 18A(2), MapTag 952 sends an HTMLTag 982 to MapCreateEditService 792 through a function call selectedHTMLTag 982. MapTag 952 informs MapCreateEditService 792 that the HTMLTag 982 must be added to the current map through a function call addSelectedHTMLTagToCurrentMappingList 984. MapEditDialog 950 of Fig. 18A(1) sets an HTML tag 986 as the last tag in the list by a function call setLastCurrentHTMLTagInList 986 to MapTag 952.

[0115] In order to update the list of Legal HTML Tags 706 of Fig. 12B, MapEditDialog 950 of Fig. 18A(1) obtains an HTMLTag 988, with a YES message 988 if there exists a legal tag, from MapTag 952 using a function call getLegalHTMLTag 988. MapTag 952 then obtains an HTMLTag 990, with a YES message 990, from HTMLSymbolTable 800 using a function call getFirstLegalElement 990, if it is the first legal element requested, or MapTag 952 obtains an HTMLTag 992, with a YES message 992, from HTMLSymbolTable 800 using a function call getNextLegalElement 992, if it is not the first legal element requested.

[0116] To fill the HTML Tag Attribute list box 720 of Fig. 12B, MapEditDialog 950 of Fig. 18A(1) sends an HTMLTag 994 in the Current HTML Tag list box 704 of Fig. 12B to AssignAttribute 954 through a function call selectedHTMLTag 994. Then the MapEditDialog 950 gets attributes of the HTML tag, HTMLAttribute 996, along with a YES message 996, from AssignAttribute 954 through a function call getNextHTMLAttribute 996. The function getNextHTMLAttribute 996 gets one HTMLAttribute 996 at a time and displays them in the HTML Tag Attribute list box 720 of Fig. 12B. AssignAttribute 954 of Fig. 18A(3) gets, one at a time, an HTMLAttribute 998, along with a YES message 998, of the HTMLTag 998 from HTMLSymbolTable 800 through a function call getFirstAttributeOfElement 998. If it is not the first attribute, AssignAttribute 954 gets an HTMLAttribute 1002 of the HTMLTag 998, along with a YES message 1002, from HTMLSymbolTable 800 through a function call getNextAttribute 1002. For each attribute AssignAttribute 954 gets from HTMLSymbolTable 800, and returns to MapEditDialog 950 of Fig. 18A(1), MapEditDialog 950 checks with AssignAttribute 954 to see if the attribute is of the required type through a function call IsCurrentAttributeOfRequiredType 1000, with a NO message 1000 indicating it is not, to be obtained from HTMLSymbolTable 800. AssignAttribute 954 of Fig. 18A(3) then checks with HTMLSymbolTable 800 to see if the attribute is of the required type through a function call IsCurrentAttributeOfRequiredType 1001, with a NO message 1001 indicating it is not.

[0117] To select an attribute to assign it a value, the user selects an attribute in the HTML Tag Attribute list box 720 of Fig. 12B by double clicking the mouse on the attribute. The MapEditDialog box 950 of Fig. 18A(1) sends a selected HTMLAttribute 1004 to AssignAttribute 954 through a function call selectedHTMLAttribute 1004. Depending upon the attribute type to which the HTMLAttribute 1006 is assigned, the MapEditDialog box 950 of Fig. 18A(1) will take different actions. For example, AssignAttribute 954 of Fig. 18A(3) gets an SGMLAttribute type 1006, an SGMLTag 1006, and an SGMLAttribute 1006 by sending an HTMLAttribute 1006 to MapCreateEditService 792 using a function call getAttributeAssignmentInformationForHTMLAttribute 1006. The MapEditDialog 950 box of Fig. 18A(1) gets an SGMLAttribute type 1008 for the selected HTML Attribute 1004 from AssignAttribute 954 through a function call getAttributeType 1008. The SGML Attribute radio button 726 of Fig. 12B is displayed depressed. The MapEditDialog 950 of Fig. 18A(1)gets a

source SGMLTag 1010 and a source SGMLAttribute 1010 assigned to the HTML Attribute 1004 through a function call getSourceSGMLTagAndAttribute 1010 and displays them in the Source SGML Tag list box 722 of Fig. 12B, and the SGML Tag Attribute list box 724 of Fig. 12B. Next, the MapEditDialog 950 gets a source SGMLTag 1012, one SGMLTag 1012 at a time, which can be assigned to the HTML attribute 1004, along with a YES message 1012, from AssignAttribute 954 through a function call getNextSourceSGMLTag 1012, and displays them in the Source SGML Tag list box 722 of Fig. 12B. AssignAttribute 954 of Fig. 18A(3) gets source SGML Tags 1014, with a YES message 1014 if getting the first element, from SGMLSymbolTable 884 through a function call GetFirstElement 1014, or, one by one, gets a source SGML Tag 1018, with a YES message 1018, through a call GetNextElement 1018 if it is not the first element. AssignAttribute 954 verifies that the source SGML Tag 1014 has an attribute by checking with the SGMLSymbolTable 884 through a function call elementHasAttribute 1016 to obtain a YES 1016 response, for the first element if it has attributes. If it is not the first element, AssignAttribute 954 verifies that the source SGML Tags 1018 have attributes by checking with the SGMLSymbolTable 884 through a function call elementHasAttribute 1020 to obtain a YES 1020 response. The MapEditDialog 950 of Fig. 18A(1) gets all attributes SGMLAttribute 1022, one SGMLAttribute 1022 at a time, with a YES message 1022, of the previously assigned source SGML tag from AssignAttribute 954 through a function call getNextSourceSGMLAttribute 1022 and displays the attributes in the source SGML Tag Attribute list box 724 in Fig. 12B. AssignAttribute 954 of Fig. 18A(3) gets SGML attributes 1024, and a YES message 1024, from SGMLSymbolTable 884 by sending an SGML tag 1024 through a function call getFirstAttributeOfElemont 1024 if it is the first attribute of the element, or it gets SGMLAttributes 1026, one SGMLAttribute 1026 at a time, along with a YES message 1026, from SGMLSymbolTable 884 through a function call getNextAttribute 1026. The user selects an SGML tag from the Source SGML Tag list box 722 of Fig. 12B by double clicking the mouse on the SGML tag. The current implementation supports selection by double clicking the mouse on the selection. However, any alternative selection technique can be used, such as highlighting the selection and pressing the Enter or Return key. MapEditDialog 950 of Fig. 18C(1) sends a selected source SGMLTag 1110 to AssignAttribute 954 through a function call selectedSourceSGMLTag 1110. Then MapEditDialog 950 of Fig. 18A(1) gets all SGMLAttributes 1022 of the selected source SGMLTag 1110, one SGMLAttribute 1022 at a time, along with a YES message 1022, from AssignAttribute 954 through a function call getNextSourceSGMLAttribute 1022 and displays them in the SGML Tag Attribute list box 724 of Fig. 12B. AssignAttribute 954 of Fig. 18A(3) gets the SGMLAttributes 1024, along with a YES message 1024, from SGMLSymbolTable 884 through a function call getFirstAttributeOfElement 1024 if it is requesting the first attribute, or AssignAttribute 954 gets the SGMLAttributes 1026, one SGMLAttribute 1026 at a time, along with a YES message 1026, from SGMLSymbolTable 884 through a function call getNextAttribute 1026 if it is not the first attribute.

[0118] The user selects an SGML attribute from the source SGML Attribute list box 724 of Fig. 12B by double clicking the mouse on the SGML attribute. MapEditDialog 950 of Fig. 18C(1) sends a selected source SGMLAttribute 1112 to AssignAttribute 954 through a function call selectedSourceSGMLAttribute 1112. The user selects the Assign button 736 of Fig. 12B. MapEditDialog 950 lets AssignAttribute 954 know that an SGML attribute was assigned to the HTML attribute through a function call assignedSGMLAttribute 1114 of Fig. 18C(1). AssignAttribute 954 of Fig. 18C(3) sends, for the HTML attribute that is being assigned, an HTMLAttribute 1116, a source SGMLTag 1116, and the source SGMLAttribute 1116 to MapCreateEditService 792 through a function call assignHTMLAttributeWithSGMLAttribute 1116.

[0119] If SGML content was assigned to the HTML attribute, then the SGML Content radio button 728 of Fig. 12B is depressed. The MapEditDialog box 950 of Fig. 18B(1) gets a content SGMLTag 1048 assigned to the HTML attribute from AssignAttribute 954 through a function call getContentSGMLTag 1048, along with a YES message 1048, and displays it in the Source SGML Tag list box 722 of Fig. 12B. The MapEditDialog box 950 gets all the content SGMLTags 1052, one SGMLTag 1052 at a time, which can be assigned to the HTML attribute, along with a YES message 1052, from AssignAttribute 954 through a function call getNextContentSGMLTag 1052 and displays them in the Source SGML Tag list box 722 of Fig. 12B. AssignAttribute 954 of Fig. 18B(3) gets content SGMLTags 1054, along with a YES message 1054, from SGMLSymbolTable 884 through a function call getFirstPrimitiveElement 1054 if it is the first SGML tag requested, or AssignAttribute 954 gets content SGMLTags 1056, one SGMLTag 1056 at a time, along with a YES message 1056, from SGMLSymbolTable 884 through a function call getNextPrimitiveElement 1056.

[0120] The user selects the content SGML tag from the Source SGML Tag list box 722 of Fig. 12B by double clicking the mouse on the SGML tag. MapEditDialog 950 of Fig. 18B(1) sends the selected content SGMLTag 1058 to AssignAttribute 954 through a function call selectedContentSGMLTag 1058.

[0121] The user selects the Assign button 736 of Fig. 12B. MapEditDialog 950 of Fig. 18B(1) lets AssignAttribute 954 know that an SGML content was assigned to the HTML attribute through a function call assignSGMLContent 1060. AssignAttribute 954 of Fig. 18B(3) sends an HTMLAttribute 1062 that is being assigned the SGML content and a content SGMLTag 1062 to MapCreateEditService 792 through a function call assignHTMLAttributeWithSGMLContent 1062.

[0122] If System was assigned to the HTML attribute, the System radio button 730 of Fig. 12B is depressed. MapEditDialog 950 takes no further action. The user selects the Assign button 736 of Fig. 12B. MapEditDialog 950 of Fig. 18B(1) lets AssignAttribute 954 know that a system value was assigned to the HTMLAttribute through a function call

assignSystem 1049. AssignAttribute 954 of Fig. 18B(3) sends the HTMLAttribute 1051 that is being assigned a system value to MapCreateEditService 792 through a function call assignHTMLAttributeWithSystem 1051.

[0123] If No Value was assigned to the HTML attribute, the No Value radio button 732 is depressed. MapEditDialog 950 will take no further action. The user selects the Assign button 736 of Fig. 12B. MapEditDialog 950 of Fig. 18C(1) lets AssignAttribute 954 know that no value is assigned to the HTML attribute through a function call assignNoValue 1102. AssignAttribute 954 of Fig. 18C(3) sends the HTMLAttributes 1104 to be assigned no value to MapCreateEditService 792 through a function call assignHTMLAttributeWithNoValue 1104.

[0124] If User Input was assigned to the HTML attribute, the User Input radio button 734 is depressed. MapEditDialog 950 of Fig. 18C(1) gets a UserInput 1100 from AssignAttribute 954 through a function call getUserInputData 1100 and then displays the information in the User Input text edit box 731 of Fig. 12B. The user enters data into the text edit box 731 of Fig. 12B. MapEditDialog 950 of Fig. 18C(1) sends a UserInputData 1106 to AssignAttribute 954 through a function call selectedUserInputData 1106. The user then selects the Assign button 736 of Fig. 12B. MapEditDialog 950 of Fig. 18C(1) lets AssignAttribute 954 know that a user input is to be assigned to the HTML attribute through the function assignUserInput 1108. AssignAttribute 954 of Fig. 18C(3) sends an HTMLAttribute 1109 that is being assigned a user input value, and the UserInputData 1109, to MapCreateEditService 792 through a function call assignHTMLAttributeWithUserInput 1109.

[0125] For any of the HTML attribute source types assigned to the HTML attribute, the user has options to change the source type of the HTML attribute. For example, if the No Value radio button 732 of Fig. 12B is depressed for a selected HTML attribute, the user has an option to change the source type of the HTML attribute by pressing any of the other radio buttons such as the User Input radio button 734 of Fig. 12B. Depending upon which radio button the user selects, the user will need to enter more information before the HTML, attribute is assigned a value.

[0126] For the radio buttons SGML Attribute 726, SGML Content 728, and User Input 734, the user can change the input selection more than one time. For example, the user can change the user input in the text edit box 731 more than one time. As another example, the user can select one content SGML tag in the Source SGML Tag list box 722, and later decide to select another content SGML tag. The most recent value given by the user is the value assigned to the HTML attribute when the Assign button 736 is selected.

[0127] The user repeats selection of an attribute to assign it a value, selecting a radio button for an HTML attribute source type, assigning a value to the HTML attribute, and selecting the Assign button 736 of Fig. 12B until the user has assigned all the HTML attributes desired. The user then selects the Assign Done button 740 of Fig. 12B. MapEditDialog 950 of Fig. 18B(1) lets MapCreateEditService 792 know that it is done assigning values to the attributes through a function call assignDoneSelected 1064 to AssignAttribute 954, which sends a call assignDoneSelected 1066 to MapCreateEditService 792 of Fig. 18B(3).

[0128] The user repeats adding more HTML tags to the Current HTML Tag list box 704 of Fig.12B at will. At any time the user can clear the Current HTML Tag list box 704 or delete HTML tags from the Current HTML Tag list box 704 by selecting the Clear HTML button 708 or the Delete HTML button 710 of Fig. 12B. Once the user has completed mapping one SGML tag, the user selects the Map SGML Tag button 714. MapEditDialog 950 of Fig. 18B(1) informs MapTag 952 that the user has completed mapping the selected SGML tag through a function call doneMappingSGMLTag 1068. MapTag 952 of Fig. 18B(2) informs MapCreateEditService 792 that the user has completed mapping the selected SGML tag through the function finishOneMapping 1070. MapEditDialog 950 of Fig. 18B(1) requests a next SGMLTag 1072 for mapping from MapTag 952, along with a YES message 1072, using a function call getNextSGMLTag 1072 and displays the SGMLTag 1072 in the SGML Tag list box 702 of Fig. 12B. MapTag 952 of Fig. 18B(2) then obtains a next SGMLTag 1074, along with a YES message 1074, from SGMLSymbolTable 884 using a function call getNextElement 1074. MapTag 952 then returns the SGMLTag 1072 to MapEditDialog 950 of Fig. 18B(1) in response to the function call getNextSGMLTag 1072.

[0129] The user repeats processing the map for all the SGML tags the user wants to map. When the user is finished with the map editor, the user selects the Done button 716 of Fig. 12B. MapEditDialog 950 of Fig. 18B(1) informs MapTag 952 that it is done mapping through a function call doneMapping 1076. MapTag 952 of Fig. 18B(2) informs MapCreateEditService 792 that the mapping of the SGML tags is completed through a function call finishCreatingMap 1078.

[0130] Fig. 19 illustrates an exemplary hardware configuration upon which the invention may be implemented. A Workstation 1200 has component parts a Display Controller 1202, a Central Processing Unit ("CPU") 1204, a Random Access Memory ("RAM") 1206, a Read Only Memory ("ROM") 1208, an Input Controller 1210, connected to a Keyboard 1212 and a Mouse 1214, a System Bus 1220, a Hard Disk 1222 and a Floppy Drive 1224 connected to a Disk Controller 1226, a Comm Controller 1228 connected to a Network 1230, and an Input/Output ("I/O") Controller 1232 connected to a Hard Disk 1236 and a Printer 1234, and a Cathode Ray Tube ("CRT") 1238 connected to the Display Controller 1202. The System Bus 1220 connects the CPU 1204, the RAM 1206, the ROM 1208, the Input Controller 1210, the Disk Controller 1226, the Comm Controller 1228, the I/O Controller 1232, and the Display Controller 1202 for transmitting data over the connection line.

[0131] For example, the computer code generated for execution is loaded into the RAM 1206 for execution by the CPU 1204, using the System Bus 1220, with input files stored on the Hard Disk 1236, with other input coming from the Keyboard 1212 and the Mouse 1214 through the Input Controller 1210, and from the Hard Disk 1222 and the Floppy Drive 1224, through the Disk Controller 1226, onto the System Bus 1220. The System Bus 1220 interacts with the ROM 1208, the Network 1230, and the Comm Controller 1228. The GUI of the system can be displayed on the CRT 1238 through the Display Controller 1202, and on output to the Printer 1234 or to the Hard Disk 1236 through the I/O Controller 1232.

[0132] Other implementations of the map creator and editor for transforming a first structured information format to a second structured information format are possible using the procedures described previously for Figs. 1A-19. Far example, variable names in a first database format may be mapped to variable names in a second database format. Another exemplary implementation is a mapping of public identifiers in an ISO/IEC 9070 format to file names in a UNIX file system format.

[0133] For other implementations, the same techniques described with regard to the SGML to HTML mapping and transformation are utilized, with structure of information defined differently from an SGML DTD. In general terms, a parser breaks down an input source file into source components and their structure, based upon a structure format specified for the input source file, for map creating and editing. The source components and their structure are presented to the user for interactive selection of components of the first structure, with candidate target components of the second structure presented to the user for selection of target components for the mapping of the source components for creation of rules for a transformation map. An exemplary implementation of a mapping of public identifiers in an ISO/IEC 9070 format to file names in a UNIX file system format is discussed below with regard to Figs. 20A-20H.

[0134] Fig. 20A illustrates a public identifier 1400 in ISO/IEC 9070 standard format. An owner name is made up of a registered owner name and an unregistered owner name. For this example, the owner name is 'XYZ'. The public identifier further has an object name, separated from the owner name by '//'. For this example, the object name is 'font::metric::x-offset::622'.

[0135] Mapping one system structure to another system structure involves transformation of strings in one allowable character set to strings of another allowable character set. For example, computer programming languages are defined as having an alphabet of acceptable characters for forming valid variable names. A first programming language may be defined as allowing the '-' (hyphen) character to be embedded in a valid variable name, but not the '_' (underscore) character. A second programming language may be defined as allowing the '_' character, but not the '-' character. A mapping of valid variable names of the first programming language to valid variable names of the second programming language would involve mapping occurrences of '-' to a different character, such as '_', which is acceptable in the second programming language environment.

[0136] In the example of mapping the ISO/IEC 9070 naming scheme to the UNIX file name scheme, the separator '//' is allowed in ISO/IEC 9070, but is not a valid character sequence in the UNIX file system naming conventions. A user wishing to map valid ISO/IEC 9070 names to valid UNIX file names needs to transform every occurrence of the separator '//' into a valid UNIX file name character string.

[0137] Fig. 20B illustrates an exemplary mapping of an ISO/IEC 9070 public identifier to a UNIX file name format. The exemplary mapping maps the structured public identifier to a flat UNIX file name. Lines 1420, 1422, 1424, and 1426 illustrate rules to map component name strings to identical strings in the UNIX format. Line 1428 illustrates a rule to map an ISO/IEC 9070 owner name component separator '::', which is not widely used, to the character '_', which is a valid UNIX character. Line 1430 illustrates a rule to map an ISO/IEC 9070 owner name, object name component separator '//', which is not a valid string in the UNIX file format, to '__' (two underscore characters).

[0138] Fig. 20C illustrates an exemplary UNIX file name 1440 resulting from mapping the exemplary ISO/IEC 9070 name of Fig. 20A through the mapping of Fig. 20B. Ownername 'XYZ' of line 1400 of Fig. 20A is mapped to 'XYZ' using the rule of line 1420 of Fig. 200. The '//' is mapped to '__' using the rule of line 1430 of Fig. 20B. The three substrings '::' in the object name of public identifier 1400 of Fig. 20A are each mapped to a character '_' using the rule 1428 of Fig. 20B.

[0139] Fig. 20D illustrates an exemplary user interface for mapping a public identifier 1502 of ISO/IEC 9070 to a UNIX file system format 1504. A map editor and creator 1500 maps names in the ISO/IEC 9070 convention to names in the UNIX file system convention. An exemplary mapping starts with system display 1502 of public identifier components and a display of valid UNIX file name components 1504. The public identifier components 1502 are registered owner name, unregistered owner name, and object name. A user selects one of these options. If owner name is selected, then the user is asked to select from prefix and owner-name components 1522 in Fig. 20E. User options presented are a create directory 1506, a map 1508, a merge all 1510, a next 1512, and a previous 1514. The create directory 1506 option allows the user to create a new directory name in the UNIX file system window 1504. The map option 1508 allows the user to request that a map be created at the time of request. The merge all 1510 option allows the user to create a UNIX file name 1504 by merging all the components of the public identifier name 1502 into a flat file name 1504. The next 1512 option allows the user to step to a next screen. The previous 1514 option allows the user to back up to the previous screen.

[0140] Fig. 20E illustrates an exemplary user interface 1520 for a registered owner 1522 component of the ISO/IEC 9070 public identifier 1502 of Fig. 20D. User interrace 1520 options presented are a window 1522 showing a prefix and an owner-name component. User options are a map individually 1524, a merge both 1526, a next 1527, and a previous 1528. The map individually 1524 option allows the user to map individual components of the ISO/IEC 9070 name 1522 to individual components of the UNIX file system scheme 1504 of Fig. 20D. The merge both 1526 option allows the user to merge components of the registered owner name 1522 into one flat UNIX file name or directory name. The next 1527 option allows the user to step to a next screen. The previous 1528 option allows the user to back up to the previous screen.

[0141] Fig. 20F illustrates an exemplary user interface 1530 for mapping the prefix, owner name component separator to the UNIX legal character set format 1534. The registered owner component has a prefix and an owner-name component separator "::" 1532 which is not a widely used character string in the UNIX environment. The user is allowed to map the '::' separator 1532 to any of the valid characters in the UNIX file system character set 1534, with a mapping to '_' as a default mapping. User options are a map 1536, a next 1537, and a previous 1538. The map 1536 option allows the user to select creating a map, with the assumption that the user has finished selecting options for creation of the map. The next 1537 option allows the user to step to a next screen. The previous 1538 option allows the user to back up to the previous screen.

[0142] Fig. 20G illustrates an exemplary user interface 1540 for mapping an owner name character 1542 to valid characters 1544 of the UNIX file system format. The user is given the options of mapping special characters 1542 which are valid in the ISO/IEC 9070 scheme to characters which are valid in the UNIX file system scheme 1544. A mapping of a character in the ISO/IEC 9070 scheme 1542 is set to '_' as a default mapping. The user is given options of a map 1546, a next 1548, and a previous 1550. The map 1546 option allows the user to select creating a map, with the assumption that the user has finished selecting options for creation of the map. The next 1548 option allows the user to step to a next screen. The previous 1550 option allows the user to back up to the previous screen.

[0143] Fig. 20H illustrates an exemplary user interface 1560 for the user to map a registered owner component 1562 to a UNIX file scheme format 1564. The user is allowed to select a prefix component of the registered owner name or an owner name component other than prefix 1562. The user is allowed to select a directory option in the UNIX scheme 1564. The user is also allowed to select a file with object name option in the UNIX file scheme 1564. The user is given an option of a map 1566 for creating the map with the options currently selected. The user is also given an option of a previous 1568 to back up to the previous screen.

[0144] The present invention has been described using an exemplary implementation of a mapping creator and editor for an SGML to HTML transformer with user interaction for creation and editing of the map, and an exemplary mapping creator and editor for an ISO/IEC 9070 to a UNIX file format transformer. The example shown in this disclosure uses OOP and Windows GUI techniques to implement the user interface, map processing, and transformation. However, the user interface can be implemented using text line queries or menus. Programming methodologies other than OOP can be used for implementing the processing. References to storage areas can be made by techniques other than using pointers.

[0145] This invention may be conveniently implemented using a conventional general purpose digital computer or microprocessor programmed according to the teachings of the present specification, as will be apparent to those skilled in the computer art. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art. The invention may also be implemented by the preparation of application specific integrated circuits or by interconnecting an appropriate network of conventional component circuits, as will be readily apparent to those skilled in the art.

[0146] The present invention includes a computer program product which is a storage medium including instructions which can be used to program a computer to perform a process of the invention. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, CD-ROMs, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions.

[0147] Obviously, numerous modifications and variations of the present invention are possible in light of the above teachings. It is therefore to be understood that within the scope of the appended claims, the invention may be practiced otherwise than as specifically described herein.

Appendix A

```
<!doctype test [
<!element test - o (front, section+)
<!element front - o (title, author, keywords?)>
<!element title - o (#PCDATA)>
<!element author - o (fname, surname, title?)>
<!element (fname, surname) - o (#PCDATA)>
<!element keywords - o (#PCDATA)>
<!element section - o (number?, title?, para*, subsec1*)>
<!element subsec1 - o (number?, title?, para*, subsec2*)>
<!element subsec2 - o (number?, title?, para+)>
<!element number - o (#PCDATA)>
<!element para - o (#PCDATA)>
]>
```

Appendix B

30	Mapping	
32	Front →	Null
34	Title →	H3
36	Author →	Null
38	Frame →	P
40	Surname →	P
42	Keywords →	P
44	Section →	Null
46	Number →	P strong
48	Para →	P
50	Subsec 1 →	Null
52	Subsec 2 →	Null

Appendix C

```
<test>
<front>
<title>
Test mapping
</title>
<author>
<fname>
Tester
</fname>
<surname>
Giver
</surname>
</author>
<keywords>
Mapping
</keywords>
</front>
<section>
<number>
1
</number>
<title>
First Major Section
</title>
<para>
The first major section para.
</para>
<subsec1>
<number>
1.1
</number>
<title>
Subsection 1.1
</title>
<para>
This is a para in the subsec1.1
</para>
</subsec1>
<subsec1>
  <number>
  1.2
  </number>
  <title>
  Subsection 1.2
  </title>
  <para>
  This is a subsection 1.2
  </para>
  </subsec1>
  </section>
  <section>
  <number>
  2
  </number>
```

```
168 <title>
170 Second Major Section
172 </title>
174 <para>
176 The second major section para.
178 </para>
180 <subsecl>
182 <number>
184 2.1
186 </number>
188 <title>
190 Subsection 2.1
192 </title>
194 <para>
196 This is a para in the subsec 2.1
198 </para>
200 </subsecl>
202 <subsecl>
204 <number>
206 2.2
208 </number>
210 <title>
212 Subsection 2.2
214 </title>
216 <para>
218 This is a subsection 2.2
220 </para>
222 </subsecl>
224 </section>
226 </test>
```

Appendix D

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage 2.0">
<title>test</title>
</head>

<body bgcolor="#FFFFFF">

<h3>Test mapping</h3>

<p>Tester</p>

<p>Giver</p>

<p>Mapping</p>

<p><strong>1</strong></p>

<h3>First Major Section</h3>

<p>The first major section para.</p>

<p><strong>1.1</strong></p>

<h3>Subsection 1.1</h3>

<p>This is a para in the subsec1.1</p>

<p><strong>1.2</strong></p>

<h3>Subsection 1.2</h3>

<p>This is a subsection 1.2</p>

<p><strong>2</strong></p>

<h3>Second Major Section</h3>

<p>The second major section para.</p>

<p><strong>2.1</strong></p>

<h3>Subsection 2.1</h3>

<p>This is a para in the subsec 2.1</p>

<p><strong>2.2</strong></p>

<h3>Subsection 2.2</h3>

<p>This is a subsection 2.2</p>
</body>
</html>
```

Claims

1. An object-oriented system for processing structured information for implementation by a computer in an object-oriented framework, comprising:

   a storage means;
   a first obtaining means for obtaining an interactive input from a user;
   a second obtaining means for obtaining a first structural description of a first structured information format;
   a third obtaining means for obtaining a second structural description of a second structured information format;
   means for creating a rule to transform an element of the first structured information format into an element of the second structured information format utilizing the interactive input from the user, the first structural description, and the second structural description; and
   means for outputting the rule,
   wherein at least one of the first obtaining means, the second obtaining means, the third obtaining means, the means for creating, and the means for outputting includes a software object.

2. A system according to Claim 1, wherein the first structured information format includes an ISO/IEC 9070 public identifier naming format, the second structured information format includes an operating system file name format, and the means for creating comprises:

   means for creating a rule to transform an ISO/IEC 9070 public identifier naming format element into an operating system file name format element utilizing the interactive input from the user, a structural description of the ISO/IEC 9070 public identifier naming format, and a structural description of the operating system file name format.

3. A system according to Claim 1, wherein the first structured information format includes a first database variable name format, the second structured information format includes a second database variable name format, and the means for creating comprises:

   means for creating a rule to transform a first database variable name format element into a second database variable name format element utilizing the interactive input from the user, a structural description of the first database variable name format, and a structural description of the second database variable name format.

4. A system according to Claim 1, wherein the structured information includes a markup language format, the first structured information format includes a first markup language format, the second structured information format includes a second markup language format, and the means for creating comprises:

   means for creating a rule to transform a first markup language format element into a second markup language format element utilizing the interactive input from the user, a structural description of the first markup language format, and a structural description of the second markup language format.

5. A system according to Claim 4, wherein the first markup language format includes a Standard Generalized Markup Language ("SGML"), the second markup language format includes a HyperText Markup Language ("HTML"), and the means for creating comprises:

   means for creating a rule to transform an SGML element of the first markup language format into an HTML element of the second markup language format utilizing the interactive input from the user, the first structural description which includes an SGML Document Type Definition ("DTD"), and the second structural description which includes an HTML DTD.

6. A system according to Claim 1, wherein the means for creating comprises:

   a map creator object.

7. A system according to Claim 6, wherein the means for outputting the rule comprises:

   an object method for outputting the rule to a map object.

8. A system according to Claim 6, wherein the map creator object comprises:

a reference to a software object for an element for transformation of the first structured information format;
a reference to a software object for an element of the second structured information format, for transformation of the element of the first structured information format;
a reference to a software object for a property of the element of the second structured information format, for transformation of the element of the first structured information format;
a reference to a software object for an attribute value of the element of the second structured information format, for transformation of the element of the first structured information format;
an object method for obtaining the element for transformation of the first structured information format, which has been interactively selected by the user, using the software object for the element for transformation of the first structured information format;
an object method for obtaining the element of the second structured information format which corresponds to the element of the first structured information format, which has been interactively selected by the user, using the software object for the element of the second structured information format;
an object method for determining a property of the element of the second structured information format which has been selected by the user, using the software object for a property of the element of the second structured information format;
an object method for obtaining a second structured information format attribute value which has been interactively input by a user, using the software object for the attribute value of the element of the second structured information format; and
an object method for assigning the attribute value which has been interactively input by a user to the second structured information format attribute value.

9. A system according to Claim 8, wherein the map creator object further comprises:

a reference to a software object for registering an instance of an element for transformation of the first structured information format; and
a reference to a software object for unregistering the instance of an element for transformation of the first structured information format when the element is no longer needed by the map creator

10. A system according to Claim 1, further comprising:

means for processing an element of the first structured information format into a plurality of first structured information format components.

11. A system according to Claim 10, wherein the means for processing an element of the first structured information format into a plurality of first structured information format components comprises:

a parser object.

12. A system according to Claim 11, wherein the parser object comprises:

a reference to the element of the first structured information format;
a reference to a storage area for the plurality of first structured information format components; and
an object method for processing the element of the first structured information format into the plurality of first structured information format components for storage in the storage area for the plurality of first structured information format components.

13. A system according to Claim 12, further comprising:

means for utilizing the rule to transform the element of the first structured information format into the element of the second structured information format.

14. A system according to Claim 13, wherein the means for utilizing the rule to transform the element of the first structured information format into the element of the second structured information format comprises:

a transformer object.

15. A system according to Claim 1, wherein the means for obtaining an interactive input from a user comprises:

a user interface object.

16. A system according to Claim 15, wherein the user interface object comprises:

a reference to a software object for user input;
an object method for obtaining interactive input from the user of a selection of an element for transformation of a first structured information format using the software object for user input; and
an object method for obtaining interactive input from the user of a selection of an element of a second structured information format which corresponds to the element of the first structured information format using the software object for user input.

17. A system according to Claim 15, wherein the user interface object further comprises:

a reference to a software object for user input of a selection of a transformation to be performed on the element of the first structured information format; and
an object method for obtaining interactive input from the user of the selection of a transformation to be performed on the element of the first structured information format using the software object for user input of the selection of the transformation.

18. A system according to Claim 17, wherein the object method for creating a rule to transform an element of a first structured information format into an element of a second structured information format utilizing the interactive input from the user further comprises:

a reference to a software object for a rule to be created; and
an object method for creating a rule to map the second element of the first structured information format to a null string using the software object for the rule to be created, when the selection of a transformation which has been input by the user indicates a null transformation is to be performed.

19. A system according to Claim 17, wherein the object method for creating a rule to transform an element of a first structured information format into an element of a second structured information format utilizing the interactive input from the user further comprises:

a reference to a software object for a rule to be created; and
an object method for creating a rule to map the second element of the first structured information format to a copy of the second element of the first structured information format using the software object for the rule to be created, when the selection of a transformation which has been input by the user indicates a transformation of the second element of the first structured information format to a copy of the second element of the first structured information format is to be performed.

20. A system according to Claim 15, wherein the user interface object further comprises:

a reference to a software object for an interactive user input of a source for inputting the second structured information format attribute value;
a reference to a software object for an interactive user input of the second structured information format attribute value;
an object method for obtaining interactive input from the user of the source for inputting the second structured information format attribute value using the software object for the interactive user input of the source for inputting the second structured information format attribute value; and
an object method for obtaining interactive input from the user of the second structured information format attribute value using the software object for the interactive user input of the second structured information format attribute value.

21. A system according to Claim 20, further comprising:

a reference to a software object for a rule to be created;
an object method for examining the source which has been input by the user; and

an object method for assigning a null value to the second structured information format attribute value using the software object for the rule to be created, when the source which has been input by the user indicates no source is to be used.

22. A system according to Claim 20, further comprising:

a reference to a software object for a rule to be created;
an object method for examining the source which has been input by the user; and
an object method for assigning a system value to the second structured information format attribute value using the software object for the rule to be created, when the source which has been input by the user indicates a system source is to be used.

23. A system according to Claim 20, further comprising:

a reference to a software object for a rule to be created;
an object method for examining the source which has been input by the user; and
an object method for assigning a first structured information format attribute value to the second structured information format attribute value using the software object for the rule to be created, when the source which has been input by the user indicates a first structured information format attribute source is to be used.

24. A system according to Claim 20, further comprising:

a reference to a software object for a rule to be created;
an object method for examining the source which has been input by the user; and
an object method for assigning a first structured information format content value to the second structured information format attribute value using the software object for the rule to be created, when the source which has been input by the user indicates a first structured information format content source is to be used.

25. A system according to Claim 20, further comprising:

a reference to a storage buffer for the source which has been input by the user;
an object method for examining the source which has been input by the user using the storage buffer for the source which has been input by the user;
an object method for interactively inputting a user input value, when the source which has been input by the user indicates a user input source is to be used; and
an object method for assigning the user input value to the second structured information format attribute value, when the source which has been input by the user indicates a user input source is to be used.

26. A system according to Claim 1, wherein the user comprises:

a software object.

27. An object-oriented computer program product for processing structured information for implementation by a computer in an object-oriented framework, comprising:

a storage means;
a first obtaining means for obtaining an interactive input from a user;
a second obtaining means for obtaining a first structural description of a first structured information format;
a third obtaining means for obtaining a second structural description of a second structured information format;
means for creating a rule to transform an element of the first structured information format into an element of the second structured information format utilizing the interactive input from the user, the first structural description, and the second structural description; and means for outputting the rule,
wherein at least one of the first obtaining means, the second obtaining means, the third obtaining means, the means for creating, and the means for outputting includes a software object.

28. A computer program product according to Claim 27, wherein the first structured information format includes ISO/IEC 9070 public identifier naming format, the second structured information format includes an operating system file name format and the means for creating comprises:

means for creating a rule to transform an element of a first structured information format which includes an ISO/IEC 9070 public identifier element into an element of a second structured information format which includes an operating system file name format element utilizing the interactive input from the user, the first structural description which includes a structural description of the ISO/IEC 9070 public identifier format, and the second structural description which includes a structural description of the operating system file name format.

29. A computer program product according to Claim 27, wherein the structured information includes database variable names, the first structured information format includes a first database variable name format, the second structured information format includes a second database variable name format, and the means for creating comprises:

means for creating a rule to transform an element of a first structured information format which includes a first database variable name format element into an element of a second structured information format which includes a second database variable name format element utilizing the interactive input from the user, the first structural description which includes a structural description of the first database variable name format, and the second structural description which includes a structural description of the second database variable name format.

30. A computer program product according to Claim 27, wherein the structured information includes markup language, the first structured information format includes a first markup language, the second structured information format includes a second markup language, and the means for creating comprises:

means for creating a rule to transform an element of a first structured information format which includes a first markup language element into an element of a second structured information format which includes a second markup language element utilizing the interactive input from the user, the first structural description which includes a structural description of the first markup language, and the second structural description which includes a structural description of the second markup language.

31. A computer program product according to Claim 30, wherein the first markup language includes SGML, the second markup language includes HTML, and the means for creating further comprises:

means for creating a rule to transform an element of a first markup language which includes an SGML element into an element of a second markup language which includes an HTML element utilizing the interactive input from the user, the first structural description which includes an SGML DTD, and the second structural description which includes an HTML DTD.

32. A computer implemented method to provide a graphical user interface for creating a mapping of a first structural description to a second structural description, comprising the steps of:

displaying an element for transformation of a first structural description;
displaying a list of candidate elements of a second structural description;
inputting, from a user, a first selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
storing the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description as a rule.

33. A method according to Claim 32, wherein the first structural description includes an ISO/IEC 9070 public identifier naming format, the second structural description includes an operating system file name format, and the step of displaying the element for transformation comprises:

displaying the element for transformation which includes an element of the ISO/IEC 9070 public identifier naming format; and
the step of displaying the list of candidate elements comprises:
displaying the list of candidate elements which includes a list of operating system file name candidate elements.

34. A method according to Claim 32, wherein the first structural description includes a first database variable name format, the second structural description includes a second database variable name format, and the step of displaying the element for transformation comprises:

displaying the element for transformation which includes an element of the first database variable name format; and
the step of displaying the list of candidate elements comprises:
displaying the list of candidate elements which includes a list of second database variable name format candidate elements.

35. A method according to Claim 32, further comprising the steps of:

obtaining the stored rule;
displaying the element for transformation of the first structural description;
displaying the first selected element of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;
displaying the list of candidate elements of the second structural description;
inputting, from the user, a second selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
storing the correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description as a rule.

36. A method according to Claim 32, further comprising the steps of:

displaying an icon for the user to input a request to clear the first selection which is being displayed;
inputting the request to clear the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed; and
clearing the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed.

37. A method according to Claim 32, wherein the first structural description includes a first markup language, the second structural description includes a second markup language, and the step of displaying the element for transformation comprises:

displaying the element for transformation which includes an element of the first markup language; and
the step of displaying the list of candidate elements comprises:
displaying the list of candidate elements which includes a list of second markup language candidate elements.

38. A method according to Claim 37, wherein the first markup language includes a Standard Generalized Markup Language ("SGML"), the second markup language includes a HyperText Markup Language ("HTML"), and the step of displaying the element for transformation comprises:

displaying the element for transformation which includes an element of SGML; and
the step of displaying the list of candidate elements comprises:
displaying the list of candidate elements which includes a list of HTML candidate elements.

39. A method according to Claim 32, wherein the storing step comprises:

storing the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description as a rule in a list of rules for transformation.

40. A method according to Claim 39, further comprising the steps of:

displaying a second element for transformation of the first structural description;

displaying a list of candidate elements of the second structural description;

inputting, from the user, a second selection of one of the candidate elements of the second structural description which defines a correspondence between the second element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description for a transformation of the second element of the first structural description to the second structural description; and

storing the correspondence between the second element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description as a rule in a list of rules for transformation.

41. A method according to Claim 39, wherein the inputting step further comprises:

displaying an icon for the user to input a request to store the first selection correspondence as a rule in the list of rules for transformation; and

the storing step further comprises:

inputting the request to store the correspondence as a rule in the list of rules for transformation, when the user inputs the request to store the first selection as a rule; and

displaying a second element for transformation of the first structural description, when the user inputs the request to store the first selection as a rule.

42. A method according to Claim 39, wherein the inputting step further comprises:

displaying an icon for the user to input a request to store the correspondence as a rule in the list of rules for transformation; and

the storing step further comprises:

inputting the request to store the correspondence as a rule in the list of rules for transformation, when the user inputs the request to store the first selection as a rule in the list of rules for transformation; and

storing the list of rules for transformation as a map.

43. A method according to Claim 39, further comprising the steps of:

displaying an icon for the user to input a request to delete the list of rules for transformation;

inputting the request to delete the list of rules for transformation, when the user inputs the request to delete the list or rules for transformation; and

deleting the list of rules for transformation, when the user inputs a request to delete the list of rules for transformation.

44. A method according to Claim 32, further comprising the steps of:

displaying the first selection of one of the candidate elements of the second structural description which defines the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description.

45. A method according to Claim 44, wherein the inputting step comprises:

inputting, from the user, a first ordered list of a plurality of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;

the step of displaying the first selection comprises:

displaying the first ordered list of the plurality of the candidate elements of the second markup language which defines a correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and

the storing step comprises:

storing the correspondence between the element for transformation of the first structural description and the

first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description as a rule.

46. A method according to Claim 45, further comprising the steps of:

obtaining the stored rule;
displaying the element for transformation of the first structural description;
displaying the first ordered list of elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;
displaying the list of candidate elements of the second structural description;
inputting, from the user, a second ordered list of a plurality of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the second ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
storing the correspondence between the element for transformation of the first structural description and the second ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description as a rule.

47. A method according to Claim 45, further comprising the steps of:

displaying an icon for the user to input a request to clear the first ordered list which is being displayed;
inputting the request to clear the first ordered list which is being displayed, when the user inputs the request to clear the first ordered list which is being displayed; and
clearing the first ordered list which is being displayed, when the user inputs the request to clear the first ordered list which is being displayed.

48. A method according to Claim 32, further comprising the steps of:

displaying an attribute of the first selection of one of the candidate elements of the second structural description which corresponds to a transformation of the element of the first structural description to the second structural description, for assignment of an attribute value of the second structural description;
displaying a plurality of icons representing sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed;
displaying the element of the first structural description;
displaying an attribute list of the element of the first structural description;
inputting a user input of a selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed; and
processing the user input of the selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed, when the user inputs the selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed.

49. A method according to Claim 48, wherein the processing step further comprises:

assigning a null value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates no source is to be used.

50. A method according to Claim 48, wherein the processing step further comprises:

assigning a system value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a system value is to be used.

51. A method according to Claim 48, wherein the processing step further comprises:

assigning a first structural description attribute value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a first structural description

attribute value source is to be used.

52. A method according to Claim 48, wherein the processing step further comprises:

assigning a first structural description content value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a first structural description content value source is to be used.

53. A method according to Claim 48, wherein the processing step further comprises:

assigning a user input value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a user input value source is to be used.

54. A method according to Claim 53, wherein the assigning step comprises:

displaying a text input area for the user to input a value to be assigned;
inputting the value entered by the user in the text input area; and
assigning the value input by the user to the attribute of the first selection which is being displayed.

55. A method according to Claim 32, wherein the step of displaying a list of candidate elements of a second structural description further comprises:

displaying a candidate for requesting removal of the first structural description element in the transformation; and
displaying a candidate for requesting ignoring of the first structural description element in the transformation.

56. An apparatus for providing a graphical user interface for creating a mapping of a first structural description to a second structural description, comprising:

an element displaying means for displaying an element for transformation of a first structural description;
a list displaying means for displaying a list of candidate elements of a second structural description;
a user inputting means for inputting, from a user, a first selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
a storing means for storing the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description as a rule.

57. An apparatus according to Claim 56, wherein the first structural description includes an ISO/IEC 9070 public identifier naming format, the second structural description includes an operating system file name format, and the element displaying means further comprises:

means for displaying the element for transformation which includes an element of the ISO/IEC 9070 public identifier naming format; and
the list displaying means further comprises:
means for displaying the list of candidate elements which includes a list of operating system file name candidate elements.

58. An apparatus according to Claim 56, wherein the first structural description includes a first database variable name format, the second structural description includes a second database variable name format, and the element displaying means further comprises:

means for displaying the element for transformation which includes an element of the first database variable name format; and
the list displaying means further comprises:
means for displaying the list of candidate elements which includes a list of second database variable name for-

mat candidate elements.

59. An apparatus according to Claim 56, further comprising:

means for obtaining the stored rule;
means for displaying the element for transformation of the first structural description;
means for displaying the first selected element of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;
means for displaying the list of candidate elements of the second structural description;
means for inputting, from the user, a second selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
means for storing the correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description as a rule.

60. An apparatus according to Claim 56, further comprising:

means for displaying an icon for the user to input a request to clear the first selection which is being displayed;
means for inputting the request to clear the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed; and
means for clearing the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed.

61. An apparatus according to Claim 56, wherein the first structural description includes a first markup language, the second structural description includes a second markup language, and the element displaying means further comprises:

means for displaying the element for transformation which includes an element of the first markup language; and
the list displaying means further comprises:
means for displaying the list of candidate elements which includes a list of second markup language candidate elements.

62. An apparatus according to Claim 61, wherein the first markup language includes an SGML, the second markup language includes an HTML, and the element displaying means further comprises:

means for displaying the element for transformation which includes an element of SGML; and
the list displaying means comprises:
means for displaying the list of candidate elements which includes a list of HTML candidate elements.

63. An apparatus according to Claim 56, wherein the storing means comprises:

means for storing the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description as a rule in a list of rules for transformation.

64. An apparatus according to Claim 63, further comprising:

means for displaying a second element for transformation of the first structural description;
means for displaying a list of candidate elements of the second structural description;
means for inputting, from the user, a second selection of one of the candidate elements of the second structural description which defines a correspondence between the second element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description for a transformation of the second element of the first structural description to the second structural description; and

means for storing the correspondence between the second element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description as a rule in a list of rules for transformation.

65. An apparatus according to Claim 63, further comprising:

means for displaying an icon for the user to input a request to store the first selection correspondence as a rule in the list of rules for transformation;
means for inputting the request to store the correspondence as a rule in the list of rules for transformation, when the user inputs the request to store the first selection as a rule; and
means for displaying a second element for transformation of the first structural description, when the user inputs the request to store the first selection as a rule.

66. An apparatus according to Claim 63, further comprising:

means for displaying an icon for the user to input a request to store the correspondence as a rule in the list of rules for transformation;
means for inputting the request to store the correspondence as a rule in the list of rules for transformation, when the user inputs the request to store the first selection as a rule in the list of rules for transformation; and
means for storing the list of rules for transformation as a map.

67. An apparatus according to Claim 63, further comprising:

means for displaying an icon for the user to input a request to delete the list of rules for transformation;
means for inputting the request to delete the list of rules for transformation, when the user inputs the request to delete the list of rules for transformation; and
means for deleting the list of rules for transformation, when the user inputs a request to delete the list of rules for transformation.

68. An apparatus according to Claim 56, further comprising:

means for displaying the first selection of one of the candidate elements of the second structural description which defines the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description.

69. An apparatus according to Claim 68, wherein the user inputting means comprises:

means for inputting, from the user, a first ordered list of a plurality of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;
the means for displaying the first selection comprises:
means for displaying the first ordered list of the plurality of the candidate elements of the second markup language which defines a correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
the storing means comprises:
means for storing the correspondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description as a rule.

70. An apparatus according to Claim 69, further comprising:

means for obtaining the stored rule;
means for displaying the element for transformation of the first structural description;
means for displaying the first ordered list of elements of the second structural description which defines a cor-

respondence between the element for transformation of the first structural description and the first ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;

means for displaying the list of candidate elements of the second structural description;

means for inputting, from the user, a second ordered list of a plurality of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the second ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and

means for storing the correspondence between the element for transformation of the first structural description and the second ordered list of the plurality of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description as a rule.

71. An apparatus according to Claim 69, further comprising:

means for displaying an icon for the user to input a request to clear the first ordered list which is being displayed;

means for inputting the request to clear the first ordered list which is being displayed, when the user inputs the request to clear the first ordered list which is being displayed; and

means for clearing the first ordered list which is being displayed, when the user inputs the request to clear the first ordered list which is being displayed.

72. An apparatus according to Claim 56, further comprising:

means for displaying an attribute of the first selection of one of the candidate elements of the second structural description which corresponds to a transformation of the element of the first structural description to the second structural description, for assignment of an attribute value of the second structural description;

means for displaying a plurality of icons representing sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed;

means for displaying the element of the first structural description;

means for displaying an attribute list of the element of the first structural description;

means for inputting a user input of a selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed; and

an input processing means for processing the user input of the selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed, when the user inputs the selection of sources for obtaining the attribute value to be assigned to the attribute of the first selection which is being displayed.

73. An apparatus according to Claim 72, wherein the input processing means further comprises:

means for assigning a null value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates no source is to be used.

74. An apparatus according to Claim 72, wherein the input processing means further comprises:

means for assigning a system value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a system value is to be used.

75. An apparatus according to Claim 72, wherein the input processing means further comprises:

means for assigning a first structural description attribute value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a first structural description attribute value source is to be used.

76. An apparatus according to Claim 72, wherein the input processing means further comprises:

an assigning means for assigning a first structural description content value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a first struc-

tural description content value source is to be used.

77. An apparatus according to Claim 72, wherein the input processing means further comprises:

means for assigning a user input value to the attribute of the first selection which is being displayed, when the selection of sources which has been input by the user indicates a user input value source is to be used.

78. An apparatus according to Claim 77, wherein the assigning means comprises:

means for displaying a text input area for the user to input a value to be assigned;
means for inputting the value entered by the user in the text input area; and
means for assigning the value input by the user to the attribute of the first selection which is being displayed.

79. An apparatus according to Claim 56, wherein the list displaying means further comprises:

means for displaying a candidate for requesting removal of the element for transformation of the first structural description in the transformation; and
means for displaying a candidate for requesting ignoring of the element for transformation of the first structural description in the transformation.

80. A computer program product including a computer readable medium for providing a graphical user interface for creating a mapping of a first structural description to a second structural description, comprising:

element displaying means for displaying an element for transformation of a first structural description;
list displaying means for displaying a list of candidate elements of a second structural description;
user inputting means for inputting, from a user, a first selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and
storing means for storing the correspondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description as a rule.

81. A computer program product according to Claim 80, wherein the first structural description includes ISO/IEC 9070 public identifier format, the second structural description includes an operating system file name format, and the element displaying means comprises:

means for displaying the element for transformation which includes an element of the ISO/IEC 9070 public identifier format; and
the list displaying means comprises:
means for displaying the list of candidate elements which includes a list of operating system file name candidate elements.

82. A computer program product according to Claim 80, wherein the first structural description includes a first database variable name format, the second structural description includes a second database variable name format, and the element displaying means comprises:

means for displaying the element for transformation which includes an element of the first database variable name format; and
the list displaying means comprises:
means for displaying the list of candidate elements which includes a list of second database variable name format candidate elements.

83. A computer program product according to Claim 80, further comprising:

means for obtaining the stored rule;
means for displaying the element for transformation of the first structural description;
means for displaying the first selected element of the second structural description which defines a corre-

spondence between the element for transformation of the first structural description and the first selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description;

means for displaying the list of candidate elements of the second structural description;

means for inputting, from the user, a second selection of one of the candidate elements of the second structural description which defines a correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description for a transformation of the element of the first structural description to the second structural description; and

means for storing the correspondence between the element for transformation of the first structural description and the second selection of one of the candidate elements of the second structural description as a rule.

84. A computer program product according to Claim 80, further comprising:

means for displaying an icon for the user to input a request to clear the first selection which is being displayed;

means for inputting the request to clear the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed; and

means for clearing the first selection which is being displayed, when the user inputs the request to clear the first selection which is being displayed.

85. A computer program product according to Claim 80, wherein the first structural description includes a first markup language, the second structural description includes a second markup language, and the element displaying means comprises:

means for displaying the element for transformation which includes an element of the first markup language; and

the list displaying means comprises:

means for displaying the list of candidate elements which includes a list of second markup language candidate elements.

86. A computer program product according to Claim 85, wherein the first markup language includes SGML, the second markup language includes HTML, and the element displaying means further comprises:

means for displaying the element for transformation which includes an element of SGML; and

the list displaying means further comprises:

means for displaying the list of candidate elements which includes a list of HTML candidate elements.

```
22 <!--    sample1.dtd   >      -->
24 <!ELEMENT t - - (t1?, t2?, t3?, t4?) ->
26 <!ELEMENT t1 - - CDATA>
28 <!ATTLIST t1 name CDATA #REQUIRED>
30 <!ELEMENT t2 - - CDATA>
32 <!ELEMENT t3 - - CDATA>
34 <!ELEMENT t4 - - CDATA>
```

Fig. 1A

```
42 t--> <html><title>Title</title>
44 t1--><H3><A NAME="the source is t1's name">
46 t2--> <P>
48 t3--> <P>
50 t4--> <P><A HREF="# the source is t1's name">
```

Fig. 1B

```
60 <!DOCTYPE t system "sample1.dtd">
62 <t>
64 <t1 name="hilarry">1.  Hi Larry</t1>
66 <t2> This is the most fun I have ever had.</t2>
68 <t3> It must be great for you as well.</t3>
70 <t4> (Back to the Hi Larry greeting.)</t4>
72 </t>
```

Fig. 1C

```
80 <DOCTYPE HTML Public "-//W3C//DTD HTML 3.2 Final//EN">
82 <html>
84 <title>Title</title>
86 <H3><A name="hilarry">1.  Hi Larry</A></H3>
88 <P> This is the most fun I have ever had.</P>
90 <P> It must be great for you as well.</P>
92 <P><A HREF="#hilarry"> (Back to the Hi Larry greeting.)</A></P>
94 </html>
```

Fig. 1D

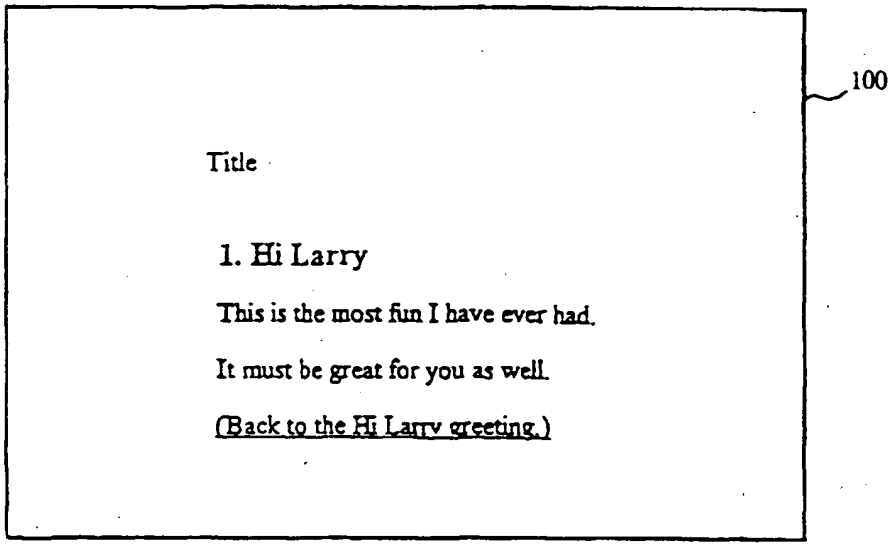


Fig. 2

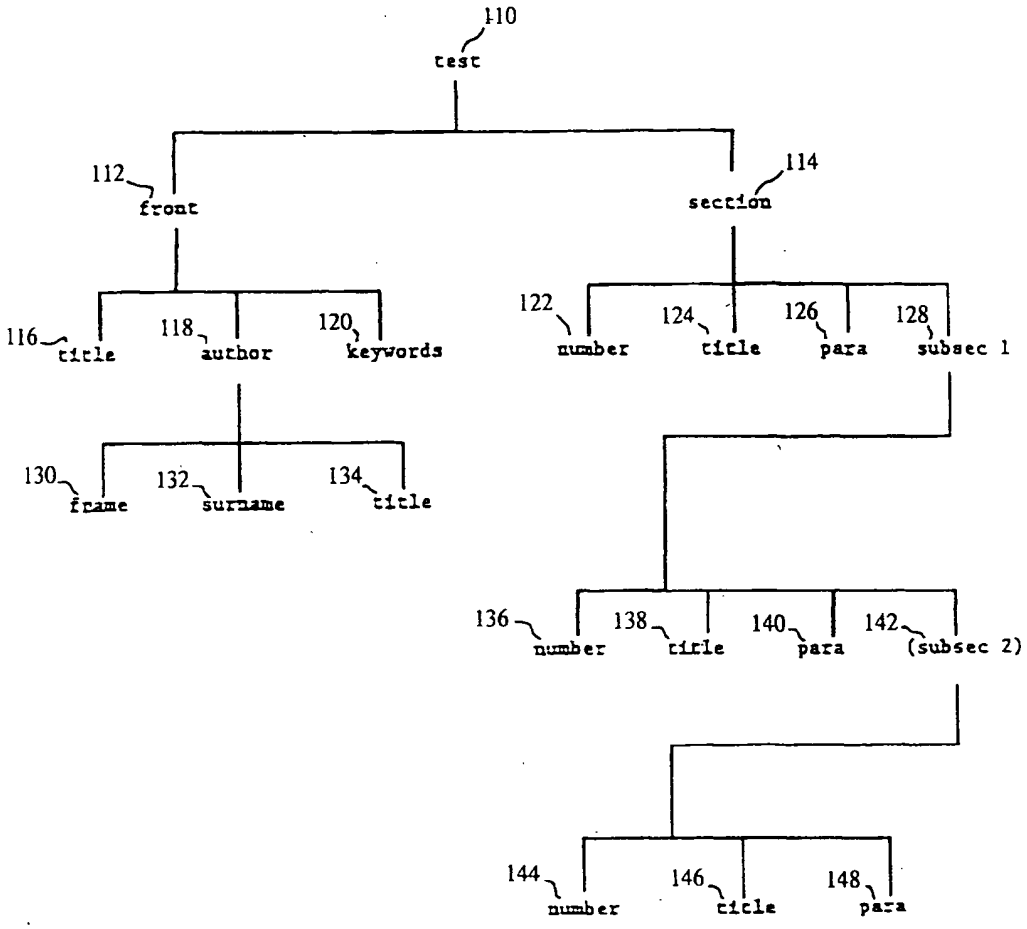


Fig. 3A

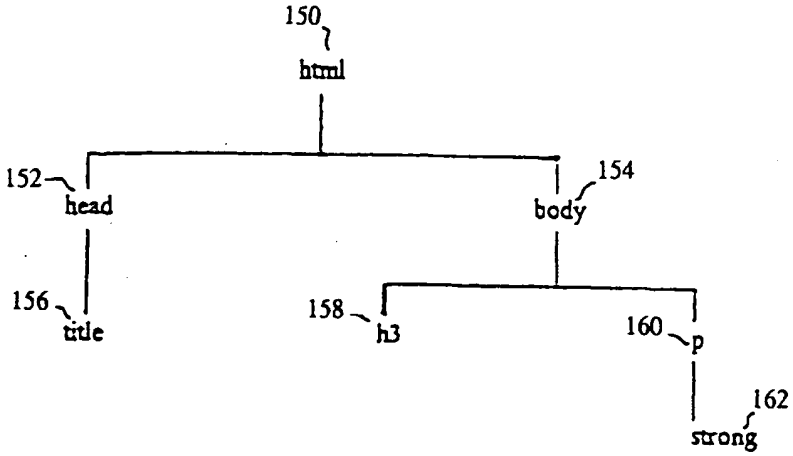


Fig. 3B

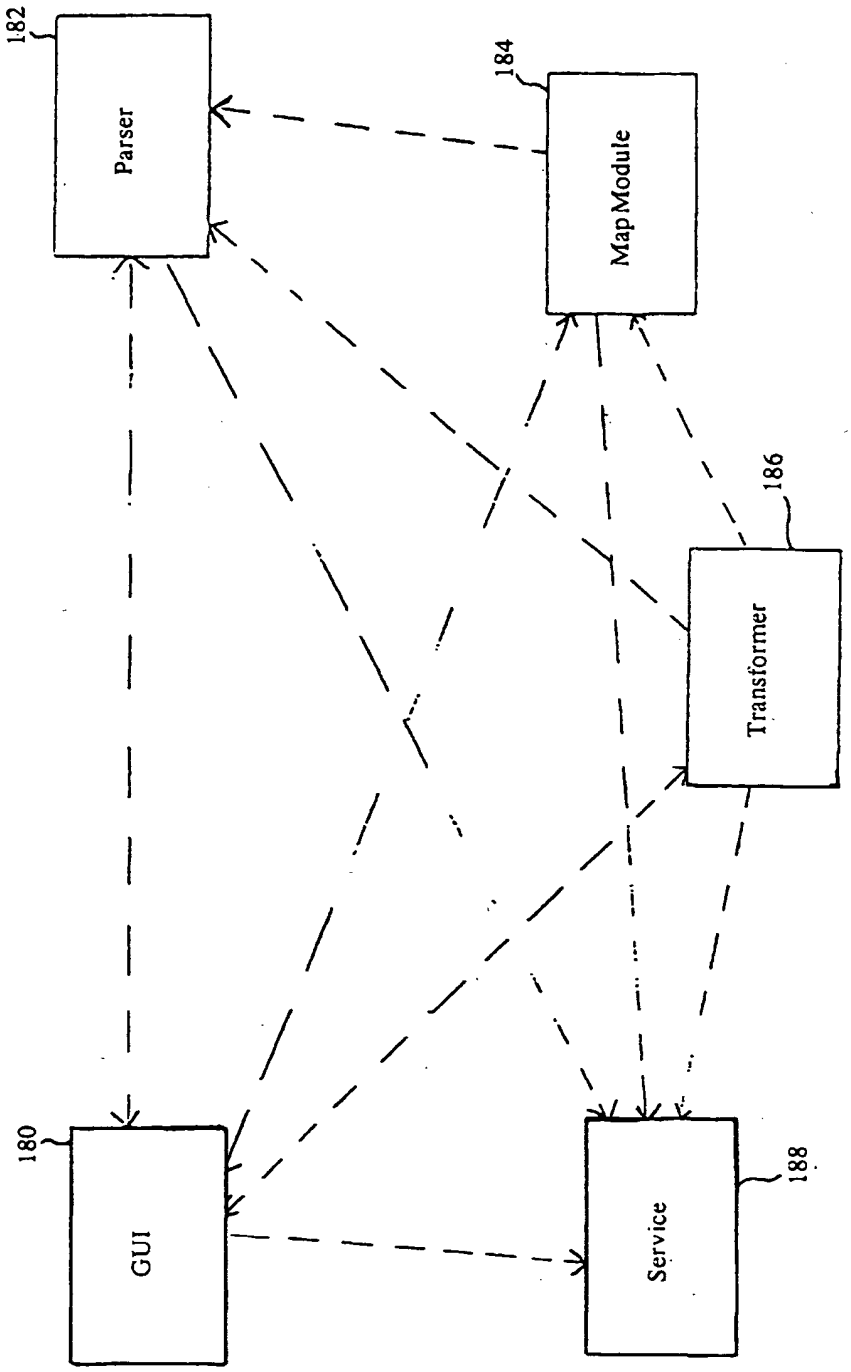


Fig. 4

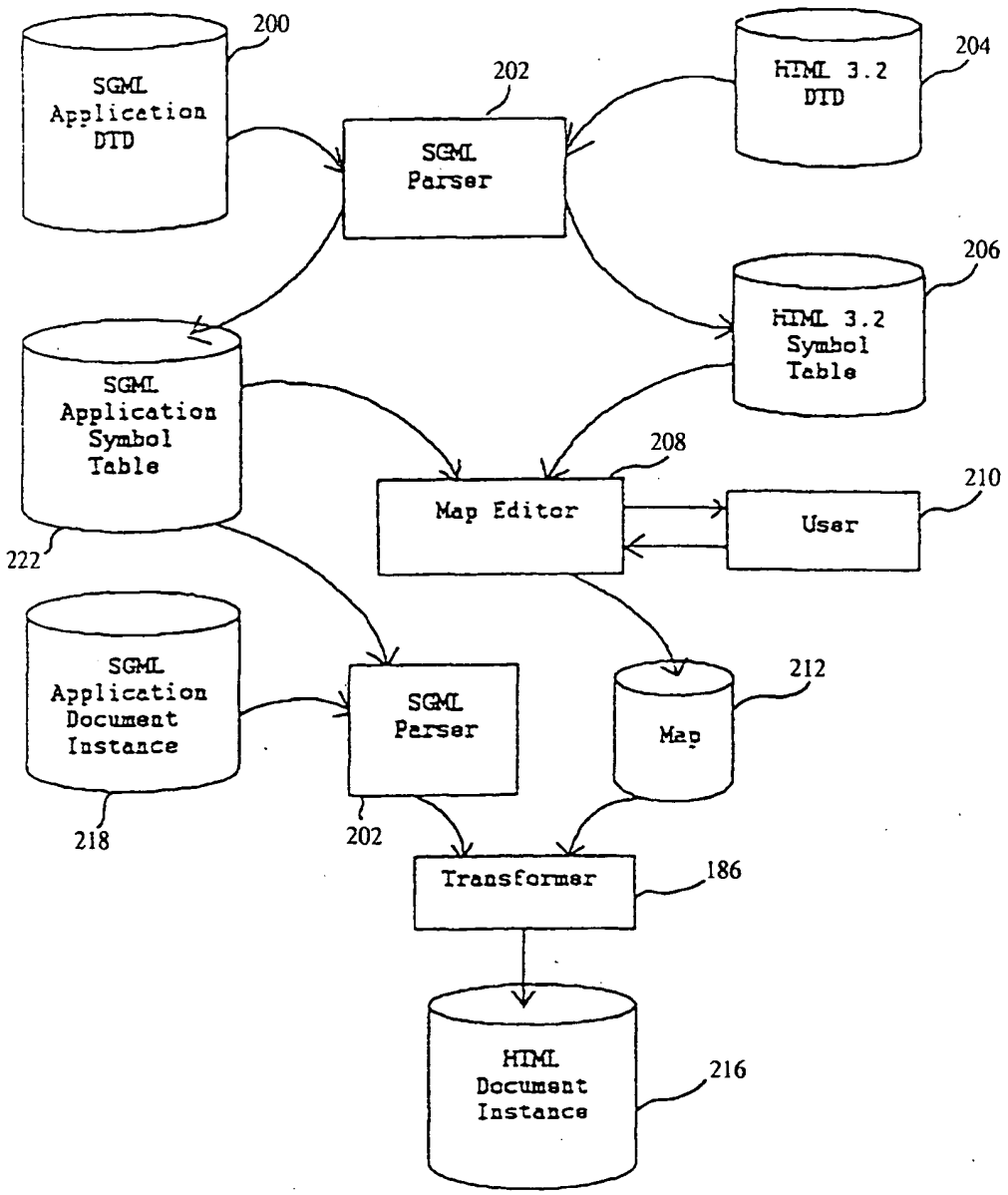


Fig. 5

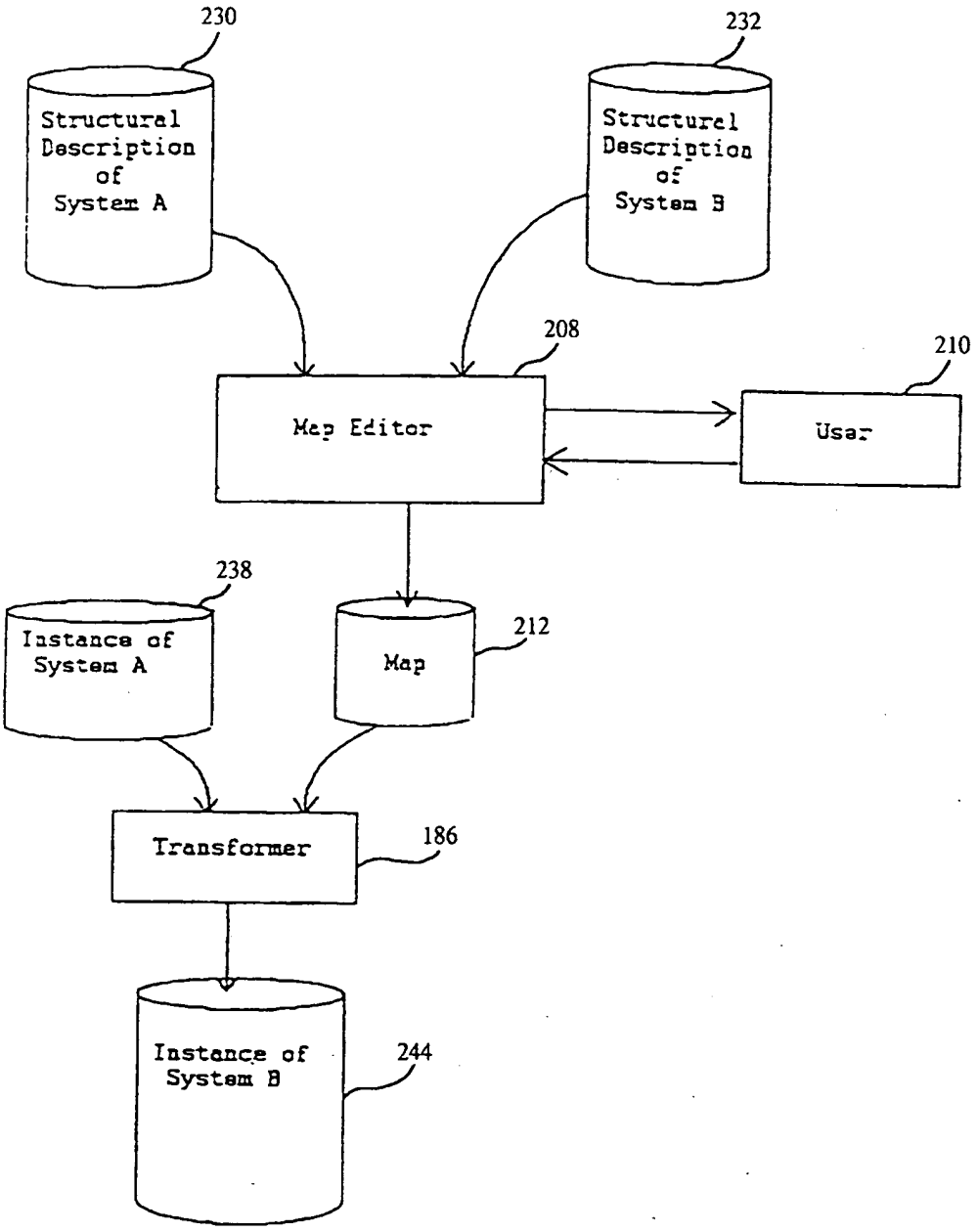


Fig. 6A

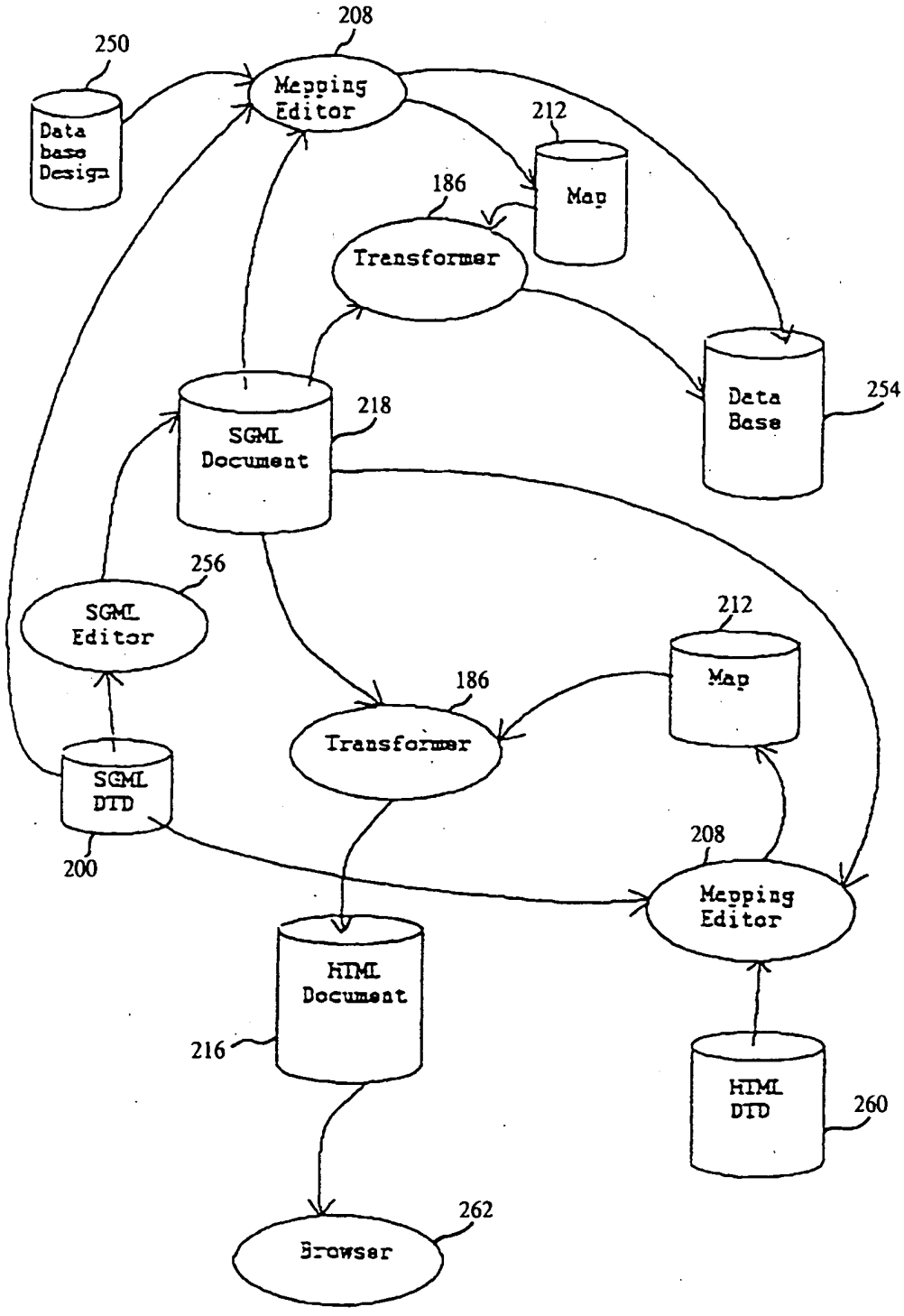


Fig. 6B

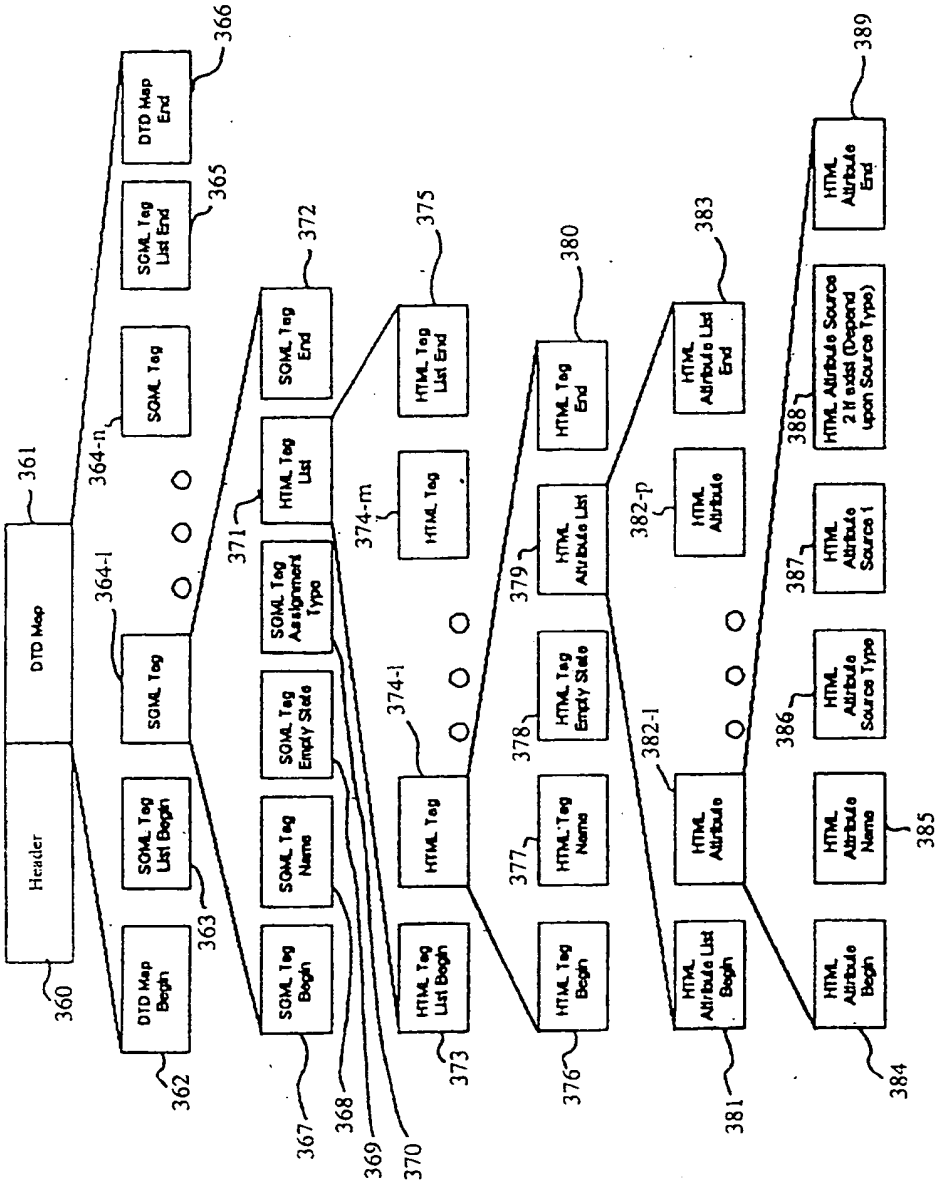


Fig. 7

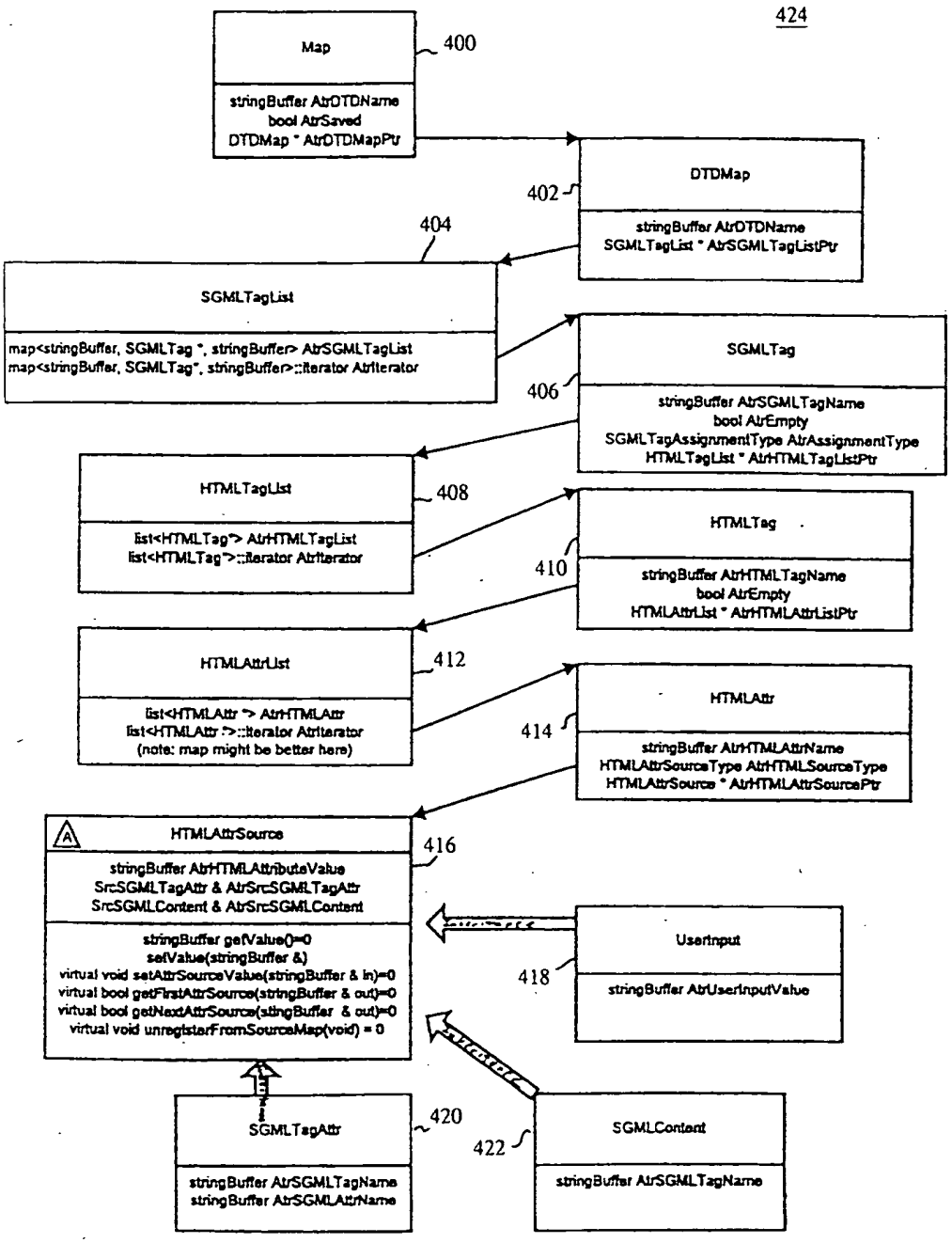


Fig. 8A

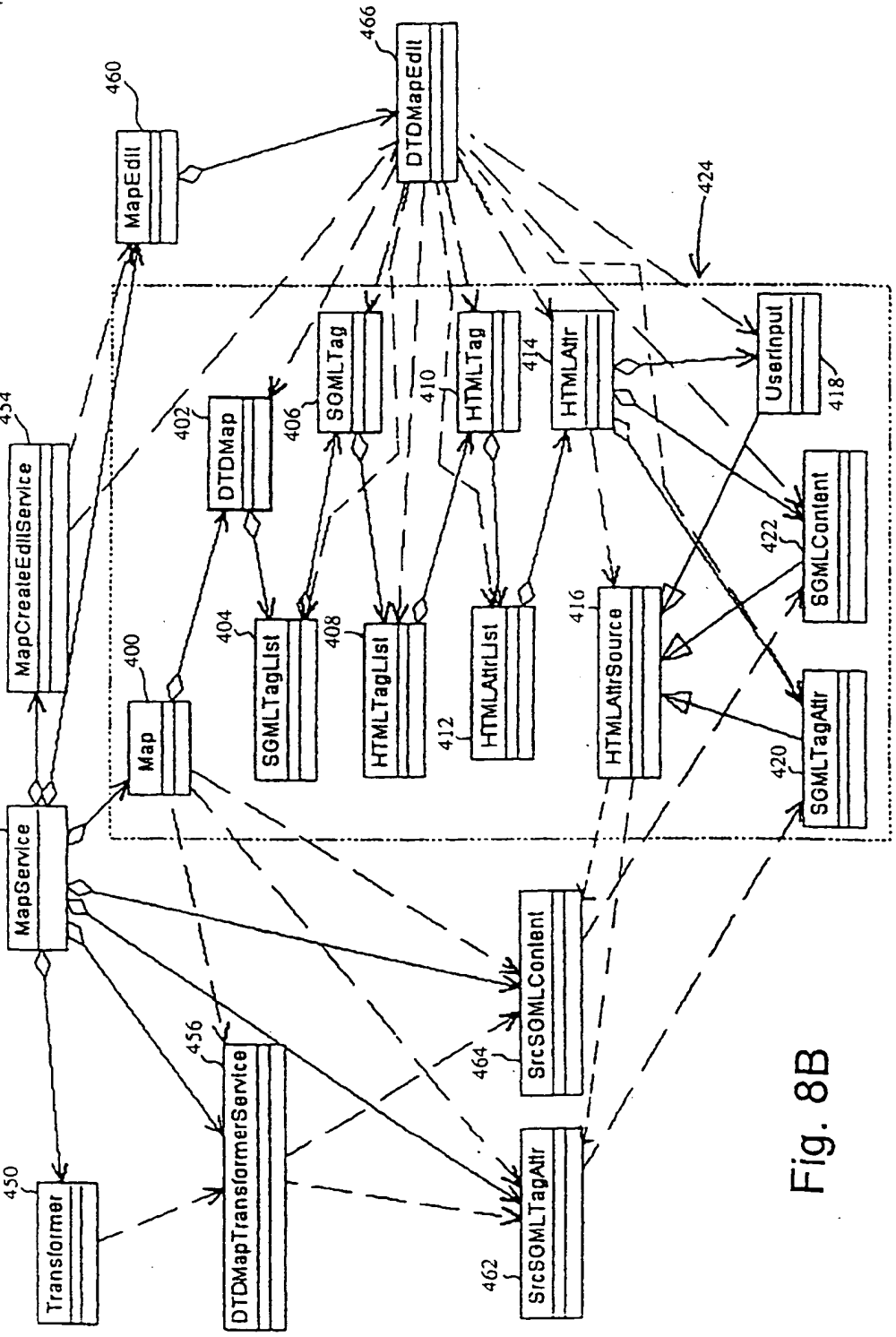


Fig. 8B

462

SrcSGMLTagAttr
stringBuffer AtrCurrentSGMLTagName stringBuffer AtrCurrentAttribute stringBuffer AtrCurrentHTMLAttributeSourceValue stringBuffer AtrTagAttrKey map<stringBuffer, list<SGMLTagAttr *>, stringBuffer> AtrKeyOfSGMLTagAndAttribute
void registerSGMLTagNameAndAttributeName(SGMLTagAttr *) void unregisterTagAttrKeyAndMapEntry(SGMLTagAttr *) void setValueForAttributeOfTag (stringBuffer & Tag, stringBuffer & Attribute, stringBuffer & value) void reset(void)

Fig. 8C(1)

464

SrcSGMLContent
stringBuffer AtrCurrentSGMLTagName stringBuffer AtrCurrentHTMLAttributeSourceVal map<stringBuffer, list<SGMLContent *>, stringBuffer> AtrTableWithSGMLTagKey
void registerSGMLTagName(SGMLContent *) void unregisterSGMLTagName(SGML Content *) void setValueForTag(stringBuffer & Tag, stringBuffer & value) void reset(void)

Fig. 8C(2)

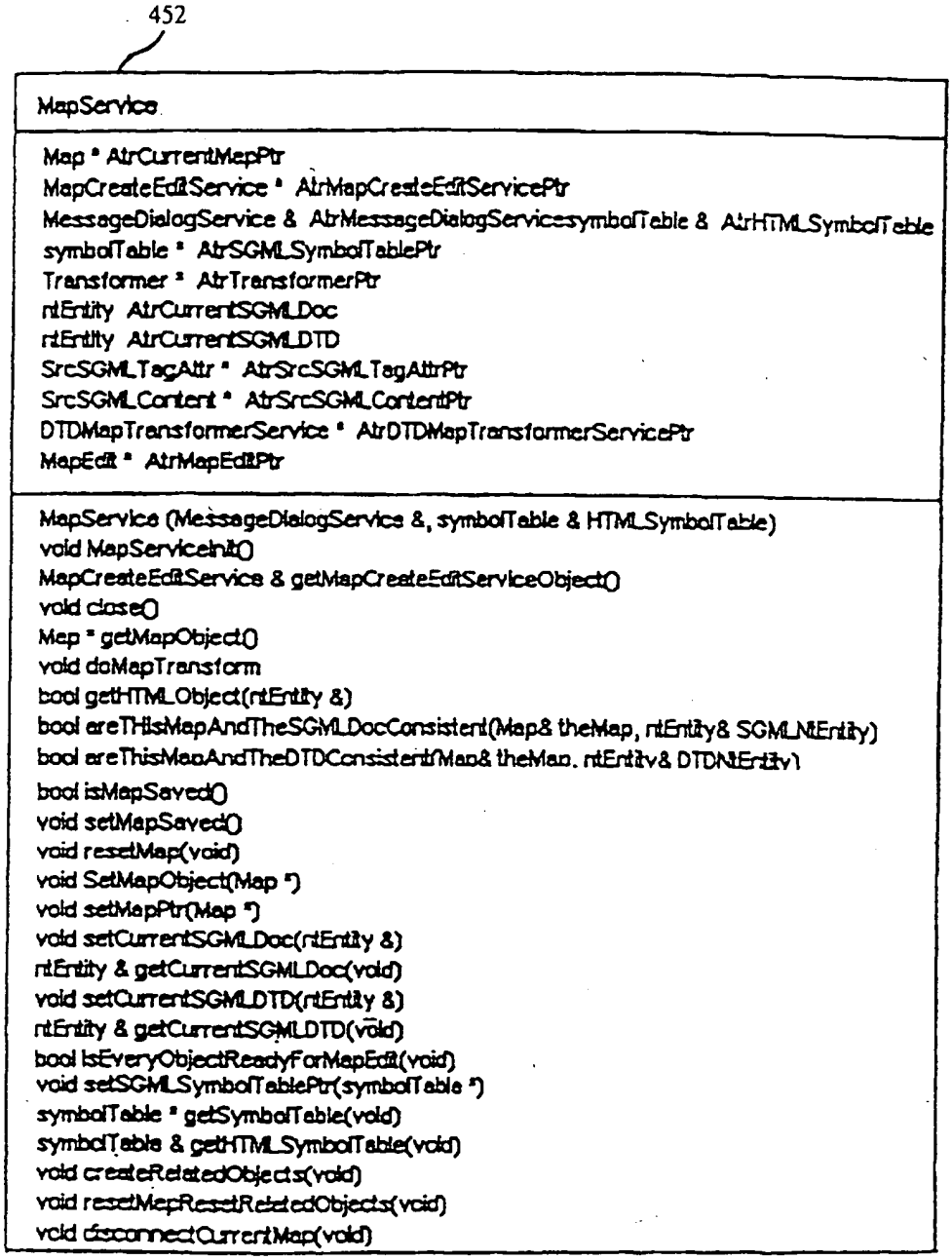


Fig. 8C(3)

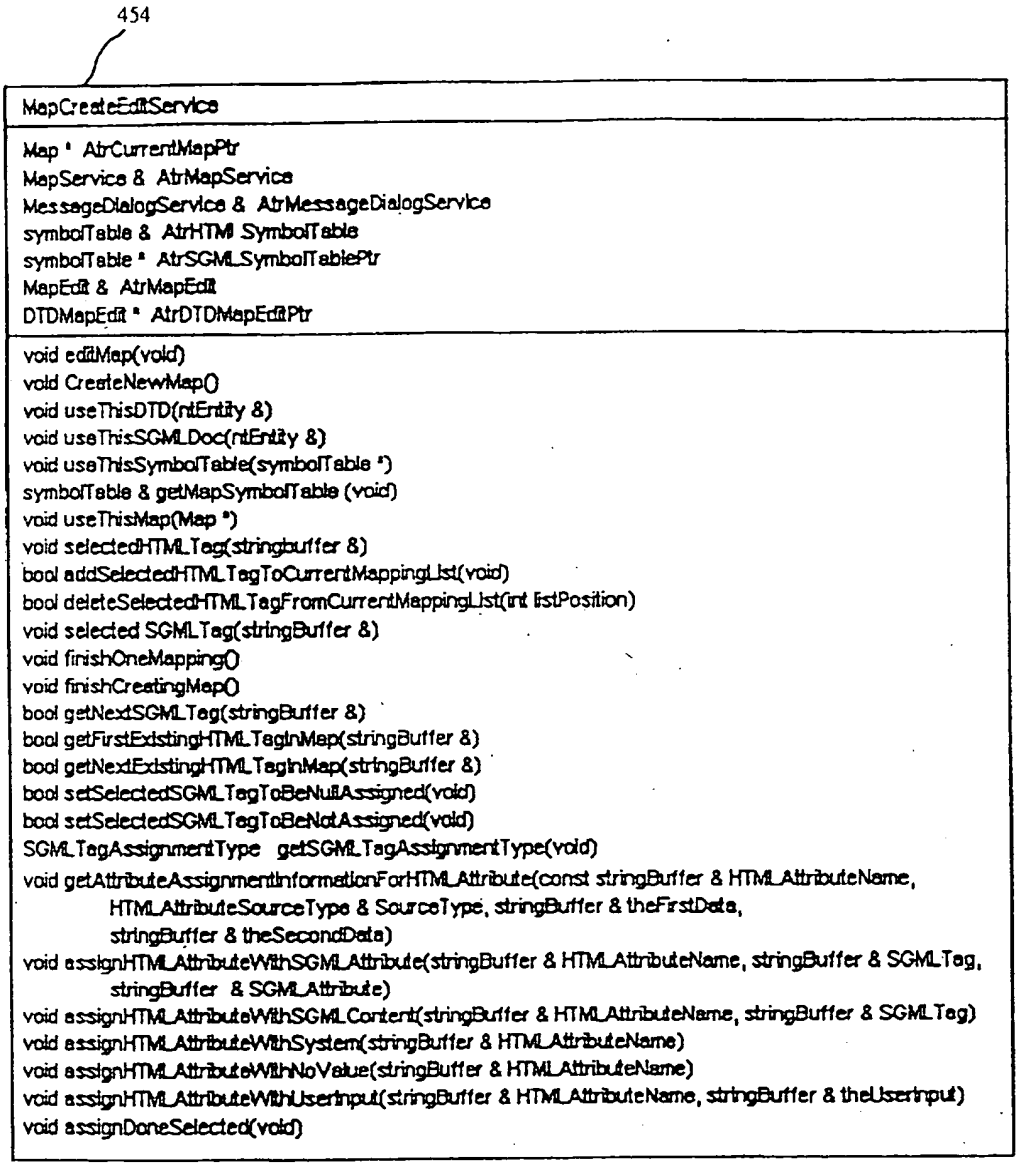


Fig. 8C(4)

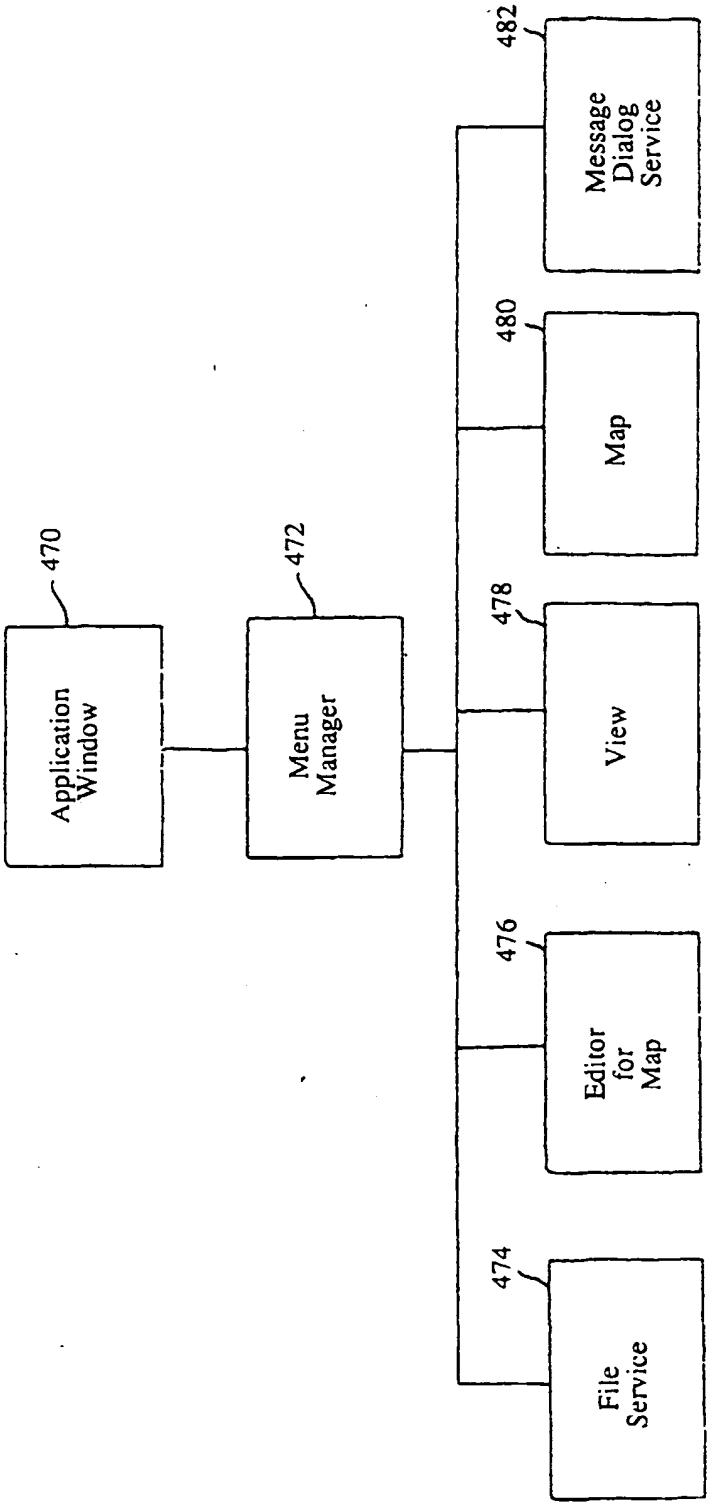


Fig. 9

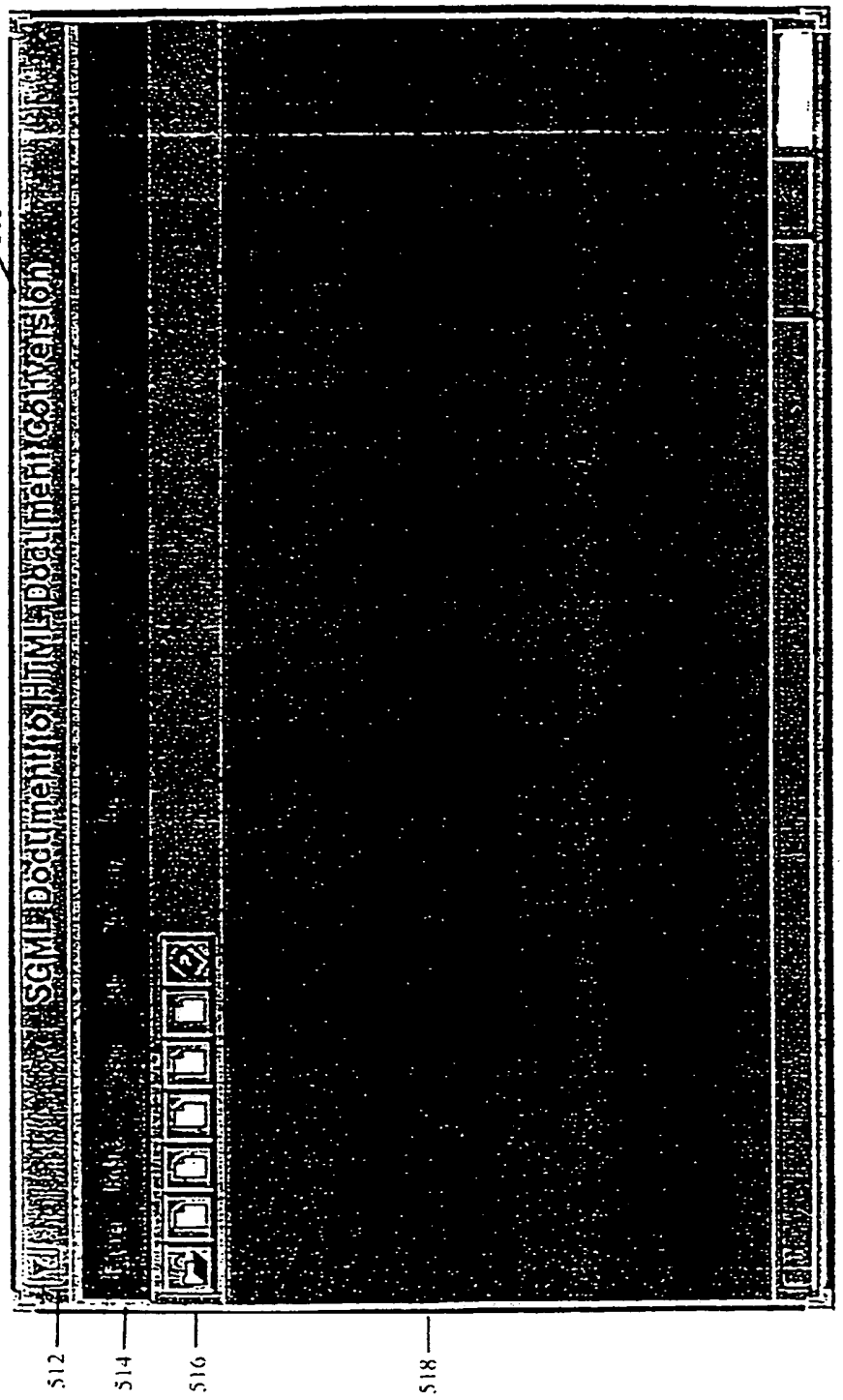


Fig. 10

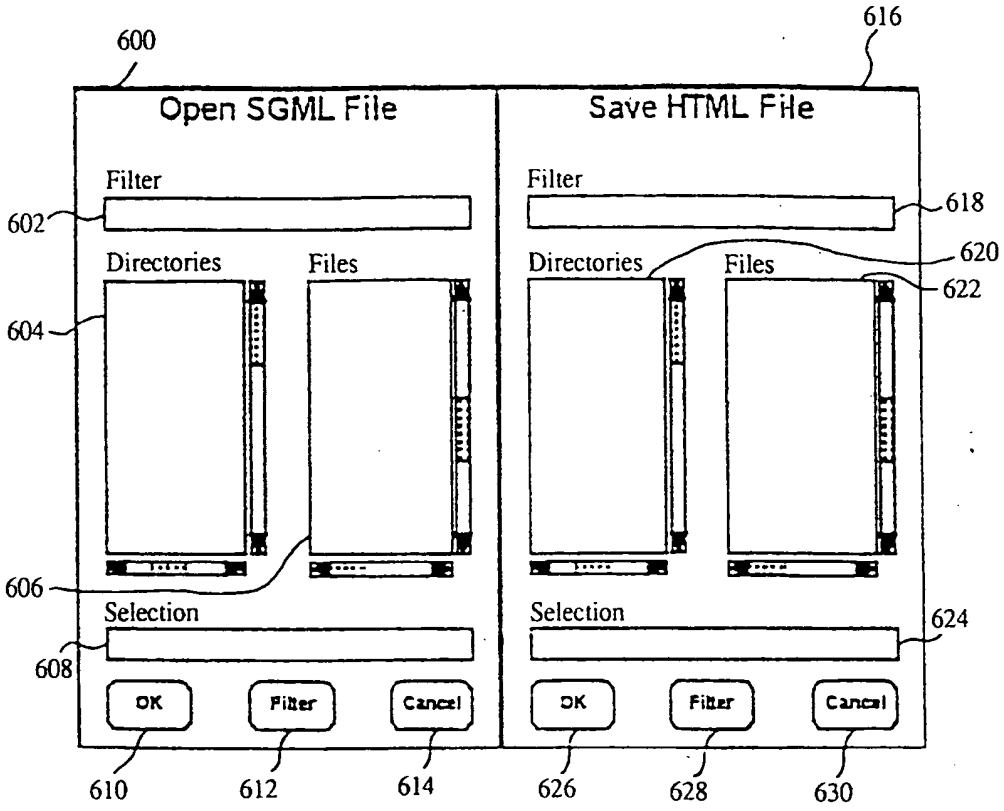


Fig. 11

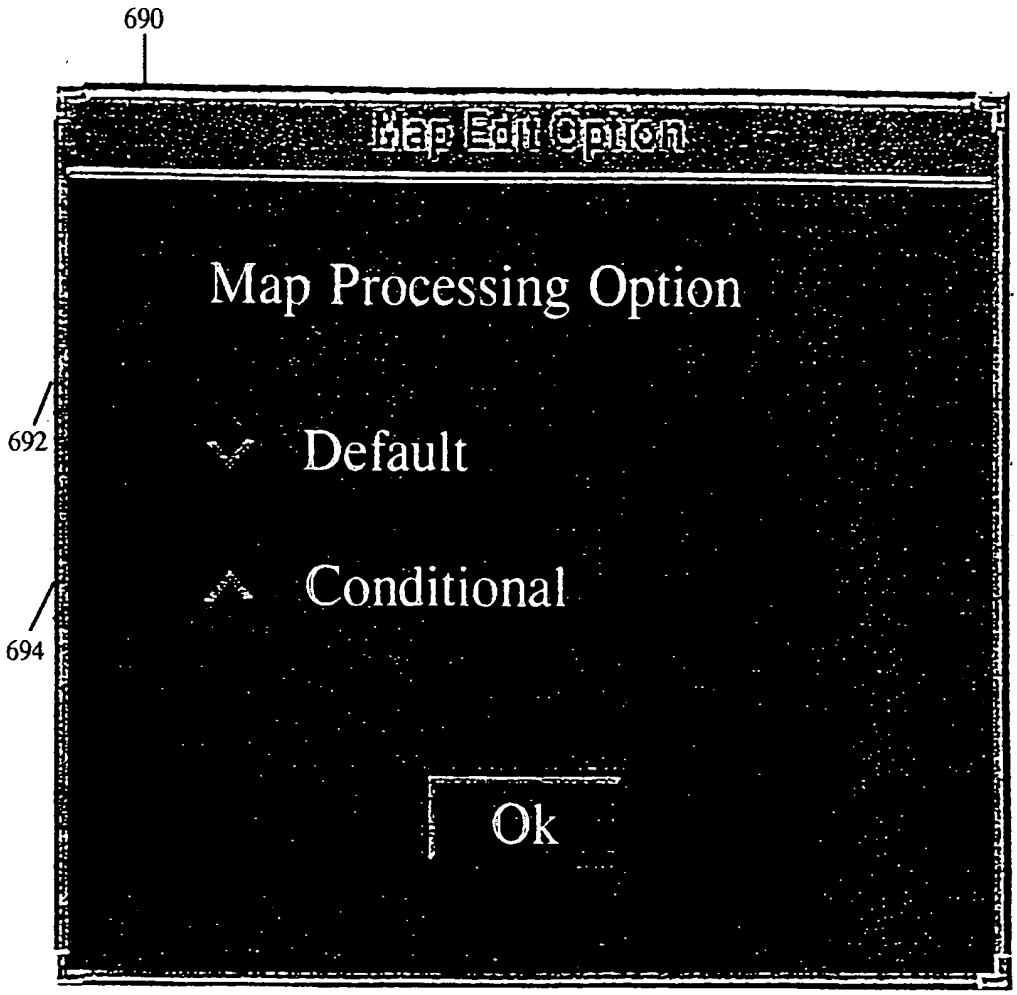


Fig. 12A

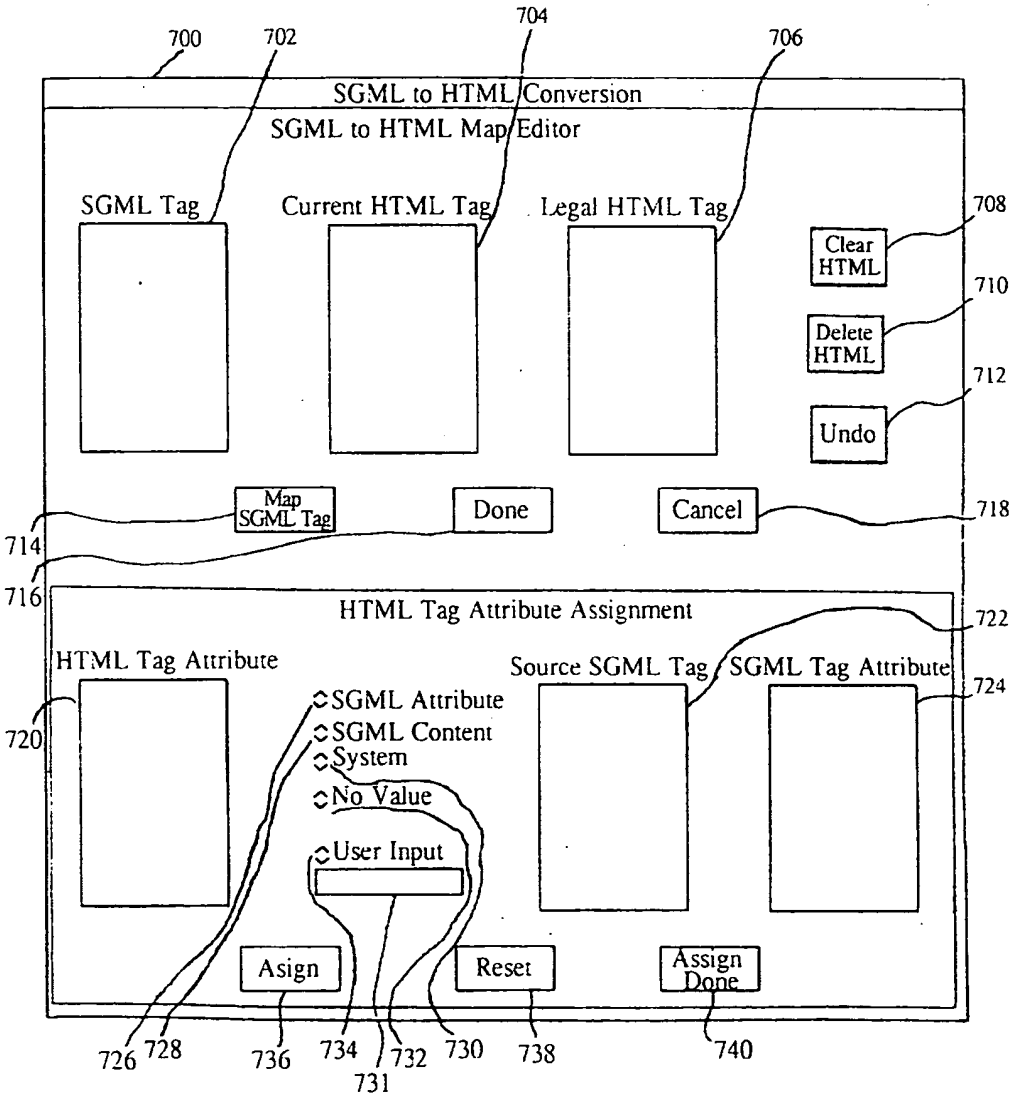


Fig. 12B

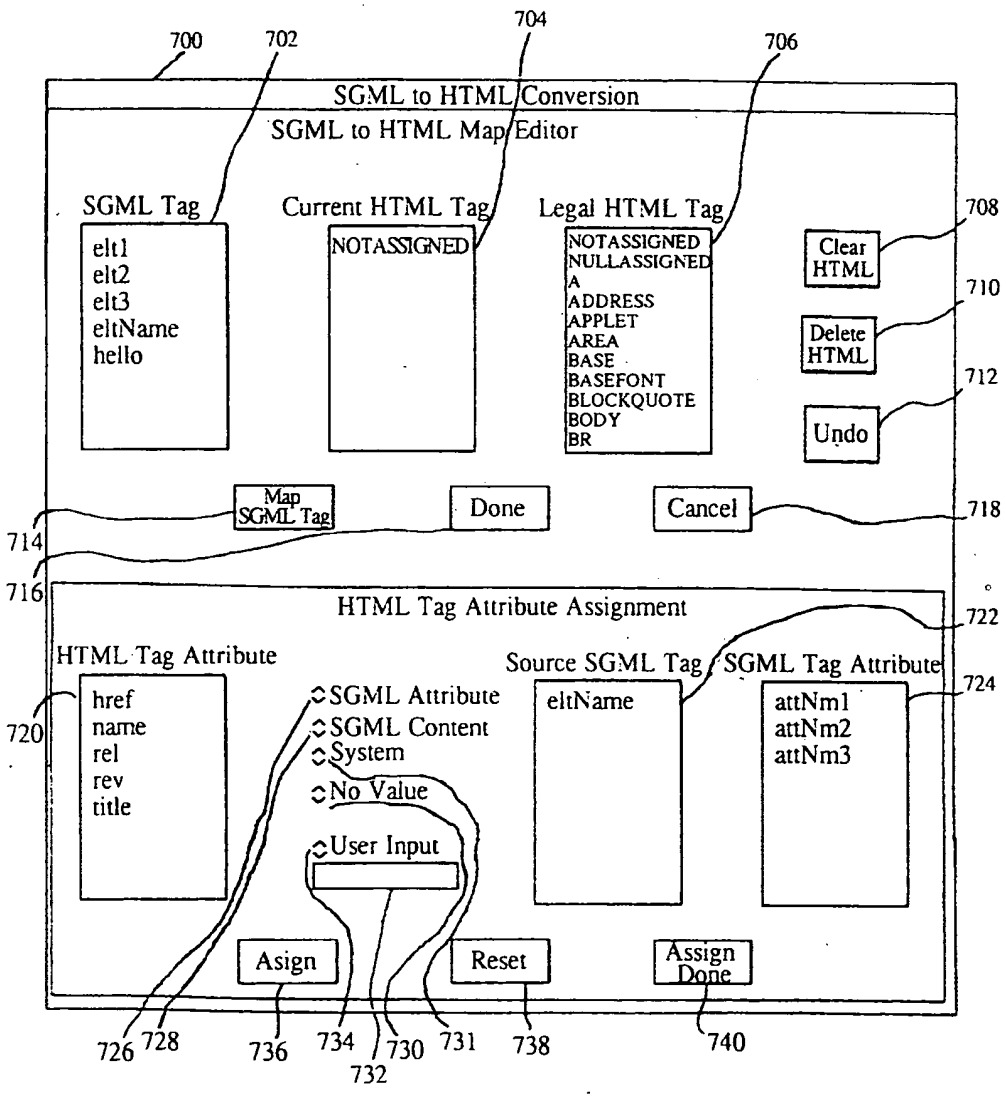


Fig. 12C

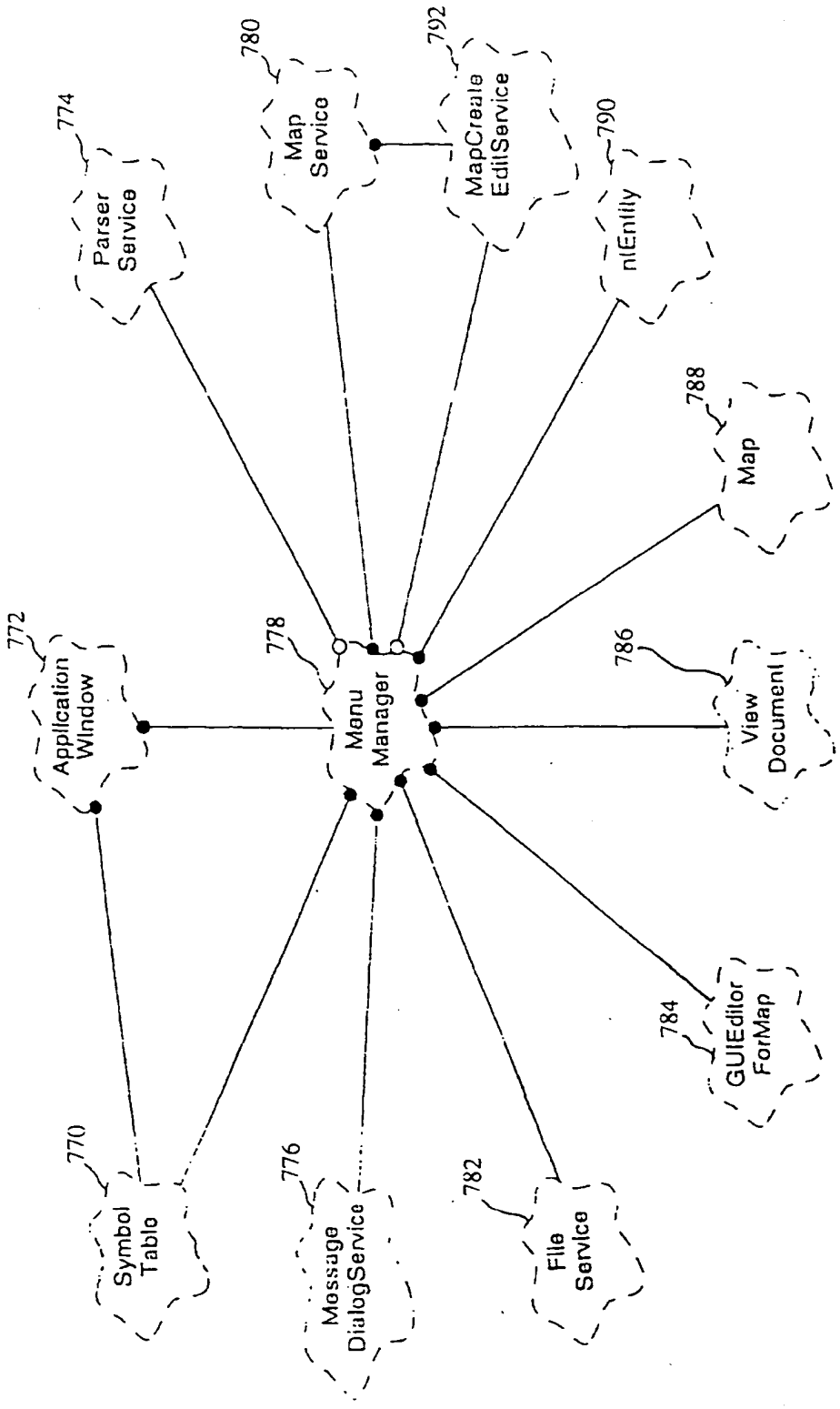


Fig. 13

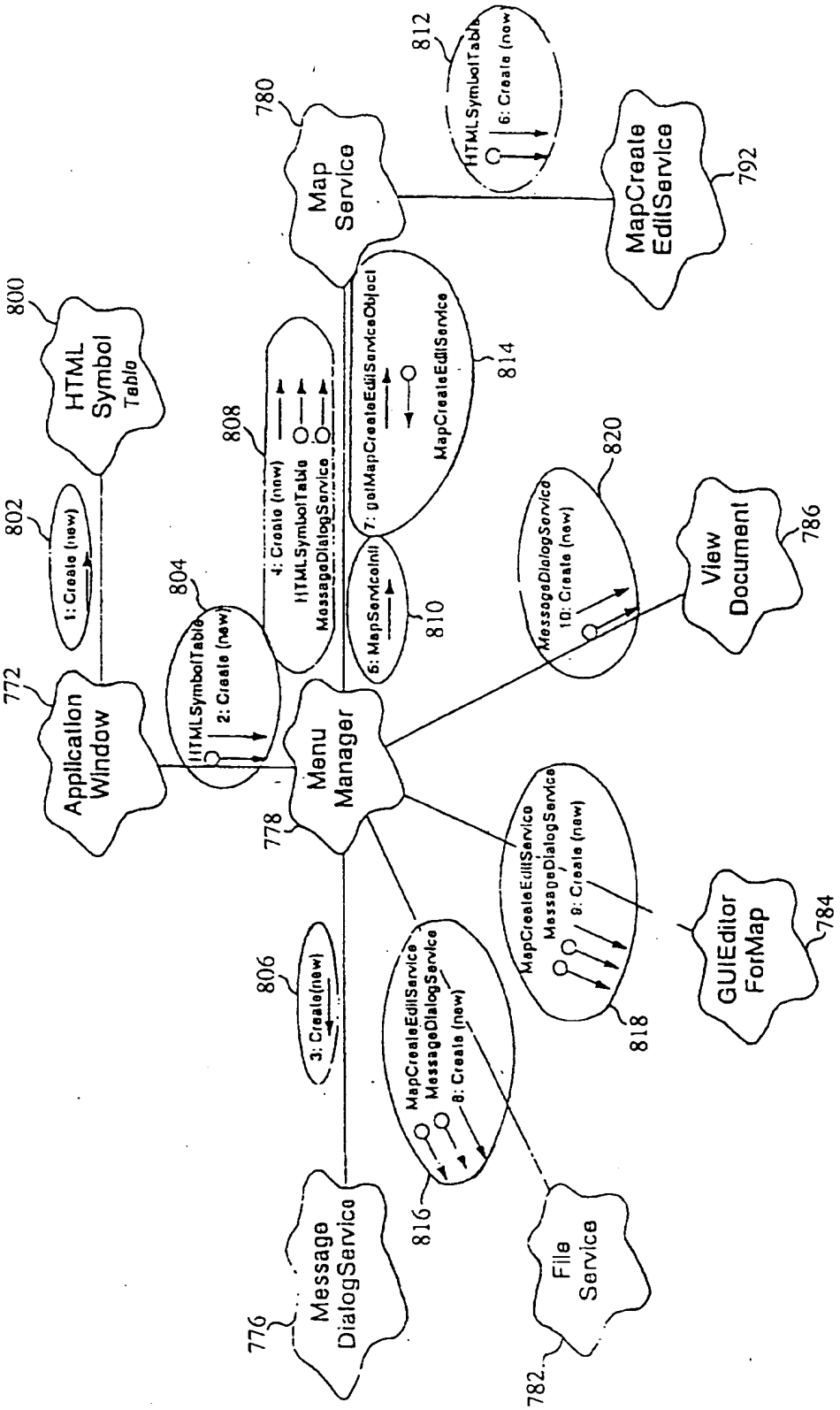


Fig. 14

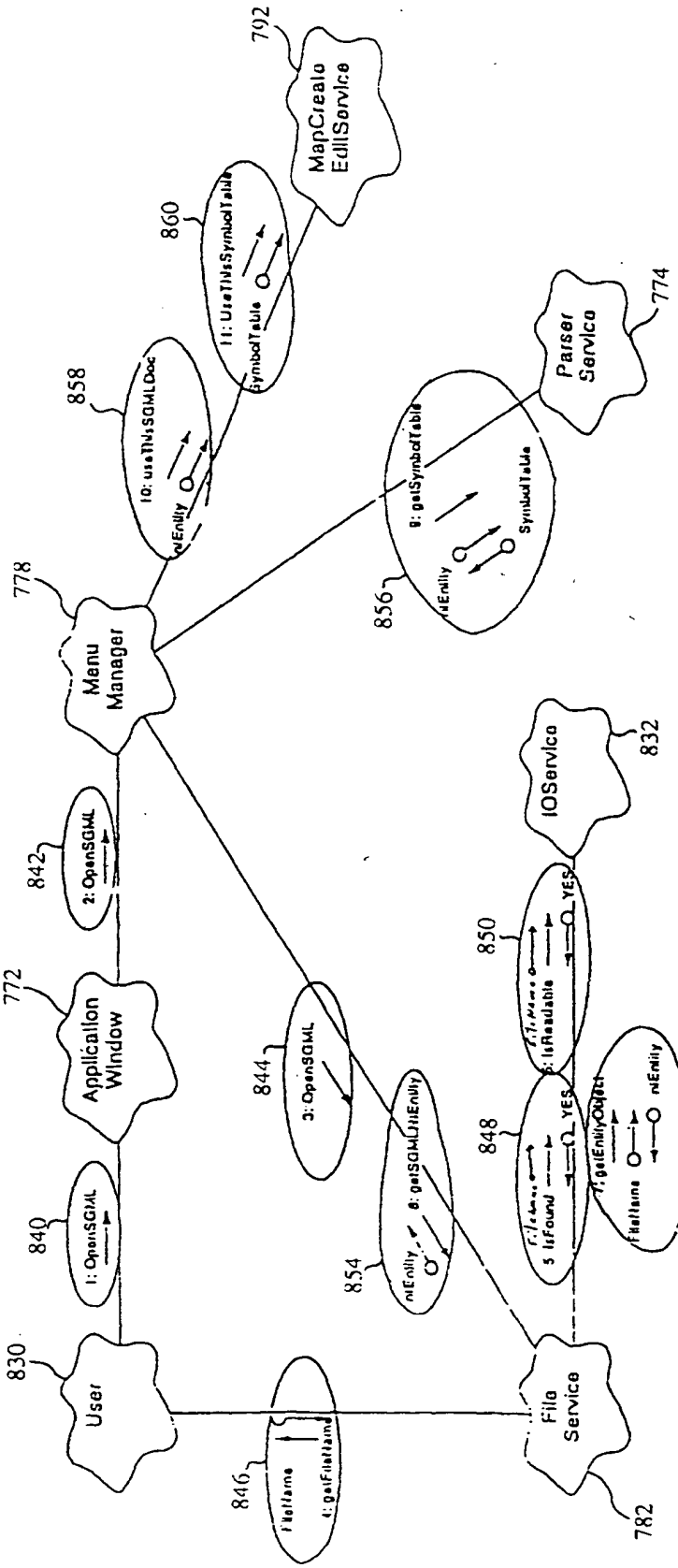


Fig. 15

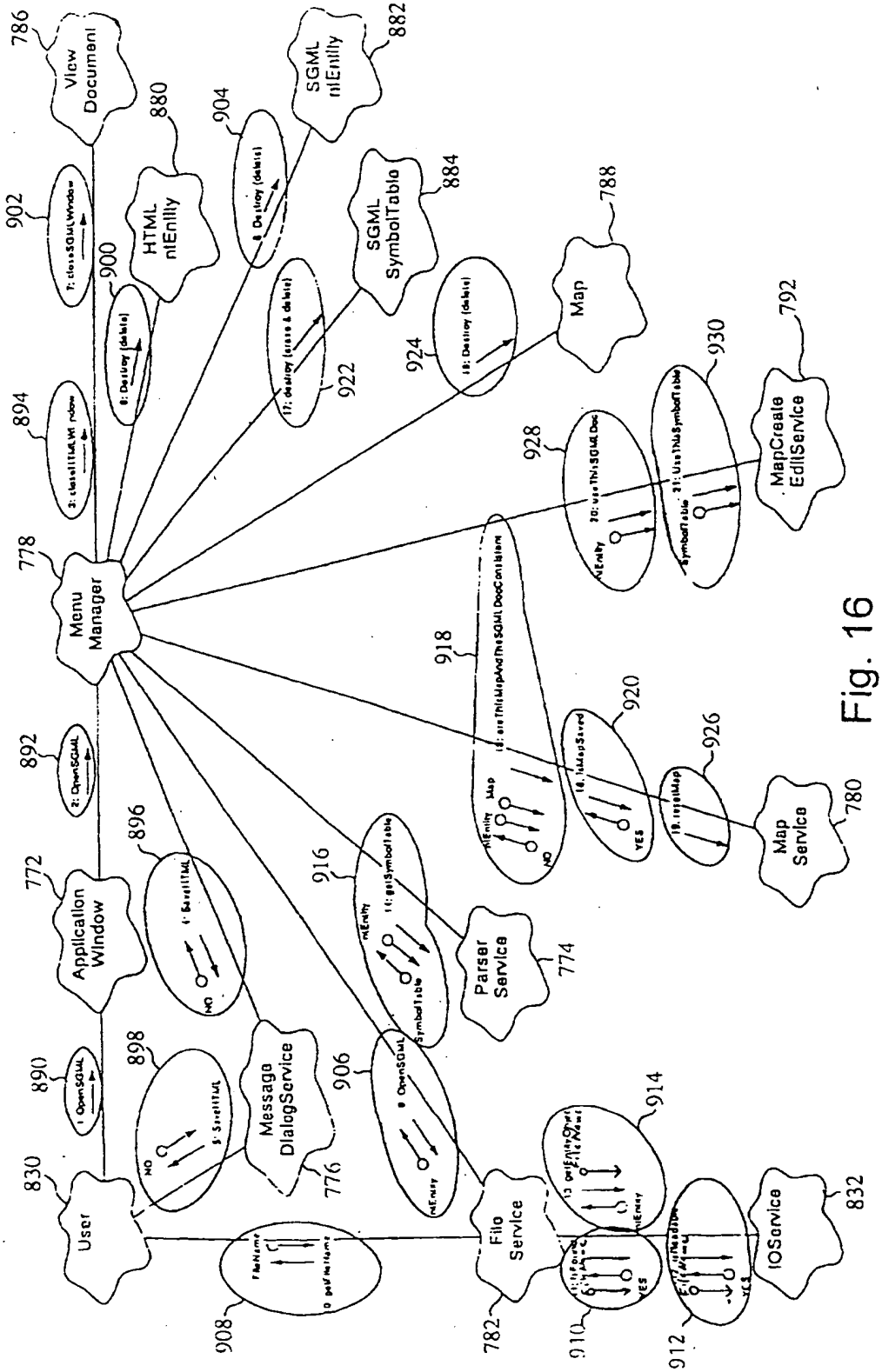


Fig. 16

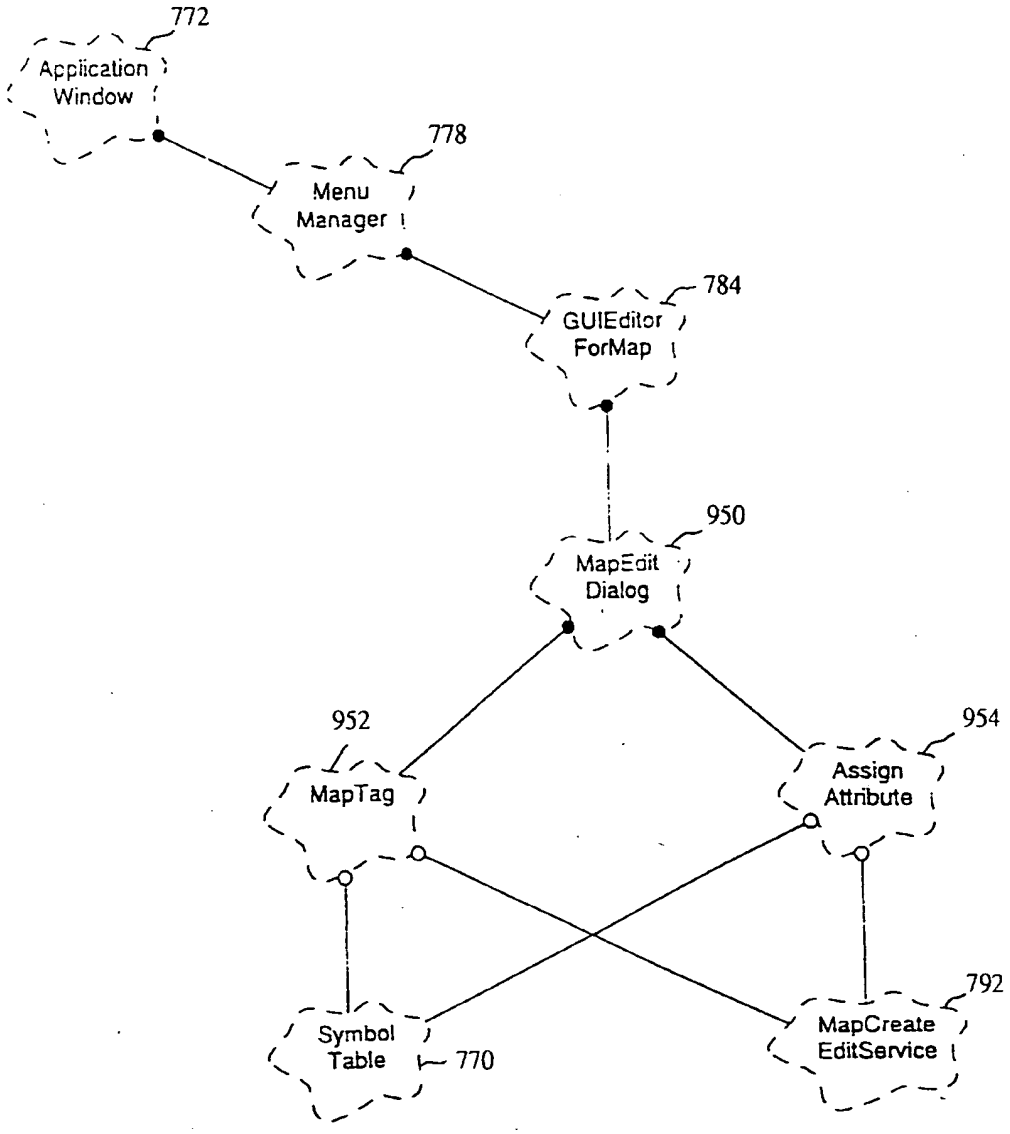


Fig. 17

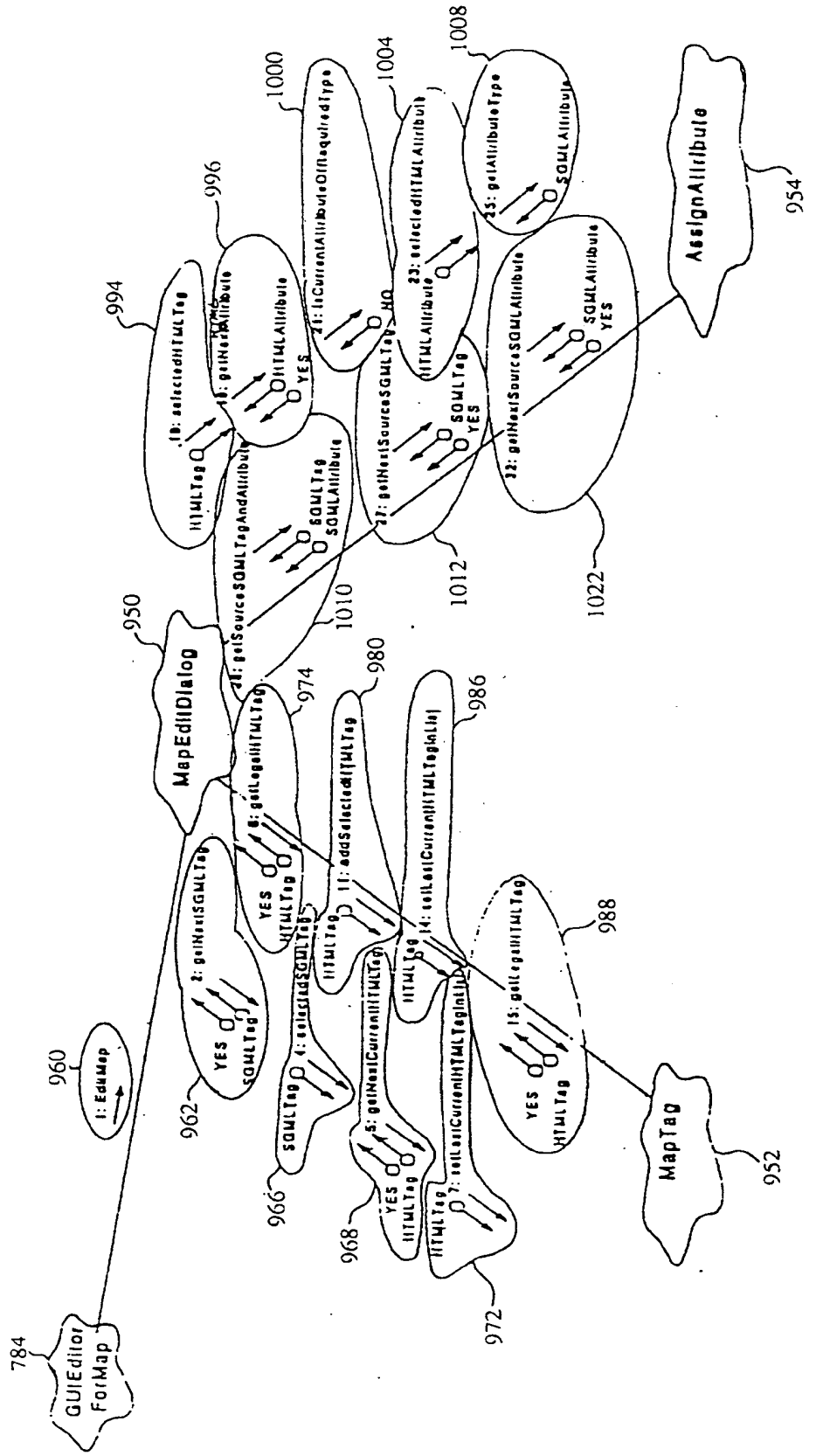


Fig. 18A(1)

Fig. 18A(2)

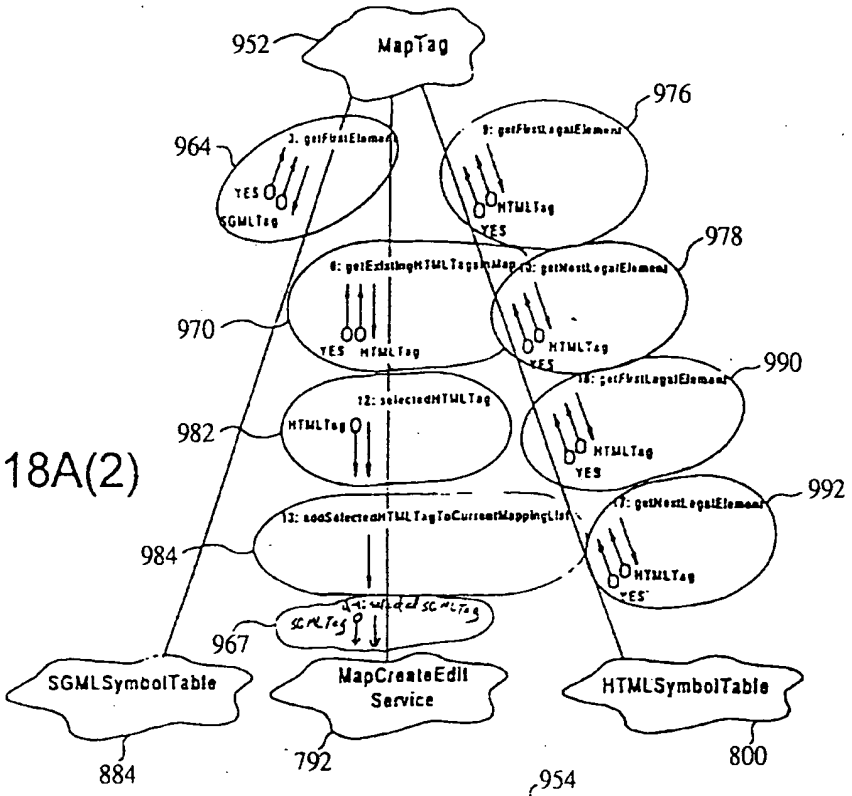
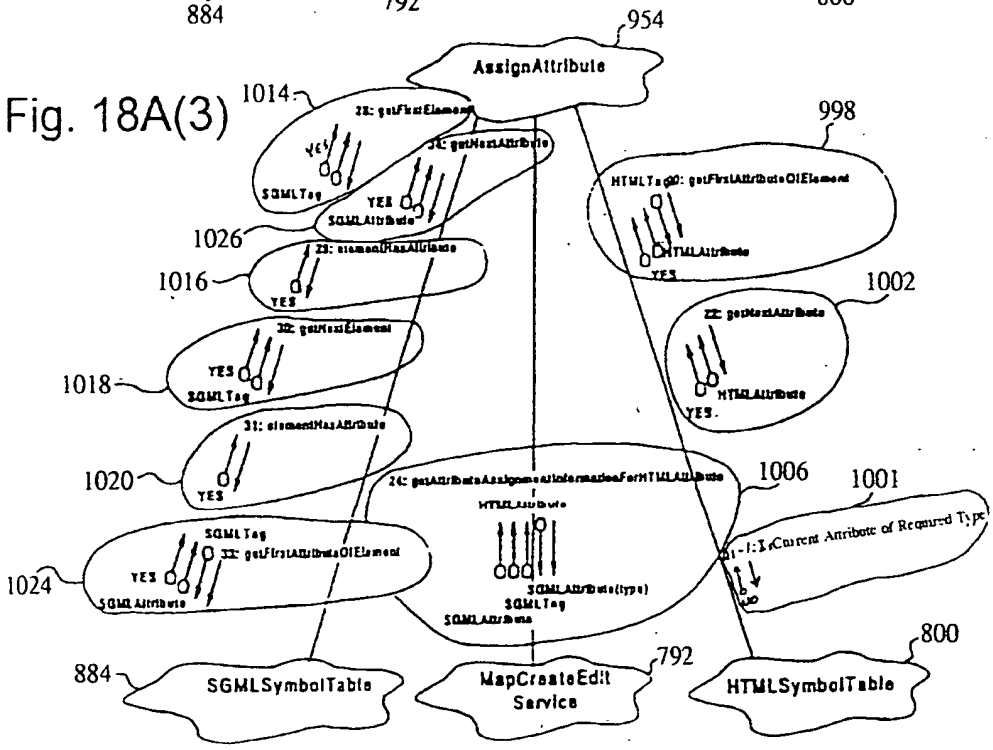


Fig. 18A(3)



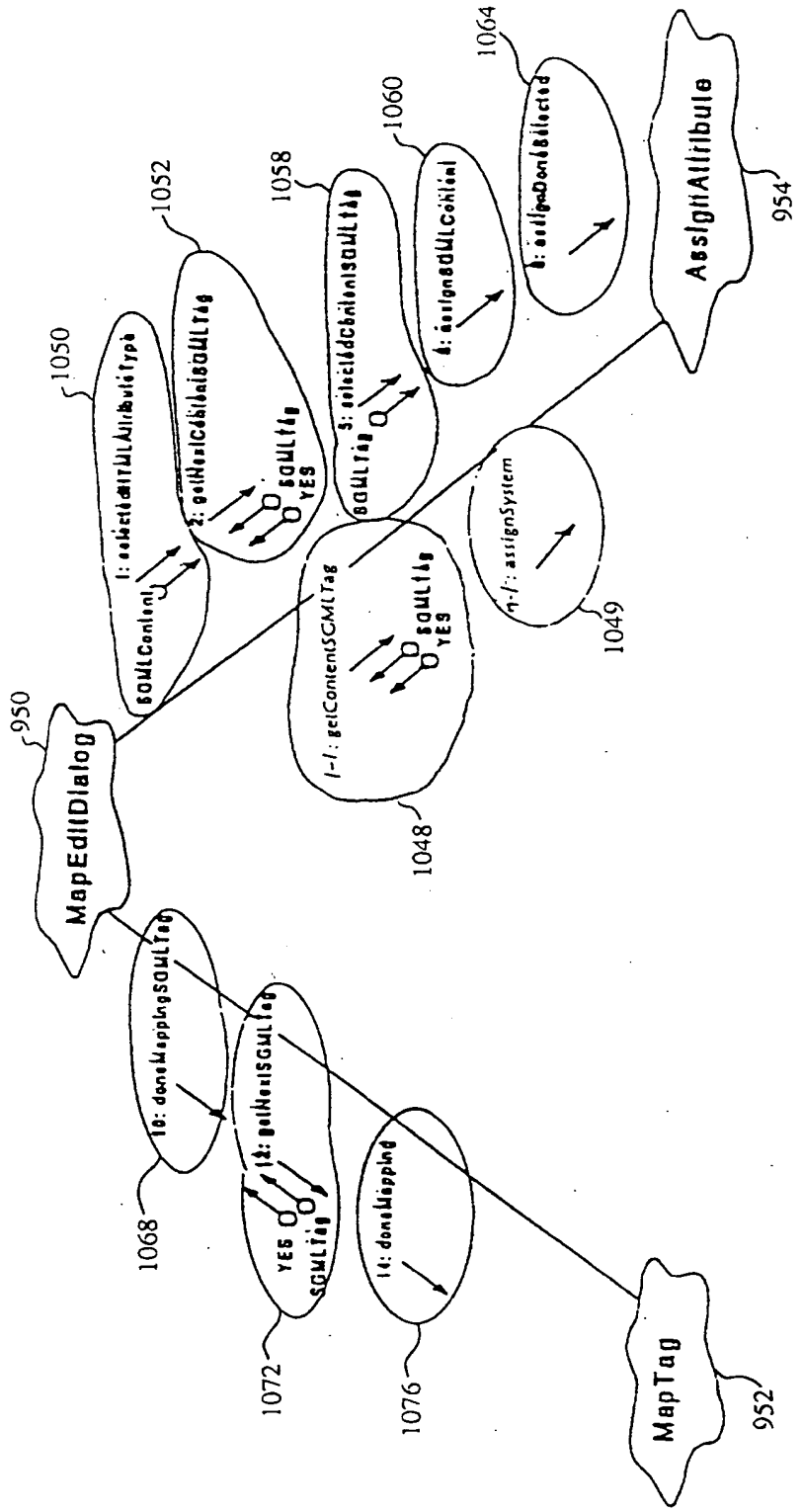

MapEditDialog
950
1050
1052
1058
1060
1064
1048
1049
1068
1072
1076
MapTag
952
AssignAttribute
954
YES
YES

Fig. 18B(1)

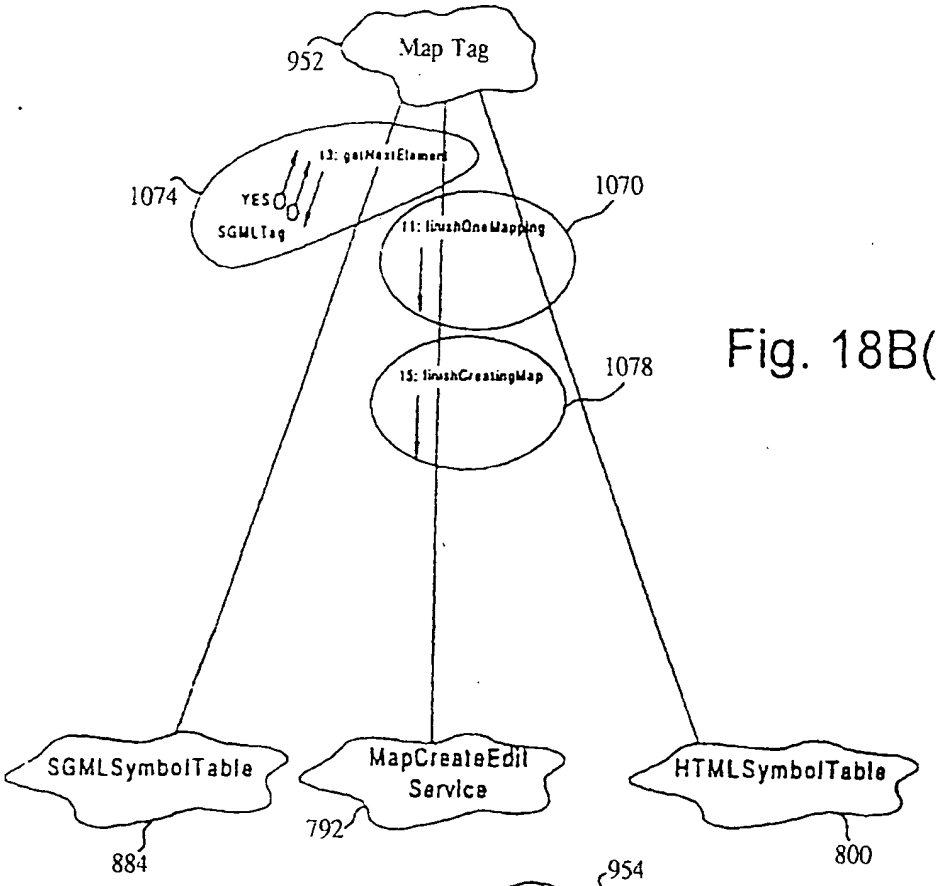


Fig. 18B(2)

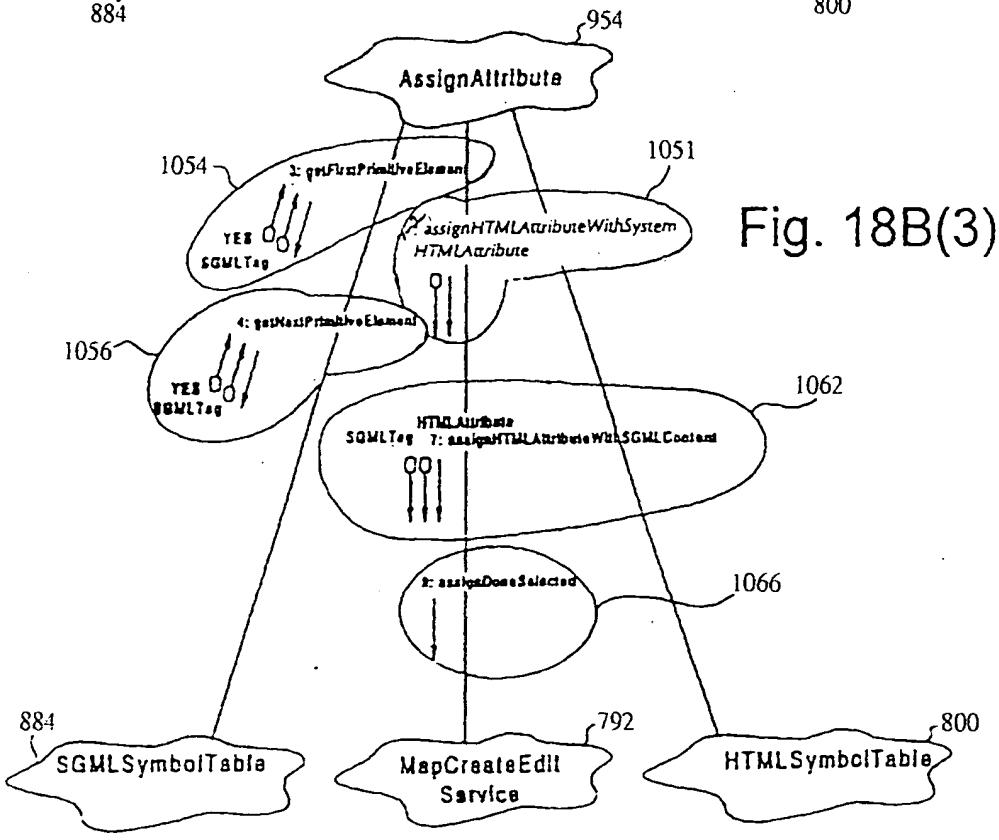


Fig. 18B(3)

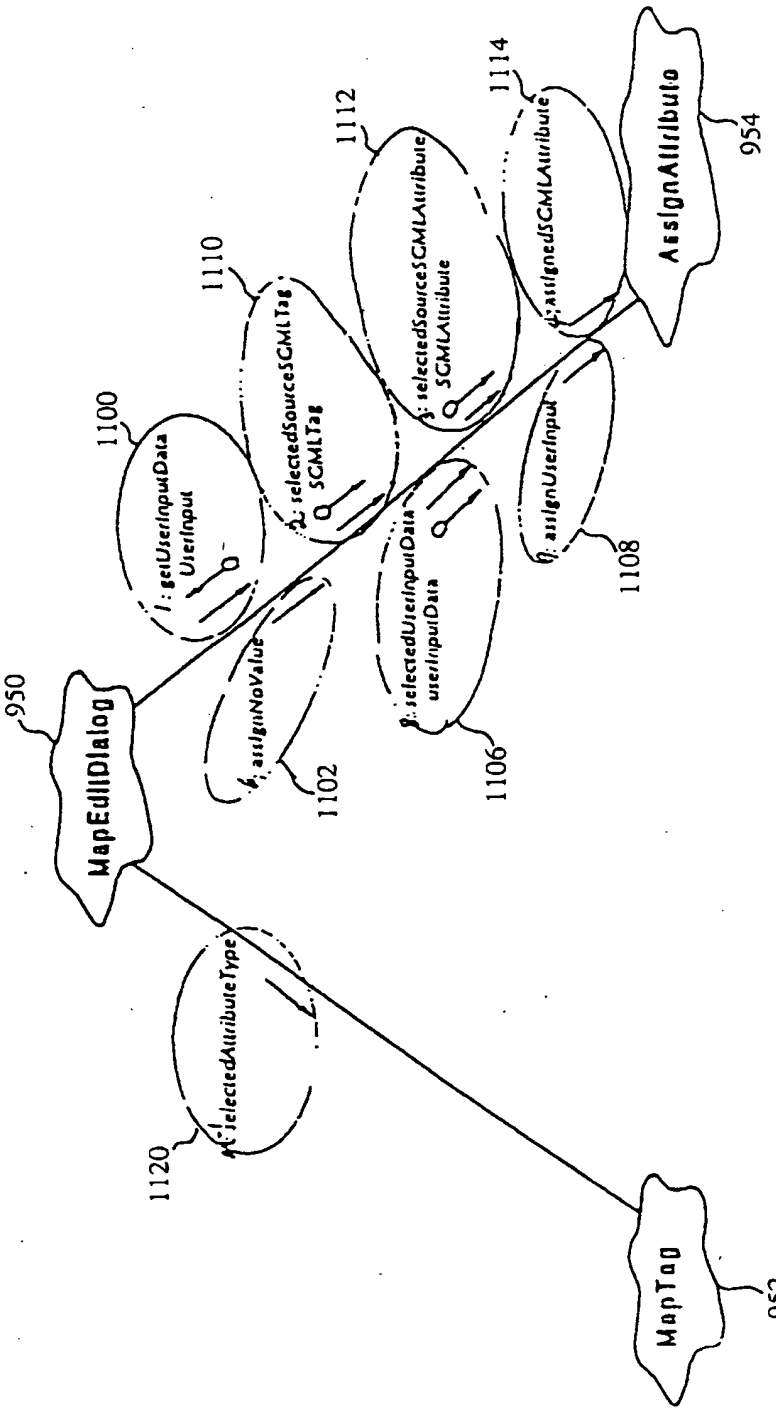


Fig. 18C(1)

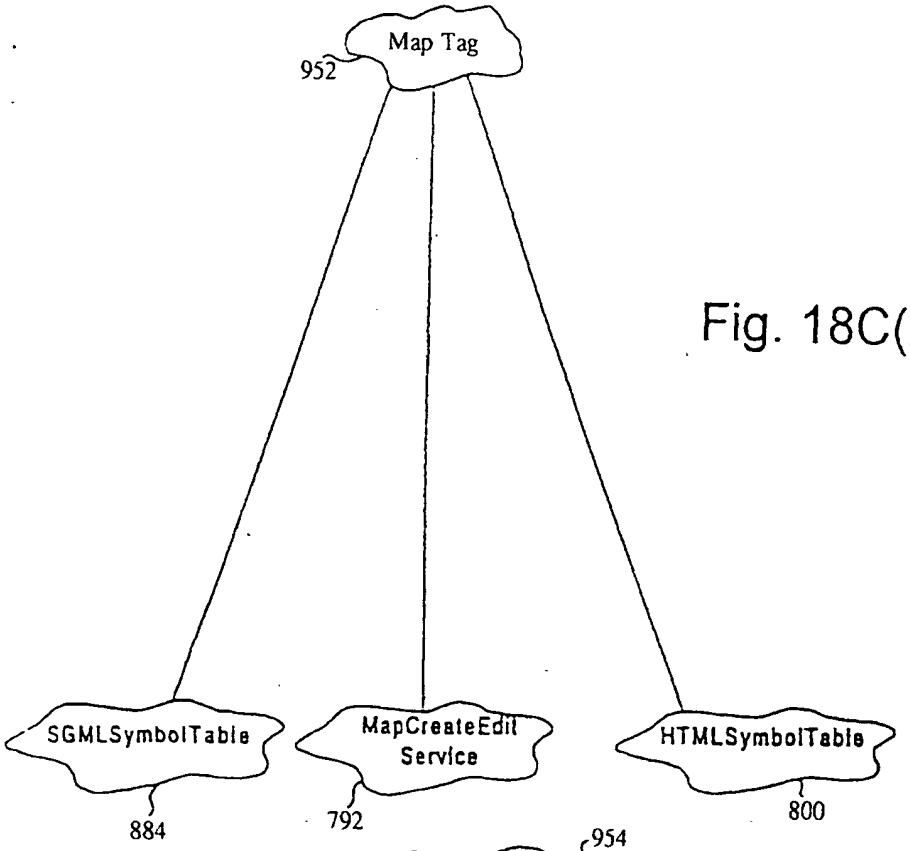


Fig. 18C(2)

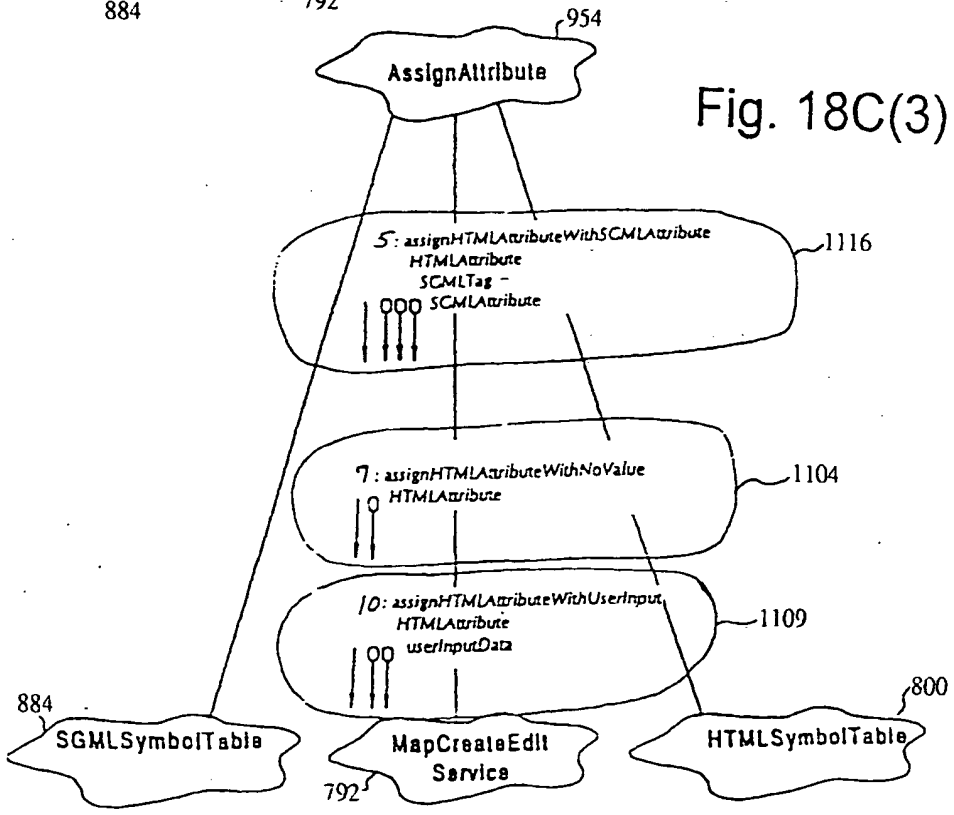


Fig. 18C(3)

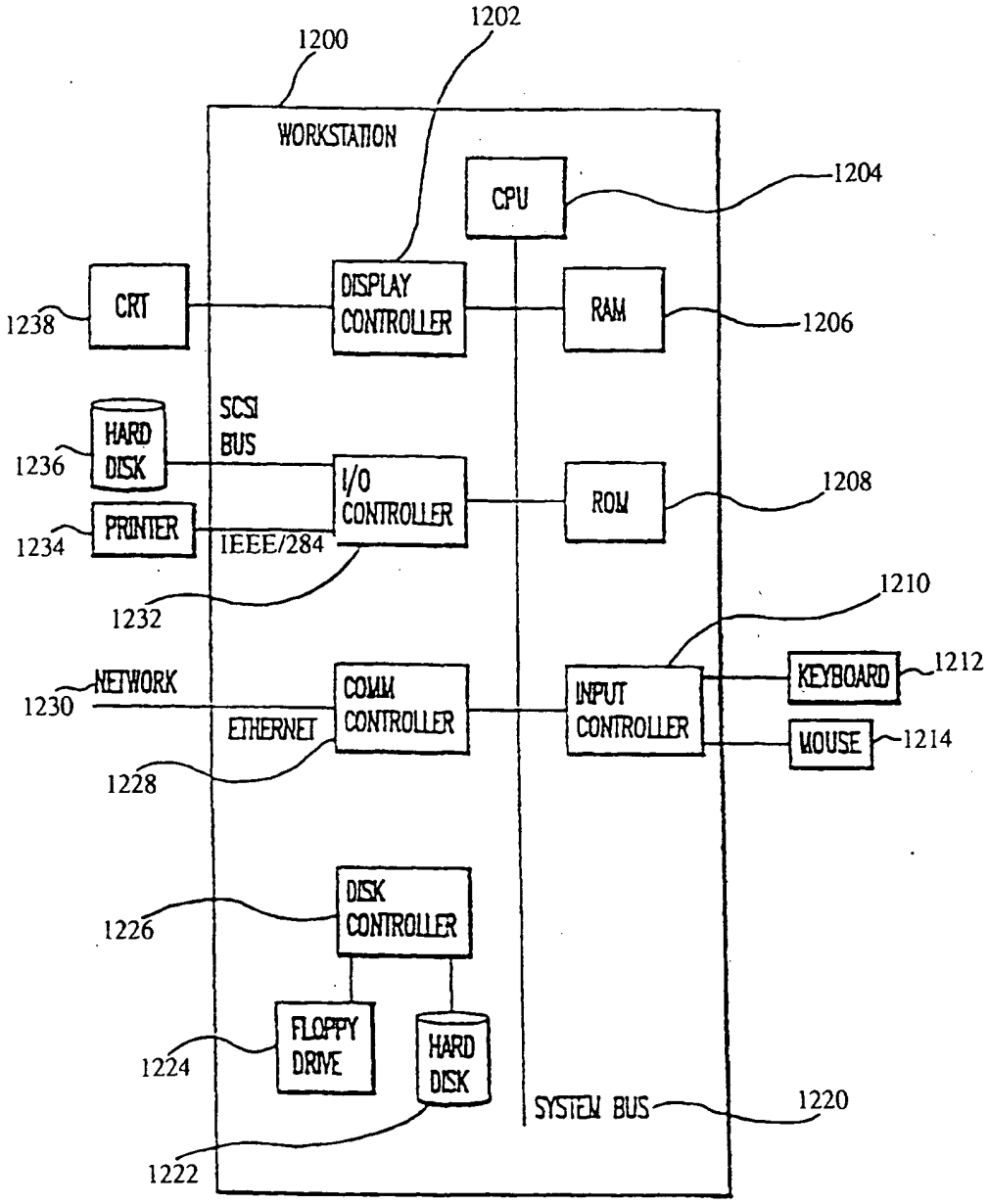


Fig. 19

1400 XYZ//font::metric::x-offset::622

Fig. 20A

1420	ownername	->	ownername
1422	ownerdescription	->	ownerdescription
1424	objectname	->	objectname
1426	objectdescription	->	objectdescription
1428	'::'	->	' '
1430	'//'	->	'_'

Fig. 20B

1440 XYZ__font_metric_x-offset_622

Fig. 20C

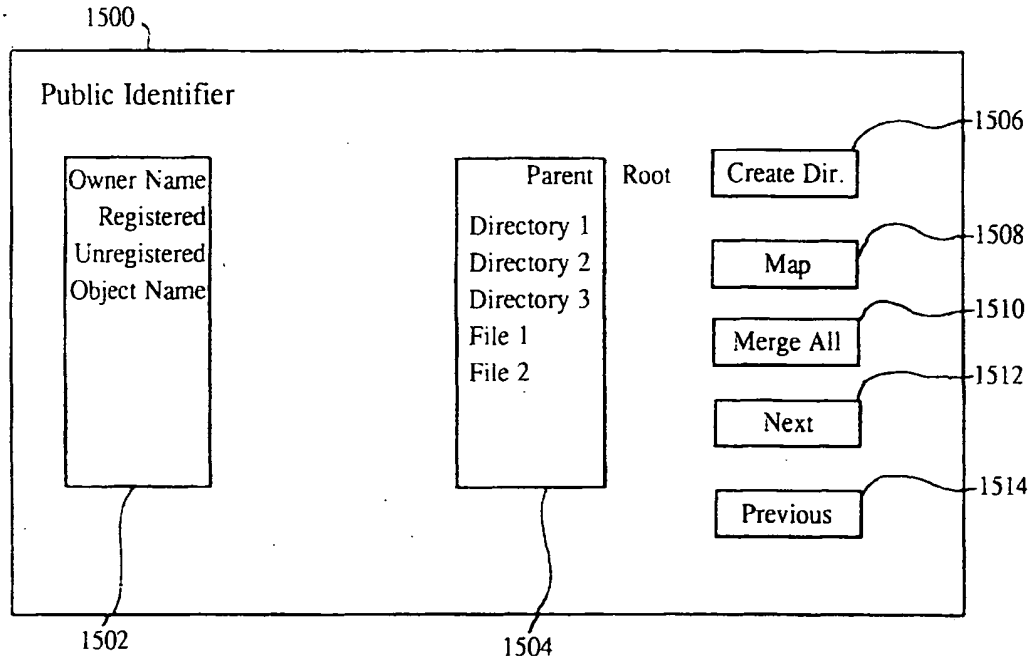


Fig. 20D

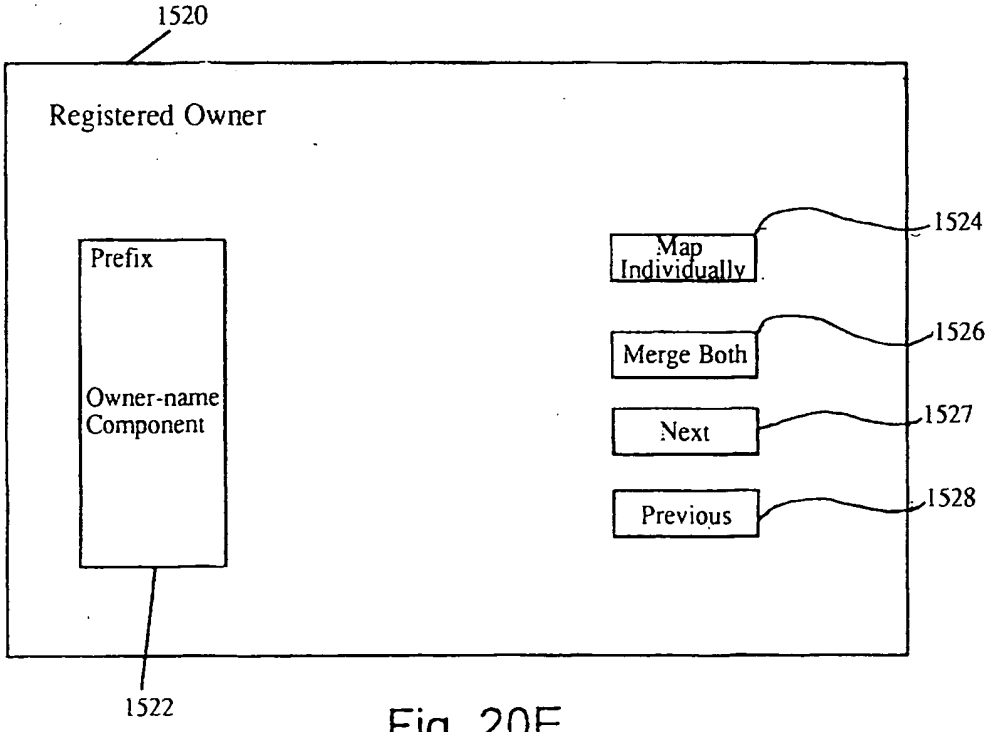


Fig. 20E

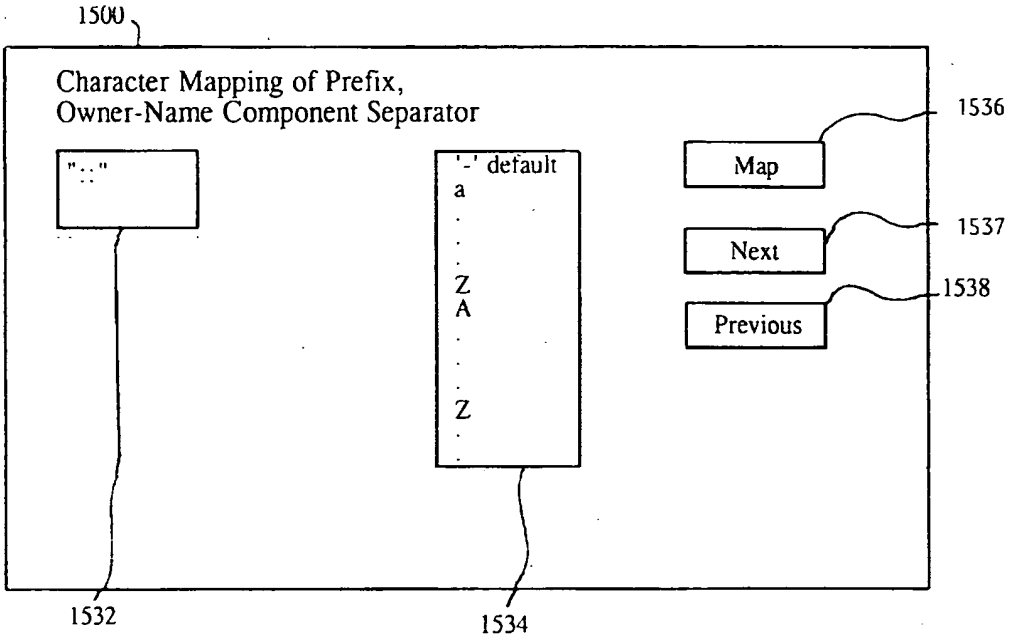


Fig. 20F

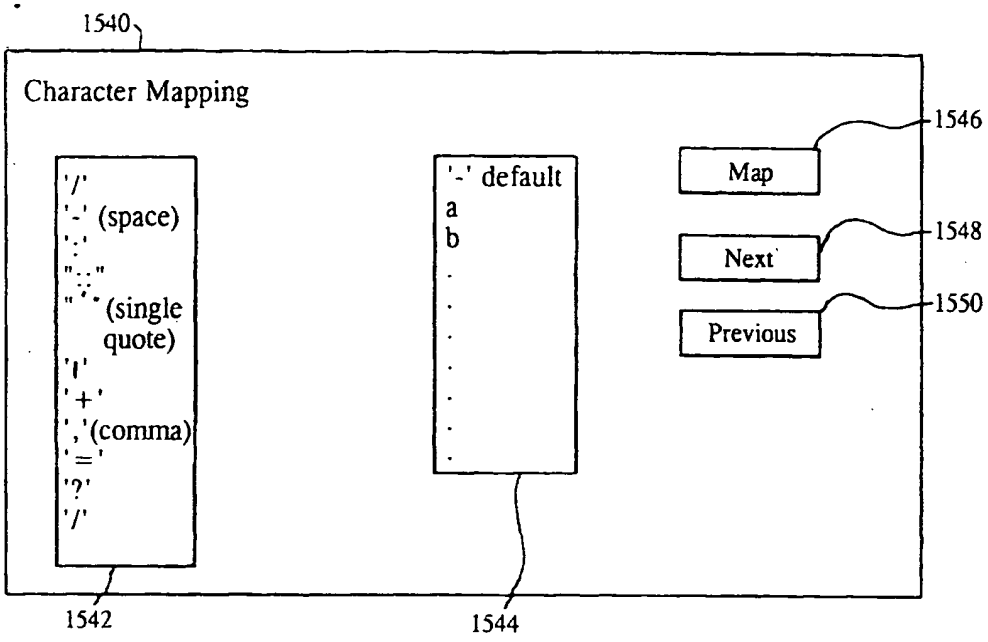


Fig. 20G

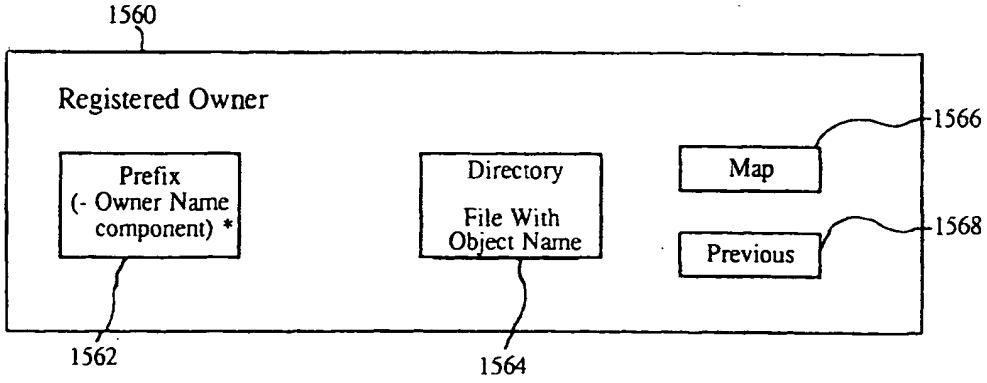


Fig. 20H

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

COPY MAILED

OCT 3 0 2006

OFFICE OF PETITIONS

GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK CA 94025

In re Application of
Barger, et al.
Application No.: 11/269,916
Filed: November 7, 2005
Attorney Docket No.: EQUI0001CIP-C

ON PETITION

This is a decision on the petition under 37 CFR 1.47(a), filed May 12, 2006.

The petition is **DISMISSED**.

Any request for reconsideration of this decision must be submitted within TWO (2) MONTHS from the mail date of this decision. Extensions of time under 37 CFR 1.136(a) are permitted. Any response should be entitled "Request for Reconsideration of Petition Under 37 CFR 1.47(a)" and may include an oath or declaration executed by the inventor. **Failure to respond will result in abandonment of the application.**

The above-identified application was filed on November 7, 2005. On December 12, 2005, the Office mailed a Notice to File Missing Parts of Nonprovisional Application, requiring submission of an executed declaration and a $65.00 late declaration surcharge. On May 12, 2006, petitioners filed, *inter alia*, a request for three month extension of time, a petition under 37 CFR 1.47(a) and a declaration signed by one of seven joint inventors. The petition states that a copy of the application and a declaration were mailed to each inventor for his signature, but that all mailings of the aforementioned documents were returned to sender. No forwarding addresses were provided.

A grantable petition under 37 CFR 1.47(a) requires:

(1) a petition including proof of the pertinent facts establishing that the joint inventor(s) refuses to join, or cannot be found or reached after diligent effort,
(2) a proper oath or Declaration executed by the available joint inventor(s),
(3) the petition fee of $200, and
(4) the last known address of the omitted inventor(s).

This petition lacks items (1) and (2) above.

As to item (1), applicant has failed to establish that the inventors cannot be reached.

A showing of **diligence** is critical in obtaining Rule 47 status when an inventor cannot be located or reached. One returned mailing does not rise to the level of diligence required to obtain Rule 47 status.

Petitioners should engage in further efforts to locate the non-signing inventors. The following is a list of evidentiary sources that are commonly relied upon to prove inability to locate an inventor: searches of Internet databases; inquiries of local telephone directories; telegrams; and documented inquiries of last known employers. Every listed type of search need not be done. However, a diligent effort to find the inventor must be made. Please provide documentary evidence of the searches, if possible, from parties with first hand knowledge of the searches.

As to item (2), an oath or declaration for the patent application in compliance with 37 CFR 1.63 and 1.64 still has not been presented. The declaration submitted with the present petition only lists three joint inventors. An oath or declaration in compliance with 37 CFR 1.63 and 1.64 signed by the Rule 1.47 applicant on behalf of the non-signing inventors is REQUIRED. See MPEP 409.03(a).

Pursuant to petitioner's authorization, deposit account no. 07-1445 will be charged a $200.00 Rule 47 petition fee.

Further correspondence with respect to this matter should be addressed as follows:

By mail: Mail Stop PETITION
Commissioner for Patents
Post Office Box 1450
Alexandria, VA 22313-1450

By hand: U.S. Patent and Trademark Office
Customer Service Window, Mail Stop Petition
Randolph Building
401 Dulany Street
Alexandria, VA 22314

By FAX: (571) 273-8300 - ATTN: Office of Petitions

Telephone inquiries should be directed to the undersigned at (571) 272-3230.

Shirene Willis Brantley

Shirene Willis Brantley
Senior Petitions Attorney
Office of Petitions

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of : Samaniego et al	Attorney Docket:	EQUI0001CIP-C
Serial No. 11/269,916	Art Unit:	2176
Filed: November 7, 2005	Examiner:	Unassigned

For: AUTOMATED MEDIA DELIVERY SYSTEM

12 May 2006

PETITION BASED UPON UNAVAILABILITY OF INVENTOR
(37 CFR § 1.47/MPEP 409.03)

Sir/Madam:

1. Applicant petitions pursuant to 37 CFR §1.47 that the examination of the above-identified patent application proceed in the absence of a Declaration signed by the inventors: Christopher Samaniego, Nelson H. Offner, Adian D. Thewlis, David R. Boyd, David C. Salmon and Joshua N. Devan.

2. Applicant provides herewith copies letters sent via Federal Express to above-mentioned inventors, at their last known addresses, which was returned from Federal Express, with no forwarding addresses. A copy of the subject patent application and oath and declaration accompanied each letter. Copies of the Federal Express return envelopes are enclosed.

3. Diligent efforts have been used to reach the inventors.

4. Applicant provides herewith a Statement from Inventor Sean Barger, Consent of Assignee, and a Declaration signed by Mr. Barger.

5. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title

10/30/2006 CKHLOK 00000013 071445 11269916

01 FC:1463 200.00 DA

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/269,916	11/07/2005	Sean Barger	EQUI0001CIP-C

CONFIRMATION NO. 5707

22862
GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK, CA94025

Title: Automated media delivery system

Publication No. US-2006-0265476-A1
Publication Date: 11/23/2006

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 703-305-3028.

Pre-Grant Publication Division, 703-605-4283

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of : Samaniego et al		Attorney Docket:	EQUI0001CIP-C
Serial No.	1/269,916	Art Unit:	2176
Filed:	November 7, 2005	Examiner:	Unassigned

Title: AUTOMATED MEDIA DELIVERY SYSTEM

January 30, 2007

REQUEST FOR RECONSIDERATION OF PETITION UNDER 37 CFR § 1.47(a)

Examiner:

1. Applicant petitions pursuant to 37 CFR §1.47(a) that the examination of the above-identified patent application proceed in the absence of a Declaration signed by the inventors: Christopher Samaniego, Nelson H. Offner, and Joshua N. Devan.

2. Applicant provides herewith copies of letters sent via Federal Express to above-mentioned inventors, at their last known addresses, which was returned from Federal Express, with no forwarding addresses. A copy of the subject patent application and Oath and Declaration accompanied each letter. Copies of the Federal Express return envelopes are enclosed for:

- Christopher Samaniego with **last known address** of 461 Second Street, San Francisco, California 94107
- Nelson H. Offner with **last known address** of 172 Ardmore Road, Kensington, California 94707
- Joshua N. Devan with **last known address** of 25 Stetson Avenue, Lower, Kentfield, California 94904

3. Applicant also provides herewith the relevant portion of an email from CEO, Sean Barger indicating that the company was unable to get signatures from the inventors: Christopher Samaniego, Nelson H. Offner, and Joshua N. Devan. Such provides documentary evidence of a search from a party with first hand knowledge of the search.

4. The free Internet Search from 411.com Online Directory Assistance which indicates no matching results for the missing inventors. Such is documentary evidence of further efforts to locate the non-signing inventors.

5. Further diligent efforts were made to contact three inventors whose signature appears on the newly executed Oath-Declaration.

6. Applicant provides herewith the following documents: Copies of the Statement from Inventor Sean Barger, Consent of Assignee, and a Declaration signed by Mr. Barger.

7. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this statement is directed.

8. This action is necessary to prevent irreparable damage and to preserve the rights of the parties.

The Commissioner is authorized to charge any fees forth in 37 CFR 1.17 (i), and any additional fees that may be due and credit any overpayments to Deposit Account No. 07-1445 (Order No. EQUI0001CIP-C).

Respectfully submitted,

Julia A. Thomas

Julia A. Thomas
Reg. No. 52,283

Customer No. 22862

Attorney Docket No. EQUI0001CIP-C

DECLARATION FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address, and citizenship are as stated below next to my name;

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

Automated Media Delivery System

the specification of which (check one) ____ is attached hereto, or _X_ was filed on 11/07/2005 as Application Serial No. 11/269,916 and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

I hereby claim foreign priority benefits under Title 35, United Sates Code, Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
			Yes	No
Number	Country	Day/Month/Year Filed		
			___	___
			___	___

Number Country Day/Month/Year Filed

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

MICHAEL A. GLENN, Reg. No. 30,176
DONALD M. HENDRICKS, Reg. No. 40,355
JULIA A. THOMAS, Reg. No. 52,283
CHRISTOPHER PEIL, Reg. No. 45,005
JEFFREY BRILL, Reg. No. 51,198

SEND CORRESPONDENCE TO:

MICHAEL A. GLENN, 3475 Edison Way, Suite L, Menlo Park, CA 94025

I hereby claim the benefit under Title 35, United States code, Section 119(3)120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

6,964,009	11/08/2005	Patented
Patent Number	Grant Date	Status: Patented, Pending, Abandoned
6,792,575	9/14/2004	Patented
Patent Number	Grant Date	Status: Patented, Pending, Abandoned

Attorney Docket No. EQUI0001CIP-C

Application Ser. No. Filing Date Status: Patented, Pending, Abandoned

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or **first** inventor: SEAN BARGER

Inventor's signature _____ Date 1/19/07

Residence 222 Marguerite Ave., Mill Valley, CA 94941

Post Office Address Same

Citizenship United States of America

Full name of **second** or joint inventor: CHRISTOPHER SAMANIEGO

Inventor's signature _____ Date

Residence 461 Second Street, San Francisco, California 94107

Post Office Address Same

Citizenship United States of America

Full name of **third** or joint inventor: NELSON H. "ROCKY" OFFNER

Inventor's signature _____ Date

Residence 172 Ardmore Road, Kensington, California 94707

Post Office Address Same

Citizenship United States of America

Full name of **fourth** or joint inventor: ADRIAN D. THEWLIS

Inventor's signature Adrian Thewlis. 1/19/07.

Residence 24 ADRIAN TERRACE, SAN RAFAEL, CA 94903 Date
~~439 Sherwood Drive, Apt. 204, Sausalito, California 94965~~

Post Office Address Same

Citizenship ~~Australia~~ UNITED STATES OF AMERICA

Full name of **fifth** or joint inventor: DAVID R. BOYD

Inventor's signature 1/4/2007
Date

Residence 1256 Kearny Street, San Francisco, California 94133

Post Office Address Same

Citizenship United States of America

Full name of **sixth** or joint inventor: DAVID C. SALMON

Inventor's signature 1/3/2007
Date

Residence 42 Sandalwood Court, San Rafael, California 94903

Post Office Address Same

Citizenship United States of America

Full name of **seventh** or joint inventor: JOSHUA N. DEVAN

Inventor's signature _____ _____
Date

Residence 25 Stetson Avenue, Lower, Kentfield, California 94904

Post Office Address Same

Citizenship United States of America

Della Revecho

COPY

From: Sean Barger [sbarger@equilibrium.com]
Sent: Thursday, April 06, 2006 9:59 AM
To: Della
Subject: Re: [EQUI0001CIP-C]

Della,

We have attempted to get signatures from the following people:
Chris Samaniego, Rocky Offner, Adrian Thewlis (think we may be still able to find him), Joshua Devan to no avail.

Dave Salmon and David Boyd should be coming soon. However, we already have an assignment from all these people...do we really need to spend our time on this for something that is already assigned?

Please let us know.

sbb

Sean Barger
CEO
Equilibrium
3 Harbor Drive, Suite 100
Sausalito, CA 94965

+1 415.332.4343 x201
+1 415.465.5556 mobile
+1 415.331.8374 fax
http://www.equilibrium.com
mailto:sbarger@equilibrium.com
The leader in Automated Media Processing Solutions.

ショーン B. バーガー
取締役会長および最高経営責任者
sbarger@equilibrium.com
直通 +1.415.332.4343 内線 201
携帯 +1.415.465.5556

〒94965 米国カリフォルニア州
サウサリート市
ハーバードライブ 3 番地 100 号室
直通 +1.415.332.4343
ファックス +1.415.331.8374
www.equilibrium.com

イクイリブリアム

From: Della <della@glenn-law.com>
Organization: Glenn Patent Group
Date: Tue, 4 Apr 2006 13:22:40 -0700
To: EQUILIBRIUM-Sean Barger <sbarger@equilibrium.com>
Cc: EQUILIBRIUM-Steve Denebeim <sdenebeim@equilibrium.com>
Subject: FW: [EQUI0001CIP-C]

Sean,

We have not received further instructions concerning the Declaration.

For your convenience, I've attached another copy of the Declaration-Oath, for the inventors to sign.

Please note that the 3rd and final extension is May 12, 2006.

We look forward to hearing from you.

Regards,
Della

Glenn Patent Group
3475 Edison Way, Suite L
Menlo Park, CA 94025
Tel 650 474 8400
Fax 650 474 8401
glenn-law.com

COPY

G|P|G

13 April 2006
Via Federal Express

CHRISTOPHER SAMANIEGO
461 Second Street
San Francisco, CA 94107

Re: Application Entitled Automated Media Delivery System
Application No. 11/269,916
Our File: EQUI0001CIP-C

Dear Christopher:

Enclosed please find a copy of the above-referenced application (specification, claims and figures) filed on 07 November 2005, naming you a joint inventor. Additionally enclosed is the Declaration and Oath, which requires your signature.

Please sign and date where indicated, and **fax** (to 650-474-8401) or mail the documents to our office by **20 April 2006.** If mailing, a self-addressed, postage paid FedEx envelope is provided herewith.

If we do not receive the signed documents by this date, we will assume that you refused to cooperate and sign the declaration for the above-referenced application.

We look forward to hearing from you soon. For any questions or need further information, please contact our office to speak with Michael Glenn, Equilibrium's attorney.

We look forward to receiving the signed Declaration.

Regards,

Della Revecho
Patent Administrator

/dcr
Enclosures

COPY

CHRISTOPHER SAMANIEGO

ipments Only

ORIGIN ID: NDBA (800) 463-3339
SFO RETURNS
FEDEX SFOA STATION
1875 MARIN ST.
SAN FRANCISCO, CA 94124
UNITED STATES US

Ship Date: 17APR06
Actual Wgt: 2.0 LB MAN
System#: 0021632/CAFE2285
Account: S *********

TO RETURNS
GLENN PATENT GROUP
3475 EDISON WAY STE L
MENLO PARK, CA 94025

FedEx Express

CLS0919805/08/15

REF: 791445428838

Delivery Address Barcode

BILL RECIPIENT

EXPRESS SAVER PACKAGE

TRK# 7245 2763 4240

Form 0201

THU
Deliver By:
20APR06

SFO A2

94025 -CA-US SC HGTA

Part # 156148-434 NRIT 10-05

FedEx

From: Origin ID:

Gle atent Group
3475 Edison Way, Suite L
Menlo Park, CA 94025
Fax 650 474 8401
glenn-law.com

COPY

G|P|G

13 April 2006
Via Federal Express

NELSON OFFNER
172 Ardmore Road
Kensington, CA 94707

Re: Application Entitled Automated Media Delivery System
Application No. 11/269,916
Our File: EQUI0001CIP-C

Dear Nelson:

Enclosed please find a copy of the above-referenced application (specification, claims and figures) filed on 07 November 2005, naming you a joint inventor. Additionally enclosed is the Declaration and Oath, which requires your signature.

Please sign and date where indicated, and **fax** (to 650-474-8401) or mail the documents to our office by **20 April 2006.** If mailing, a self-addressed, postage paid FedEx envelope is provided herewith.

If we do not receive the signed documents by this date, we will assume that you refused to cooperate and sign the declaration for the above-referenced application.

We look forward to hearing from you soon. For any questions or need further information, please contact our office to speak with Michael Glenn, Equilibrium's attorney.

We look forward to receiving the signed Declaration.

Regards,

Della Revecho
Patent Administrator

/dcr
Enclosures

NELSON OFFNER

COPY

FedEx

SHIP DATE: 25APR06 1/1
SYSTEM #662404 / CAFE2285
ACCOUNT #: 136853406
ACTUAL WGT: 1.0 LBS MAN-WGT

ORIGIN ID: JEMA
FEDEX
FDX/JEMA STATION
1600 63RD ST
EMERYVILLE, CA 94608

TO: RETURNS
GLENN PATENT GROUP
3475 EDISON WAY STE L
STE L
MENLO PARK, CA 94025

BILL RECIPIENT

6614 5480 7922

6614 5480 7922

REF: 7914-454us-0840

FRI
Deliver by
28APR06
A2

EXPRESS SAVER

TRK# 6614 5480 7922 FORM 0201

SFO

FRI

FedEx EXPRESS SAVER

FDX PAK Deliver By:
28APR06
A2

SHP# 323121 26APR06 14:02

FedEx FORM 0201

Trk# 6614 5480 7922

SFO

94025 -CA-US

SCHGTA

153077 RIK 11/01

FedEx Express Urgent

136596 9/00 MWI

Name no phone # to call

Company recip RTS 4/24/06

Address

City, State, Zip ACT 16905 BB 98

Telephone

The IA

C Patent Group
3 Edison Way, Suite L
Menlo Park, CA 94025
Tel 650 474 8400
Fax 650 474 8401
glenn-law.com

G|P|G

13 April 2006
Via Federal Express

JOSHUA N. DEVAN
25 Stetson Avenue, Lower
Kentfield, CA 94904

Re: Application Entitled Automated Media Delivery System
Application No. 11/269,916
Our File: EQUI0001CIP-C

Dear Joshua:

Enclosed please find a copy of the above-referenced application (specification, claims and figures) filed on 07 November 2005, naming you a joint inventor. Additionally enclosed is the Declaration and Oath, which requires your signature.

Please sign and date where indicated, and **fax** (to 650-474-8401) or mail the documents to our office by **20 April 2006.** If mailing, a self-addressed, postage paid FedEx envelope is provided herewith.

If we do not receive the signed documents by this date, we will assume that you refused to cooperate and sign the declaration for the above-referenced application.

We look forward to hearing from you soon. For any questions or need further information, please contact our office to speak with Michael Glenn, Equilibrium's attorney.

We look forward to receiving the signed Declaration.

Regards,

Della Revecho
Patent Administrator

/dcr
Enclosures

COPY

From: Origin ID: (650)474-8400
GLENN PATENT GROUP

3475 Edison Way, Ste. L

Menlo Park, CA 94025

FedEx Express

CLS#22366/16/19

Ship Date: 13APR06
ActWgt: 1 LB
System#: 2949728/INET2400
Account#: S *********

REF: EQUI0001CIP-C

Delivery Address Bar Code

SHIP TO: (000)000-0000 BILL SENDER

Joshua N. Devan

25 Stetson Avenue, Lower

Kentfield, CA 94904

PRIORITY OVERNIGHT

FRI
Deliver By:
14APR06

TRK# 7914 4550 6116 FORM 0201

OAK A2

94904 -CA-US

WA SRFA

Shipping Label

This shipping label constitutes the air waybill for this shipment.

View all labels

1. Use the "Print" feature from your browser to send this page to your laser printer.
2. Fold the printed page along the horizontal line.
3. Place label in shipping label pouch and affix it to your shipment so that the barcode portion of the label can be read and scanned.

Warning: Use only the printed original label for shipping. Using a photocopy of this label for shipping purposes is fraudulent and could result in additional billing charges, along with the cancellation of your FedEx account number.

Use of this system constitutes your agreement to the service conditions in the current FedEx Service Guide, available on fedex.com. FedEx will not be responsible for any claim in excess of $100 per package, whether the result of loss, damage, delay, non-delivery, misdelivery, or misinformation, unless you declare a higher value, pay an additional charge, document your actual loss and file a timely claim. Limitations found in the current FedEx Service Guide apply. Your right to recover from FedEx for any loss, including intrinsic value of the package, loss of sales, income interest, profit,attorney's fees, costs, and other forms of damage whether direct, incidental, consequential, or special is limited to the greater of $100 or the authorized declared value. Recovery cannot exceed actual documented loss. Maximum for items of extraordinary value is $500, e.g. jewelry, precious metals, negotiable instruments and other items listed in our Service Guide. Written claims must be filed within strict time limits, see current FedEx Service Guide.

411.com

ACCOUNT | SERVICES | LOGIN REGISTER

WHITE PAGES | YELLOW PAGES | EMAIL SEARCH | REVERSE PHONE | REVERSE ADDRESS | AREA & ZIP CODES | MAILING LISTS

FIND NEIGHBORS | INTERNATIONAL DIRECTORIES | BATCH PEOPLE SEARCHES | AUTOMATE PEOPLE LOOKUPS

NEW SEARCH

0 Results matching "Christopher Samaniego, 461 Second Street, San Francisco, CA"

Search Suggestions

Search for the **last name** Samaniego in San Francisco, CA

Search for Christopher Samaniego in the San Francisco, CA **metro area**

Search for the first name **beginning with** C, last name with Sam in San Francisco, CA

Revise Your Search

Christopher Samaniego is in our Database

Powered by 1.800.US.SEARCH

First Name: Christopher Last Name: Samaniego

City: san francisco State: CA

Search

Search by Social Security Number, Maiden Name, Age and More!

Search

Home
People Search
Business Search
Business Search
Email Search
Reverse Phone
Reverse Address
Area Codes
ZIP Codes
Tools & Resources
Site Map

Tools

About WhitePages
Services
Phone Append
Address Append
XML Lookups
People Search by State
Business Search by State

Resources

Wedding Center
Auto Center
Moving and Home
Improvement Center

About Our Searches

White Pages
Business Search
Reverse Phone Directory
Reverse Address Lookup
Email Address Search
Area Code and ZIP Code Directory

WhitePages.com, Inc.

FAQ
About Us
Advertising
Affiliate Program
Contact Us
Tell friends about 411.com

411.com

ACCOUNT | SERVICES | LOGIN / REGISTER

WHITE PAGES | YELLOW PAGES | EMAIL SEARCH | REVERSE PHONE | REVERSE ADDRESS | AREA & ZIP CODES | MAILING LISTS

FIND NEIGHBORS | INTERNATIONAL DIRECTORIES | BATCH PEOPLE SEARCHES | AUTOMATE PEOPLE LOOKUPS

< NEW SEARCH

0 Results matching "Nelson Offner, 172 Ardmore Road, Kensington, CA"

Search Suggestions

Search for the **last name** Offner in Kensington, CA

Search for Nelson Offner in the Kensington, CA **metro area**

Search for the first name **beginning with** N, last name with Off in Kensington, CA

Revise Your Search

Nelson Offner is in our Database

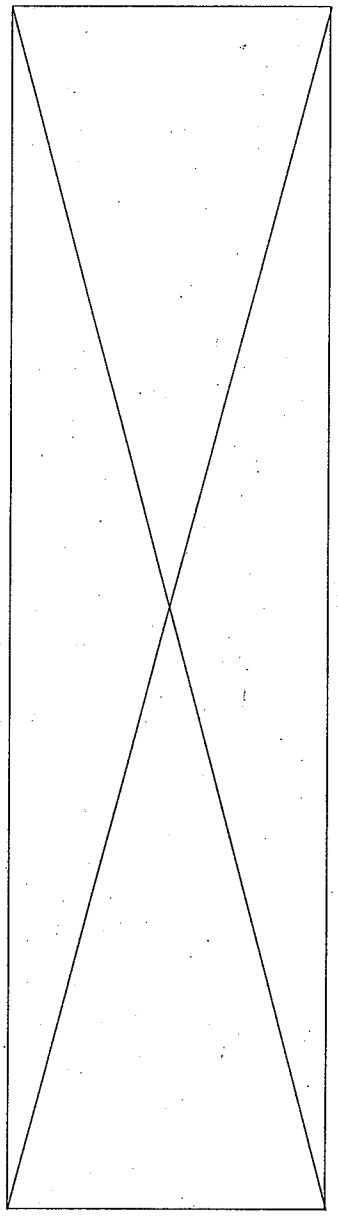
Powered by 1.800.US.SEARCH

First Name: Nelson Last Name: Offner

City: kensington State: CA

Search

Search by Social Security Number, Maiden Name, Age and More!



Search

Home
People Search
Business Search
Business Search
Email Search
Reverse Phone
Reverse Address
Area Codes
ZIP Codes
Tools & Resources
Site Map

Tools

About WhitePages
Services
Phone Append
Address Append
XML Lookups
People Search by State
Business Search by State

Resources

Wedding Center
Auto Center
Moving and Home
Improvement Center

About Our Searches

White Pages
Business Search
Reverse Phone Directory
Reverse Address Lookup
Email Address Search
Area Code and ZIP Code Directory

WhitePages.com, Inc.

FAQ
About Us
Advertising
Affiliate Program
Contact Us
Tell friends about 411.com

411.com

ACCOUNT | SERVICES | LOGIN / REGISTER

WHITE PAGES | YELLOW PAGES | EMAIL SEARCH | REVERSE PHONE | REVERSE ADDRESS | AREA & ZIP CODES | MAILING LISTS

FIND NEIGHBORS | INTERNATIONAL DIRECTORIES | BATCH PEOPLE SEARCHES | AUTOMATE PEOPLE LOOKUPS

< NEW SEARCH

0 Results matching "Joshua Devan, 25 Stetson Av, San Rafael, CA"

Search Suggestions

Search for the **last name** Devan in San Rafael, CA

Search for Joshua Devan in the San Rafael, CA **metro area**

Search for the first name **beginning with** J, last name with Dev in San Rafael, CA

Revise Your Search

Joshua Devan is in our Database
Powered by 1.800.US.SEARCH

First Name: Joshua Last Name: Devan

City: san rafael State: CA

Search

Search by Social Security Number, Maiden Name, Age and More!

Search

Home
People Search
Business Search
Business Search
Email Search
Reverse Phone
Reverse Address
Area Codes
ZIP Codes
Tools & Resources
Site Map

Tools

About WhitePages
Services
Phone Append
Address Append
XML Lookups
People Search by State
Business Search by State

Resources

Wedding Center
Auto Center
Moving and Home
Improvement Center

About Our Searches

White Pages
Business Search
Reverse Phone Directory
Reverse Address Lookup
Email Address Search
Area Code and ZIP Code Directory

WhitePages.com, Inc.

FAQ
About Us
Advertising
Affiliate Program
Contact Us
Tell friends about 411.com

PTO/SB/22 (09-06)
Approved for use through 03/31/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARMENT OF COMMERCE

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) **FY 2006** ***(Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).)***	Docket Number (Optional) EQUI0001CIP-C
Application Number 11/269,916	Filed 11/07/2005
For Automated Media Delivery System	
Art Unit 2176	Examiner Unassigned

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

		Fee	Small Entity Fee	
[X]	One month (37 CFR 1.17(a)(1))	$120	$60	$ 60.00
[ ]	Two months (37 CFR 1.17(a)(2))	$450	$225	$
[ ]	Three months (37 CFR 1.17(a)(3))	$1020	$510	$
[ ]	Four months (37 CFR 1.17(a)(4))	$1590	$795	$
[ ]	Five months (37 CFR 1.17(a)(5))	$2160	$1080	$

[ ] Applicant claims small entity status. See 37 CFR 1.27.

[ ] A check in the amount of the fee is enclosed.

[ ] Payment by credit card. Form PTO-2038 is attached.

[ ] The Director has already been authorized to charge fees in this application to a Deposit Account.

[X] The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 07-1445 (Glenn Patent Group). I have enclosed a duplicate copy of this sheet.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

I am the

[ ] applicant/inventor.

[ ] assignee of record of the entire interest. See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed (Form PTO/SB/96).

[X] attorney or agent of record. Registration Number 52,283

[ ] attorney or agent under 37 CFR 1.34. Registration number if acting under 37 CFR 1.34 _____

Julia A. Thomas
Signature

01/30/2007
Date

Julia A. Thomas
Typed or printed name

650-474-8400
Telephone Number

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

[X] Total of 1 forms are submitted.

COPY

PTO/SB/22 (09-06)
Approved for use through 03/31/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARMENT OF COMMERCE
Under the paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless if displays a valid OMB control number.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) **FY 2006** *(Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).)*	Docket Number (Optional) EQUI0001CIP-C
Application Number 11/269,916	Filed 11/07/2005
For Automated Media Delivery System	
Art Unit 2176	Examiner Unassigned

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

		Fee	Small Entity Fee	
[X]	One month (37 CFR 1.17(a)(1))	$120	$60	$ 60.00
[ ]	Two months (37 CFR 1.17(a)(2))	$450	$225	$
[ ]	Three months (37 CFR 1.17(a)(3))	$1020	$510	$
[ ]	Four months (37 CFR 1.17(a)(4))	$1590	$795	$
[ ]	Five months (37 CFR 1.17(a)(5))	$2160	$1080	$

[ ] Applicant claims small entity status. See 37 CFR 1.27.

[ ] A check in the amount of the fee is enclosed.

[ ] Payment by credit card. Form PTO-2038 is attached.

[ ] The Director has already been authorized to charge fees in this application to a Deposit Account.

[X] The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 07-1445 (Glenn Patent Group). I have enclosed a duplicate copy of this sheet.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

I am the

[ ] applicant/inventor.

[ ] assignee of record of the entire interest. See 37 CFR 3.71.
Statement under 37 CFR 3.73(b) is enclosed (Form PTO/SB/96).

[X] attorney or agent of record. Registration Number 52,283

[ ] attorney or agent under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34 _____

Julia A. Thomas
Signature

01/30/2007
Date

Julia A. Thomas
Typed or printed name

650-474-8400
Telephone Number

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

[X] Total of 1 forms are submitted.

This collection of information is required by 37 CFR 1.136(a). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 6 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Electronic Patent Application Fee Transmittal

Application Number:	11269916
Filing Date:	07-Nov-2005
Title of Invention:	Automated media delivery system
First Named Inventor/Applicant Name:	Sean Barger
Filer:	Michael Glenn/Della Revecho
Attorney Docket Number:	EQUI0001CIP-C

Filed as Small Entity

Utility Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 1 month with $0 paid	2251	1	60	60

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**60**

Electronic Acknowledgement Receipt

EFS ID:	1478368
Application Number:	11269916
International Application Number:	
Confirmation Number:	5707
Title of Invention:	Automated media delivery system
First Named Inventor/Applicant Name:	Sean Barger
Customer Number:	22862
Filer:	Michael Glenn/Della Revecho
Filer Authorized By:	Michael Glenn
Attorney Docket Number:	EQUI0001CIP-C
Receipt Date:	30-JAN-2007
Filing Date:	07-NOV-2005
Time Stamp:	20:35:00
Application Type:	Utility

Payment information:

Submitted with Payment	yes
Payment was successfully received in RAM	$60
RAM confirmation Number	1157
Deposit Account	071445
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows: Charge any Additional Fees required under 37 C.F.R. Section 1.16 and 1.17	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part /.zip	Pages (if appl.)
1		PetnUncoopEFIled013007.pdf	1262012	yes	22

Multipart Description/PDF files in .zip description		
Document Description	**Start**	**End**
Miscellaneous Incoming Letter	1	1
Petition for review by the Office of Petitions.	2	3
Oath or Declaration filed	4	7
Miscellaneous Incoming Letter	8	20
Extension of Time	21	22

Warnings:

Information:

2	Fee Worksheet (PTO-06)	fee-info.pdf	8136	no	2

Warnings:

Information:

Total Files Size (in bytes):	1270148

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

ELECTRONIC TRANSMITTAL COVER SHEET

Application Serial No.: 11/269,916

Attorney Docket No. EQUI0001CIP-C

I hereby certify that this correspondence is being ELECTRONICALLY TRANSMITTED to the United States Patent and Trademark Office

From: GLENN PATENT GROUP
Customer No.: 22,862
Tel: (650) 474-8400
Fax: (650) 474-8401

on January 30, 2007
Date

Signature

Della Revecho

Typed or printed name of person signing Certificate

Note: Each paper must have its own certificate of transmission, or this certificate must identify each submitted paper.

Attached to this cover sheet please find the following documents:

- Electronic Transmittal Cover Sheet (1 page);
- Request for Reconsideration of Petition (2 pages);
- Oath and Declaration (4 pages);
- Copies of: E-mail dated 4/06/06, Returned Letters and Fed-Ex Receipts, and 411.com Search (13 pages); and
- Petition for Extension of Time (1 page in duplicate)

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK CA 94025

COPY MAILED

MAR 0 8 2007

OFFICE OF PETITIONS

In re Application of :
Barger, et al. : ON PETITION
Application No.: 11/269,916 :
Filed: November 7, 2005 :
Attorney Docket No.: EQUI0001CIP-C :

This is a decision on the reconsideration petition under 37 CFR 1.47(a), filed January 30, 2007.

The petition is **DISMISSED**.

Any request for reconsideration of this decision must be submitted within TWO (2) MONTHS from the mail date of this decision. Extensions of time under 37 CFR 1.136(a) are permitted. Any response should be entitled "Request for Reconsideration of Petition Under 37 CFR 1.47(a)" and may include an oath or declaration executed by the inventor. **Failure to respond will result in abandonment of the application.**

The above-identified application was filed on November 7, 2005. On December 12, 2005, the Office mailed a Notice to File Missing Parts of Nonprovisional Application, requiring submission of an executed declaration and a $65.00 late declaration surcharge. On May 12, 2006, petitioners filed, *inter alia*, a request for three month extension of time, a petition under 37 CFR 1.47(a) and a declaration signed by one of seven joint inventors. The petition states that a copy of the application and a declaration were mailed to each inventor for his signature, but that all mailings of the aforementioned documents were returned to sender. No forwarding addresses were provided.

A grantable petition under 37 CFR 1.47(a) requires:

(1) a petition including proof of the pertinent facts establishing that the joint inventor(s) refuses to join, or cannot be found or reached after diligent effort,
(2) a proper oath or Declaration executed by the available joint inventor(s),
(3) the petition fee of $200, and
(4) the last known address of the omitted inventor(s).

The petition was dismissed on October 30, 2006 for failure to provide items (1) and (2) above.

The present petition was filed on January 30, 2006 with a petition for a one month extension of time and required fee. Four of seven joint inventors signed the declaration filed with the reconsideration petition. Petitioners have shown that non-signing joint inventors Samaniego, Offner, and Devan cannot be located after diligent search.

However, the reconsideration petition lacks item (2).

As to item (2), an oath or declaration for the patent application in compliance with 37 CFR 1.63 and 1.64 still has not been presented. The declaration contains a noninitialed, nondated alteration to Inventor Thewlis' residence and citizenship. 37 CFR 1.52(c) states that "[a]ny interlineation, erasure, cancellation or other alteration of the application papers filed should be made on or before the signing of the accompanying oath or declaration pursuant to 1.63...." This includes the oath or declaration. The Office will not consider whether noninitialed and or nondated alterations were made before or after signing of the oath or declaration but will require a new oath or declaration.

Fortunately, the alterations were to a signing inventor's information. Therefore, pursuant to 37 CFR 1.67(a)(2), Mr. Thewlis may correct his information in a supplemental declaration that lists complete information for all joint inventors, but is executed by only him.

An oath or declaration in compliance with 37 CFR 1.63 and 1.64 signed by the Rule 1.47 applicant on behalf of the non-signing inventor is REQUIRED. See MPEP 409.03(a)

Further correspondence with respect to this matter should be addressed as follows:

By mail: Mail Stop PETITION
Commissioner for Patents
Post Office Box 1450
Alexandria, VA 22313-1450

By hand: U.S. Patent and Trademark Office
Customer Service Window, Mail Stop Petition
Randolph Building
401 Dulany Street
Alexandria, VA 22314

By FAX: (571) 273-8300 - ATTN: Office of Petitions

Telephone inquiries should be directed to the undersigned at (571) 272-3230.

Shirene Willis Brantley
Senior Petitions Attorney
Office of Petitions

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Named Inventor	:	Barger et al
Serial No.	:	11/269,916
Filed	:	November 7, 2005
Art Unit	:	2176
Confirmation Number	:	5707
Examiner	:	Unassigned
Title	:	Automated Media Delivery System
Attorney Docket No.	:	EQUI0001CIP-C

June 12, 2007

Commissioner for Patents
Mail Stop – PETITION
P.O. Box 1450
Alexandria, VA 22313-1450

REQUEST FOR RECONSIDERATION OF PETITION UNDER 37 CFR § 1.47(a)

Dear Examiner:

Applicant respectfully petitions the Commissioner for reconsideration to include the newly signed oath-declaration executed by the inventor Adrian D. Thewlis.

This petition is accompanied by the declaration that lists the complete information for all joint inventors, and executed only by Mr. Adrian D. Thewlis.

Respectfully submitted,

Michael A. Glenn
Reg. No. 30,176

Customer No. 22862

Attorney Docket No. EQUI0001CIP-C

DECLARATION FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address, and citizenship are as stated below next to my name;

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

Automated Media Delivery System

the specification of which (check one) ____ is attached hereto, or _X_ was filed on 11/07/2005 as Application Serial No. 11/269,916 and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)	Priority Claimed	
	Yes	No
_____	____	____
Number Country Day/Month/Year Filed		
_____	____	____
Number Country Day/Month/Year Filed		

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

MICHAEL A. GLENN, Reg. No. 30,176
DONALD M. HENDRICKS, Reg. No. 40,355
CHRISTOPHER PEIL, Reg. No. 45,005
JEFFREY BRILL, Reg. No. 51,198

SEND CORRESPONDENCE TO: CUSTOMER NUMBER **22862**

MICHAEL A. GLENN, 3475 Edison Way, Suite L, Menlo Park, CA 94025

I hereby claim the benefit under Title 35, United States code, Section 119(3)120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

6,964,009	11/08/2005	Patented
Patent Number	Grant Date	Status: Patented, Pending, Abandoned
6,792,575	9/14/2004	Patented
Patent Number	Grant Date	Status: Patented, Pending, Abandoned
Application Ser. No.	Filing Date	Status: Patented, Pending, Abandoned

---

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or **first** inventor: SEAN BARGER

Inventor's signature _____ _____

Residence 222 Marguerite Ave., Mill Valley, CA 94941

Post Office Address Same

Citizenship United States of America

Full name of **second** or joint inventor: CHRISTOPHER SAMANIEGO

Inventor's signature _____ _____
Date

Residence 461 Second Street, San Francisco, California 94107

Post Office Address Same

Citizenship United States of America

Full name of **third** or joint inventor: NELSON H. "ROCKY" OFFNER

Inventor's signature _____ _____
Date

Residence 172 Ardmore Road, Kensington, California 94707

Post Office Address Same

Citizenship United States of America

Full name of **fourth** or joint inventor: ADRIAN D. THEWLIS

Inventor's signature Adrian Thewlis 5/21/07
Date

Residence 24 Adrian Terrace, San Rafael, CA 94903

Post Office Address Same

Citizenship United States of America

Full name of **fifth** or joint inventor: DAVID R. BOYD

Inventor's signature _____ Date _____

Residence 1256 Kearny Street, San Francisco, California 94133

Post Office Address Same

Citizenship United States of America

Full name of **sixth** or joint inventor: DAVID C. SALMON

Inventor's signature _____ Date _____

Residence 42 Sandalwood Court, San Rafael, California 94903

Post Office Address Same

Citizenship United States of America

Full name of **seventh** or joint inventor: JOSHUA N. DEVAN

Inventor's signature _____ Date _____

Residence 25 Stetson Avenue, Lower, Kentfield, California 94904

Post Office Address Same

Citizenship United States of America

PTO/SB/22 (04-07)
Approved for use through 09/30/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARMENT OF COMMERCE

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) **FY 2006** ***(Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).)***	Docket Number (Optional) EQUI0001CIP-C
Application Number 11/269,916	Filed 11/07/2006
For Automated Media Delivery System	
Art Unit 2176	Examiner Unassigned

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

		Fee	Small Entity Fee	
☐	One month (37 CFR 1.17(a)(1))	$120	$60	$
☒	Two months (37 CFR 1.17(a)(2))	$450	$225	$ 225.00
☐	Three months (37 CFR 1.17(a)(3))	$1020	$510	$
☐	Four months (37 CFR 1.17(a)(4))	$1590	$795	$
☐	Five months (37 CFR 1.17(a)(5))	$2160	$1080	$

☒ Applicant claims small entity status. See 37 CFR 1.27.

☐ A check in the amount of the fee is enclosed.

☐ Payment by credit card. Form PTO-2038 is attached.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 07-1445. I have enclosed a duplicate copy of this sheet.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

I am the

☐ applicant/inventor.

☐ assignee of record of the entire interest. See 37 CFR 3.71.
Statement under 37 CFR 3.73(b) is enclosed (Form PTO/SB/96).

☒ attorney or agent of record. Registration Number 30,176

☐ attorney or agent under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34 _____

Signature	June 12, 2007 Date
Michael A. Glenn Typed or printed name	650-474-8400 Telephone Number

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

☒ Total of 1 forms are submitted.

Electronic Patent Application Fee Transmittal

Application Number:	11269916
Filing Date:	07-Nov-2005
Title of Invention:	Automated media delivery system
First Named Inventor/Applicant Name:	Sean Barger
Filer:	Michael Glenn/Della Revecho
Attorney Docket Number:	EQUI0001CIP-C

Filed as Small Entity

Utility Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 2 months with $0 paid	2252	1	225	225

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**225**

Electronic Acknowledgement Receipt

EFS ID:	1866102
Application Number:	11269916
International Application Number:	
Confirmation Number:	5707
Title of Invention:	Automated media delivery system
First Named Inventor/Applicant Name:	Sean Barger
Customer Number:	22862
Filer:	Michael Glenn/Della Revecho
Filer Authorized By:	Michael Glenn
Attorney Docket Number:	EQUI0001CIP-C
Receipt Date:	12-JUN-2007
Filing Date:	07-NOV-2005
Time Stamp:	19:13:24
Application Type:	Utility

Payment information:

Submitted with Payment	yes
Payment was successfully received in RAM	$225
RAM confirmation Number	3564
Deposit Account	071445
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows: Charge any Additional Fees required under 37 C.F.R. Section 1.16 and 1.17	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part /.zip	Pages (if appl.)
1		Petition-EFiled-061207.pdf	276229	yes	7

Multipart Description/PDF files in .zip description		
Document Description	**Start**	**End**
Miscellaneous Incoming Letter	1	1
Petition for review by the Office of Petitions.	2	2
Oath or Declaration filed	3	6
Extension of Time	7	7

Warnings:

Information:

2	Fee Worksheet (PTO-06)	fee-info.pdf	8131	no	2

Warnings:

Information:

Total Files Size (in bytes):	284360

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

ELECTRONIC TRANSMITTAL COVER SHEET

Application Serial No. 11/269,916 Attorney Docket No. EQUI0001CIP-C

I hereby certify that this correspondence is being ELECTRONICALLY TRANSMITTED to the United States Patent and Trademark Office

From: GLENN PATENT GROUP
Customer No.: 22,862
Tel: (650) 474-8400
Fax: (650) 474-8401

on June 12, 2007
Date

Signature

Della Revecho

Typed or printed name of person signing Certificate

Note: Each paper must have its own certificate of transmission, or this certificate must identify each submitted paper.

Attached to this cover sheet please find the following documents:

- Electronic Transmittal Cover Sheet (1 page);
- Request for Reconsideration of Petition (1 page);
- Oath-Declaration of Inventor Adrian D. Thewlis (4 pages); and
- Petition for Extension of Time (1 page)

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

COPY MAILED

JUN 26 2007

OFFICE OF PETITIONS

GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK CA 94025

In re Application of :
Barger, et al. : ON PETITION
Application No.: 11/269,916 :
Filed: November 7, 2005 :
Attorney Docket No.: EQUI0001CIP-C :

This is a decision on the reconsideration petition under 37 CFR 1.47(a), filed June 12, 2007.

The petition is **GRANTED**.

Petitioners have shown that non-signing joint inventors Samaniego, Offner, and Devan cannot be located after diligent search. Petitioners have supplied a declaration executed by the available joint inventors.

The application and papers have been reviewed and found in compliance with 37 CFR 1.47(a). This application is hereby accorded Rule 1.47(a) status.

As provided in 37 CFR 1.47(c), this Office will forward notice of this application's filing to the non-signing inventors at the addresses given in the petition. Notice of the filing of this application will also be published in the Official Gazette.

After the mailing of this decision, the application will be forwarded to Technology Center Art Unit 2176 for examination in due course.

Telephone inquiries should be directed to the undersigned at (571) 272-3230.

Shirene Willis Brantley

Shirene Willis Brantley
Senior Petitions Attorney
Office of Petitions

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

CHRISTOPHER SAMANIEGO
461 SECOND STREET
SAN FRANCISCO CA 94107

COPY MAILED

JUN 26 2007

OFFICE OF PETITIONS

In re Application of :
Barger, et al. : LETTER
Application No.: 11/269,916 :
Filed: November 7, 2005 :
Attorney Docket No.: EQUI0001CIP-C :

Dear Mr. Samaniego:

You are named as a joint inventor in the above-identified United States patent application, filed under the provisions of 35 U.S.C. 116 (United States Code), and 37 CFR 1.47(a), Rules of Practice in Patent Cases. Should a patent be granted on the application you will be designated therein as a joint inventor.

As a named inventor you are entitled to inspect any paper in the file wrapper of the application, order copies of all or any part thereof (at a prepaid cost per 37 CFR 1.19) or make your position of record in the application. Alternatively, you may arrange to do any of the preceding through a registered patent attorney or agent presenting written authorization from you. If you care to join the application, counsel of record (see below) would presumably assist you. Joining the application would entail the filing of an appropriate oath or declaration by you pursuant to 37 CFR 1.63.

Telephone inquiries regarding this communication should be directed to the undersigned at (571) 272-3230. Requests for information regarding your application should be directed to the File Information Unit at (703) 308-2733. Information regarding how to pay for and order a copy of the application, or a specific paper in the application, should be directed to the Certification Division at (571) 272-3150 or 1 (800) 972-6382 (outside the Washington, DC area).

Shirene Willis Brantley

Shirene Willis Brantley
Senior Petitions Attorney
Office of Petitions

ATTORNEYS OF RECORD: GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK CA 94025

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

NELSON H. OFFNER
172 ARDMORE ROAD
KENSINGTON CA 94707

COPY MAILED

JUN 26 2007

OFFICE OF PETITIONS

In re Application of :
Barger, et al. : LETTER
Application No.: 11/269,916 :
Filed: November 7, 2005 :
Attorney Docket No.: EQUI0001CIP-C :

Dear Mr. Offner:

You are named as a joint inventor in the above-identified United States patent application, filed under the provisions of 35 U.S.C. 116 (United States Code), and 37 CFR 1.47(a), Rules of Practice in Patent Cases. Should a patent be granted on the application you will be designated therein as a joint inventor.

As a named inventor you are entitled to inspect any paper in the file wrapper of the application, order copies of all or any part thereof (at a prepaid cost per 37 CFR 1.19) or make your position of record in the application. Alternatively, you may arrange to do any of the preceding through a registered patent attorney or agent presenting written authorization from you. If you care to join the application, counsel of record (see below) would presumably assist you. Joining the application would entail the filing of an appropriate oath or declaration by you pursuant to 37 CFR 1.63.

Telephone inquiries regarding this communication should be directed to the undersigned at (571) 272-3230. Requests for information regarding your application should be directed to the File Information Unit at (703) 308-2733. Information regarding how to pay for and order a copy of the application, or a specific paper in the application, should be directed to the Certification Division at (571) 272-3150 or 1 (800) 972-6382 (outside the Washington, DC area).

Shirene Willis Brantley
Senior Petitions Attorney
Office of Petitions

ATTORNEYS OF RECORD: GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK CA 94025

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

JOSHUA N. DEVAN
25 STETSON AVENUE
LOWER KENTFIELD CA 94904

COPY MAILED

JUN 26 2007

OFFICE OF PETITIONS

In re Application of :
Barger, et al. : LETTER
Application No.: 11/269,916 :
Filed: November 7, 2005 :
Attorney Docket No.: EQUI0001CIP-C :

Dear Mr. Devan:

You are named as a joint inventor in the above-identified United States patent application, filed under the provisions of 35 U.S.C. 116 (United States Code), and 37 CFR 1.47(a), Rules of Practice in Patent Cases. Should a patent be granted on the application you will be designated therein as a joint inventor.

As a named inventor you are entitled to inspect any paper in the file wrapper of the application, order copies of all or any part thereof (at a prepaid cost per 37 CFR 1.19) or make your position of record in the application. Alternatively, you may arrange to do any of the preceding through a registered patent attorney or agent presenting written authorization from you. If you care to join the application, counsel of record (see below) would presumably assist you. Joining the application would entail the filing of an appropriate oath or declaration by you pursuant to 37 CFR 1.63.

Telephone inquiries regarding this communication should be directed to the undersigned at (571) 272-3230. Requests for information regarding your application should be directed to the File Information Unit at (703) 308-2733. Information regarding how to pay for and order a copy of the application, or a specific paper in the application, should be directed to the Certification Division at (571) 272-3150 or 1 (800) 972-6382 (outside the Washington, DC area).

Shirene Willis Brantley
Senior Petitions Attorney
Office of Petitions

ATTORNEYS OF RECORD: GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK CA 94025

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIBDATASHEET

Bib Data Sheet

CONFIRMATION NO. 5707

SERIAL NUMBER	FILING OR 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
11/269,916	11/07/2005 RULE 1.47	707	2176	EQUI0001CIP-C

APPLICANTS

Sean Barger, M. Valley, CA;
Christopher Samaniego, San Francisco, CA;
Nelson H. Rocky Offner, Kensington, CA;
Adrian D. Thewlis, Sausalito, CA;
David R. Boyd, San Francisco, CA;
David C. Salmon, San Rafael, CA;
Joshua N. Devan, Kentfield, CA;

**** CONTINUING DATA **********

This application is a CIP of 09/929,904 08/14/2001 PAT 6,964,009
which is a CON of 09/425,326 10/21/1999 PAT 6,792,575

**** FOREIGN APPLICATIONS **********

IF REQUIRED, FOREIGN FILING LICENSE GRANTED ** SMALL ENTITY **
** 12/08/2005

Foreign Priority claimed	STATE OR COUNTRY	SHEETS DRAWING	TOTAL CLAIMS	INDEPENDENT CLAIMS
☐ yes ☐ no	CA	23	16	5
35 USC 119 (a-d) conditions met ☐ yes ☐ no ☐ Met after Allowance				
Verified and Acknowledged Examiner's Signature Initials				

ADDRESS
22862

TITLE

Automated media delivery system

FILING FEE RECEIVED	
765	FEES: Authority has been given in Paper No. _____ to charge/credit DEPOSIT ACCOUNT No. _____ for following:

☐ All Fees
☐ 1.16 Fees (Filing)
☐ 1.17 Fees (Processing Ext. of time)
☐ 1.18 Fees (Issue)
☐ Other _____
☐ Credit

IFW

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

EDISON H. OFFNER
172 ARDMORE ROAD
KENSINGTON CA 94707

In re Application of :
Barger, et al. : LETTER
Application No.: 11/269,916 :
Filed: November 7, 2005 :
Attorney Docket No.: EQUI0001CIP-C :

Dear Mr. Offner:

You are named as a joint inventor in the above-identified United States patent application, filed under the provisions of 35 U.S.C. 116 (United States Code), and 37 CFR 1.47(a), Rules of Practice in Patent Cases. Should a patent be granted on the application you will be designated therein as a joint inventor.

As a named inventor you are entitled to inspect any paper in the file wrapper of the application, order copies of all or any part thereof (at a prepaid cost per 37 CFR 1.19) or make your position of record in the application. Alternatively, you may arrange to do any of the preceding through a registered patent attorney or agent presenting written authorization from you. If you care to join the application, counsel of record (see below) would presumably assist you. Joining the application would entail the filing of an appropriate oath or declaration by you pursuant to 37 CFR 1.63.

Telephone inquiries regarding this communication should be directed to the undersigned at (571) 272-3230. Requests for information regarding your application should be directed to the File Information Unit at (703) 308-2733. Information regarding how to pay for and order a copy of the application, or a specific paper in the application, should be directed to the Certification Division at (571) 272-3150 or 1 (800) 972-6382 (outside the Washington, DC area).

Shirene Willis Brantley
Senior Petitions Attorney
Office of Petitions

ATTORNEYS OF RECORD: GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK CA 94025

Official Business
Penalty For Private Use, $300

RECEIVED
JUL 10 2007
USPTO MAIL CENTER

NIXIE 945 SE 1 30 07/07/07
RETURN TO SENDER
NOT DELIVERABLE AS ADDRESSED
UNABLE TO FORWARD

BC: 22313145050 *2905-07500-07-25

22313@1450

AN EQUAL OPPORTUNITY EMPLOYER

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

CHRISTOPHER SAMANIEGO
461 SECOND STREET
SAN FRANCISCO CA 94107

COPY MAILED

JUN 26 2007

OFFICE OF PETITIONS

In re Application of :
Barger, et al. : LETTER
Application No.: 11/269,916 :
Filed: November 7, 2005 :
Attorney Docket No.: EQUI0001CIP-C :

Dear Mr. Samaniego:

You are named as a joint inventor in the above-identified United States patent application, filed under the provisions of 35 U.S.C. 116 (United States Code), and 37 CFR 1.47(a), Rules of Practice in Patent Cases. Should a patent be granted on the application you will be designated therein as a joint inventor.

As a named inventor you are entitled to inspect any paper in the file wrapper of the application, order copies of all or any part thereof (at a prepaid cost per 37 CFR 1.19) or make your position of record in the application. Alternatively, you may arrange to do any of the preceding through a registered patent attorney or agent presenting written authorization from you. If you care to join the application, counsel of record (see below) would presumably assist you. Joining the application would entail the filing of an appropriate oath or declaration by you pursuant to 37 CFR 1.63.

Telephone inquiries regarding this communication should be directed to the undersigned at (571) 272-3230. Requests for information regarding your application should be directed to the File Information Unit at (703) 308-2733. Information regarding how to pay for and order a copy of the application, or a specific paper in the application, should be directed to the Certification Division at (571) 272-3150 or 1 (800) 972-6382 (outside the Washington, DC area).

Shirene Willis Brantley

Shirene Willis Brantley
Senior Petitions Attorney
Office of Petitions

ATTORNEYS OF RECORD: GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK CA 94025

Organization _____ Bldg./Room _____

UNITED STATES PATENT AND TRADEMARK OFFICE
P.O. Box 1450
Alexandria, VA. 22313-1450
If Undeliverable Return In Ten Days

Official Business
Penalty For Private Use, $300

AN EQUAL OPPORTUNITY EMPLOYEF

RECEIV
JUL 10 2007
USPTO MAIL CENTER

94107+1438

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

JOSHUA N. DEVAN
25 STETSON AVENUE
LOWER KENTFIELD CA 94904

COPY MAILED

JUN 26 2007

OFFICE OF PETITIONS

In re Application of :
Barger, et al. : LETTER
Application No.: 11/269,916 :
Filed: November 7, 2005 :
Attorney Docket No.: EQUI0001CIP-C :

Dear Mr. Devan:

You are named as a joint inventor in the above-identified United States patent application, filed under the provisions of 35 U.S.C. 116 (United States Code), and 37 CFR 1.47(a), Rules of Practice in Patent Cases. Should a patent be granted on the application you will be designated therein as a joint inventor.

As a named inventor you are entitled to inspect any paper in the file wrapper of the application, order copies of all or any part thereof (at a prepaid cost per 37 CFR 1.19) or make your position of record in the application. Alternatively, you may arrange to do any of the preceding through a registered patent attorney or agent presenting written authorization from you. If you care to join the application, counsel of record (see below) would presumably assist you. Joining the application would entail the filing of an appropriate oath or declaration by you pursuant to 37 CFR 1.63.

Telephone inquiries regarding this communication should be directed to the undersigned at (571) 272-3230. Requests for information regarding your application should be directed to the File Information Unit at (703) 308-2733. Information regarding how to pay for and order a copy of the application, or a specific paper in the application, should be directed to the Certification Division at (571) 272-3150 or 1 (800) 972-6382 (outside the Washington, DC area).

Shirene Willis Brantley

Shirene Willis Brantley
Senior Petitions Attorney
Office of Petitions

ATTORNEYS OF RECORD: GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK CA 94025

Organization _____ Bldg./Room _____

UNITED STATES PATENT AND TRADEMARK OFFICE
P.O. Box 1450
Alexandria, VA. 22313-1450
If Undeliverable Return In Ten Days

Official Business
Penalty For Private Use, $300

AN EQUAL OPPORTUNITY EMPLOYER

UNITED STATES POSTAGE
$ 00.41⁰
MAILED FROM ZIP CODE

RECEIVED
JUL 25 2007
USPTO MAIL CENTER

FORWARDED
T Douglas
Petaluma, CA 94952

NIXIE 949 DA 1 00 07/21/07
RETURN TO SENDER
NOT DELIVERABLE AS ADDRESSED
UNABLE TO FORWARD
BC: 22313145050 *1417-10207-26-41

22313@1450

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

JOSHUA N. DEVAN
25 STETSON AVENUE
LOWER KENTFIELD CA 94904

In re Application of :
Barger, et al. : LETTER
Application No.: 11/269,916 :
Filed: November 7, 2005 :
Attorney Docket No.: EQUI0001CIP-C :

Dear Mr. Devan:

You are named as a joint inventor in the above-identified United States patent application, filed under the provisions of 35 U.S.C. 116 (United States Code), and 37 CFR 1.47(a), Rules of Practice in Patent Cases. Should a patent be granted on the application you will be designated therein as a joint inventor.

As a named inventor you are entitled to inspect any paper in the file wrapper of the application, order copies of all or any part thereof (at a prepaid cost per 37 CFR 1.19) or make your position of record in the application. Alternatively, you may arrange to do any of the preceding through a registered patent attorney or agent presenting written authorization from you. If you care to join the application, counsel of record (see below) would presumably assist you. Joining the application would entail the filing of an appropriate oath or declaration by you pursuant to 37 CFR 1.63.

Telephone inquiries regarding this communication should be directed to the undersigned at (571) 272-3230. Requests for information regarding your application should be directed to the File Information Unit at (703) 308-2733. Information regarding how to pay for and order a copy of the application, or a specific paper in the application, should be directed to the Certification Division at (571) 272-3150 or 1 (800) 972-6382 (outside the Washington, DC area).

Shirene Willis Brantley
Senior Petitions Attorney
Office of Petitions

ATTORNEYS OF RECORD: GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK CA 94025

Organization _____ Bldg./Room _____

UNITED STATES PATENT AND TRADEMARK OFFICE
P.O. Box 1450
Alexandria, VA. 22313-1450
If Undeliverable Return In Ten Days

Official Business
Penalty For Private Use, $300

AN EQUAL OPPORTUNITY EMPLOYER

UNITED STATES POSTAGE
$ 00.41⁰
MAILED FROM ZIP CODE

RECEIVED
JUL 25 2007
USPTO MAIL CENTER

FORWARDED
7 Douglas
Petaluma, CA 94952

NIXIE 949 DA 1 00 07/21/07
RETURN TO SENDER
NOT DELIVERABLE AS ADDRESSED
UNABLE TO FORWARD
BC: 22313145050 *1417-10207-26-41

22313@1450

Form 1449 (Modified)	Atty. Docket No.	Serial No.:
	EQUI0001CIP-C	11/269,916
Information Disclosure Statement By Applicant	**Applicant:** Sean Barger, et al.	
	Filing Date:	**Group:** 4112
(Use Several Sheets if Necessary)	November 7, 2005	**Confirmation No:** 5707

U.S. Patent Documents

Examiner Initial	No.	Patent No.	Issue Date	Patentee	Class	Sub-class	Filing Date
	1	5,088,052	2/1/1992	Spielman, et al.			
	2	5,355,472	10/1/1994	Lewis			
	3	5,530,852	6/1/1996	Meske, Jr., et al.			
	4	5,701,451	12/1/1997	Rogers, et al.			
	5	5,708,845	1/1/1998	Wistendahl, et al.			
	6	5,710,918	1/1/1998	Lagarde, et al.			
	7	5,737,619	4/1/1998	Judson			
	8	5,745,908	4/1/1998	Anderson, et al.			
	9	5,758,110	5/26/1998	Boss, et al.			
	10	5,761,655	6/2/1998	Hoffman, Michael T.			
	11	5,793,964	8/1/1998	Rogers, et al.			
	12	5,819,261	10/6/1998	Takahashi, et al.			
	13	5,822,436	10/1/1998	Rhoads			
	14	5,845,084	12/1/1998	Cordell, et al.			
	15	5,845,299	12/1/1998	Arora, et al.			
	16	5,860,068	1/1/1999	Cook			
	17	5,860,073	1/1/1999	Ferrel, et al.			
	18	5,861,881	1/1/1999	Freeman, et al.			
	19	5,862,325	1/1/1999	Reed, et al.			
	20	5,864,337	1/26/1999	Marvin, John			
	21	5,870,552	2/1/1999	Dozier, et al.			
	22	5,880,740	3/1/1999	Halliday, et al.			
	23	5,890,170	3/1/1999	Sidana			
	24	5,895,476	4/1/1999	Orr, et al.			
	25	5,895,477	4/20/1999	Orr, et al.			
	26	5,903,892	5/11/1999	Hoffert, et al.			
	27	5,937,160	8/1/1999	Davis, et al.			
	28	5,943,680	8/24/1999	Ohga, et al.			
	29	5,956,737	9/21/1999	King, et al.			
	30	6,009,436	12/1/1999	Motoyama, et al.			
	31	6,456,305	9/1/2002	Qureshi, et al.			
	32	6,563,517	5/1/2003	Bhagwat, et al.			
	33	6,591,280	7/1/2003	Orr			
	34	6,623,529	9/1/2003	Lakritz			

Published U.S. Patent Application

Examiner Initial	No.	Document No.	Publication Date	Assignee	Class	Sub-class	Translation Yes	Translation No

Foreign Patent or Published Foreign Patent Application

Examiner Initial	No.	Document No.	Publication Date	Assignee	Class	Sub-class	Translation Yes	Translation No
	35	EP 0747842	12/11/1996	International Business Machines Corp.				
	36	EP 0782085	7/2/1997	International Business Machines Corp.				
	37	EP 0818907	1/14/1998	AT&T Corp				
	38	EP 0843276	5/20/1998	Canon Information Systems Inc.				
	39	EP 0876034	11/4/1998	International Business Machines Corp.				
	40	EP 0883068	12/9/1998	Home Information Services, Inc.				
	41	EP 0886409	12/23/1998	Digital Vision Laboratories Corporation				
	42	EP 0895171	2/3/1999	Neoforma, Inc.				
	43	EP 0926607	6/30/1999	Ricoh KK				
	44	EP 0949571	10/13/1999	Xerox Corp				
	45	WO 98/40842	9/17/1998	Computer Information and Sciences, Inc.				
	46	WO 98/43177	10/1/1998	Intel Corp				
	47	WO 97/49252	12/24/1997	Senthilkumar, et al.				
	48	AU-A-53031/98	8/27/1998	Dudley, John Mills				

Other Documents

Examiner Initial	No.	Author, Title, Date, Place (e.g. Journal) of Publication
	A	Sakaguchi, et al.; " A browsing tool for multi-lingual documents for users without multi-lingual fonts"; 1996; ACM International Conference On Digital Libraries, pp. 63-71
	B	Zaiane, et al.; "Mining multimedia data"; Nov. 1998; ACM Conference of the Center for Advanced Studies on Collaborative research, pp. 1-18
	C	BULTERMAN, DICK.C.A.; Models, Media and Motion: Using the Web to Support Multimedia Documents; Proceedings of 1997 Int'l Conf on Multimedia Modeling; p. 17-20; November 1997; SINGAPORE
	D	MOHLER, J.L.; Migrating Course Materials to the World Wide Web: A Case Study of the Department of Technical Graphics at Purdue University; Computer Networks and ISDN Systems; Vol. 30, Issues 20-21, p,1981-1990; November 12, 1988
	E	DOBSON, R.; Animating Your Web Pages with Direct Animation; Web Techniques; vol.3, no. 6, p. 49-52; June 1998
	F	BERINSTEIN, Paula; "The Big Picture; Text and Graphics on UMI's ProQuest Direct: The Best (Yet) of Both Words"; March 1997; retrieved on 3/23/04 from website: http://www.infotoday.com/online/MarOL97/picture3.html

	G	McNeil, Sara; Research Interests; retrieved on March 18, 2004 from website: http//www.coe.uh.edu/`smcneil/research.htm
	H	Tables of Contents service for Computers & Geosciences; Copyright 1997; Computers and GeoSciences, Volume 23, Issue 5, retrieved on 3/18/04 from website: http://library.iem.ac.ru/comp&geo/00983004/sz977014.html

Examiner's Signature_____ Date_____

Examiner: Initial citation considered. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 747 842 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
11.12.1996 Bulletin 1996/50

(51) Int. Cl.[6]: **G06F 17/30**

(21) Application number: **96108976.0**

(22) Date of filing: **05.06.1996**

(84) Designated Contracting States:
AT BE CH DE ES FR GB IT LI NL SE

(30) Priority: **07.06.1995 US 479481**

(71) Applicant: **International Business Machines Corporation**
Armonk, N.Y. 10504 (US)

(72) Inventors:
- **Rogers, Richard Michael**
  Beacon, New York 12508 (US)
- **Lagarde, Konrad Charles**
  Milford, Connecticut 06460 (US)

(74) Representative: **Schäfer, Wolfgang, Dipl.-Ing.**
IBM Deutschland
Informationssysteme GmbH
Patentwesen und Urheberrecht
70548 Stuttgart (DE)

(54) A web browser system

(57) A World Wide Web browser makes requests to web servers on a network which receive and fulfill requests as an agent of the browser client, organizing distributed sub-agents as distributed integration solution (DIS) servers on an intranet network supporting the web server which also has an access agent servers accessible over the Internet. DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext document, requested information from multiple data bases located at different types of databases geograhically dispersed, performing calculations, formatting, and other services prior to reporting to the web browser or to other locations, in a selected format, as in a display, fax, printer, and to customer installations or to TV video subscribers, with account tracking.

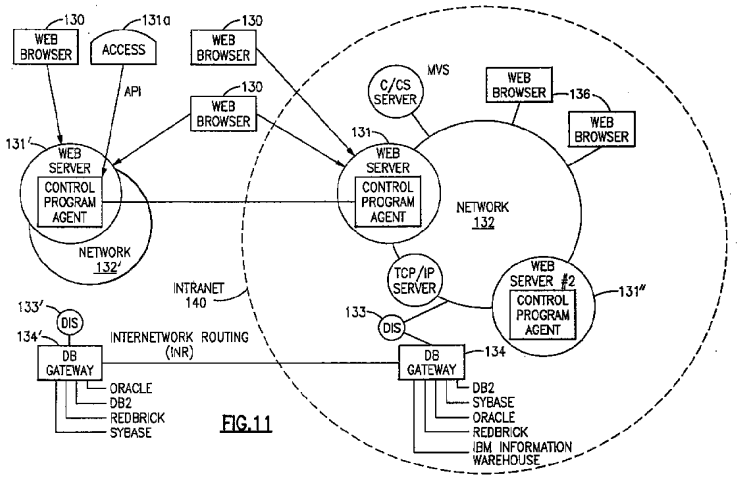


FIG.11

EP 0 747 842 A1

Description

Copyright Authorization

FIELD OF THE INVENTION

This invention is related to computers and computer systems and particularly to a method and system for use of the World Wide Web and other sources of information and for utilization of existing equipment advantageously for web server data access over networks and the Internet.

RELATED APPLICATIONS

This application entitled "A Web Browser System", is related to other United States of America Patent applications filed concurrently herewith, and specifically to the applications entitled "Computer Network for WWW Server Data Access over Internet", USSN 08/474,571, filed June 7, 1995; and "A Service Agent for Fulfilling requests of a Web Browser", USSN 08/474,576, filed June 7, 1995; and "A Sub-Agent Service Agent for Fulfilling Requests of a Web Browser", USSN 08/474,575, filed June 7, 1995; and "A Method for Fulfilling Requests of a Web Browser" USSN 08/474,577, filed June 7, 1995; and "A Method for Distributed Task Fulfillment of Web Browser Requests",USSN 08/474,572, filed June 7, 1995.

These applications have a common assignee, International Business Machines Corporation, Armonk, New York.

GLOSSARY OF TERMS

While dictionary meanings are also implied by certain terms used here, the following glossary of some terms may be useful.

World Wide Web (WWW)

The Internet's application that lets people seeking information on the Internet switch from server to server and database to database by clinking on highlighted words or phrases of interest. An Internet WWW server supports clients and provides information.

Home page

A multi-media table of contents that guides a web user to stored information about an organization on the Internet.

Gopher

A menu-based search scheme, which as developed at the University of Minnesota, lets a user reach a destination on the Internet by selecting items from a series of text menus.

Access Agent

A logical component that provides support for different access protocols and data streams -- Frame Relay, HDLC (High Data Link Control) CBO (Continuous bit Operations, ATM (Asynchronous Transfer Mode), or TCP/IP.

Application Processing Agent

A data processing agent running in a server data processing system which performs tasks based on received requests from a client in a distributed environment. In our preferred embodiment, our application processing agent for database retrieval is our DIS server, a data interpretation system server and database gateway which is coupled to our web server HTTPD via a network. In our preferred embodiment an application processing agent employs executable object programs as command file objects, which in the preferred embodiment are capsule objects.

Client

A client is a computer serviced by the server which provides commands to the server.

Data Interpretation System (DIS).

IBM's object oriented decision support tool.

Capsule

A DIS capsule is a program created by a DIS programmer and executed in the DIS environment. A DIS capsule is a preferred example of a capsule object. A capsule object is a specialized form of a command file (which is a list of commands to be executed, as in an EXEC or *.BAT batch file. The capsule object is created with an object environment, as is supplied by IBM's DIS. Other object environments are IBM's SOM and DSOM, and Microsoft's COM environment.

Internet

The connection system that links computers worldwide in a web.

Server

A machine which supports one or more clients and is part of the web. Any computer that performs a task at the command of another computer is a server.

Slip or PPP connection.

Serial-line Internet protocol and point-to-point protocol, respectively, for providing a full access connection for a computer to the Internet. Transmission control protocol/Internet protocol. A packet switching scheme the Internet uses to chop, route, and reconstruct the data it handles, from e-mail to video.

InterNetwork Routing (INR)

The link between systems which routes data from one
physical unit to another according to the applicable protocol. The protocol will employ a URL address for Internet locations.

URL

Universal resource locater, a Web document version of an e-mail address. URLs are very cumbersome if they belong to documents buried deep within others. They can be accessed with a Hyperlink.

Web browser

An program running on a computer that acts as an Internet tour guide, complete with pictorial desktops, directories and search tools used when a user "surfs" the Internet. In this application the Web browser is a client service which communicates with the World Wide Web.

HTTPD

An IBM OS/2 Web Server or other server having Hypertext Markup Language and Common Gateway Interface. In our preferred embodiment, the HTTPD incorporates our control program agent and is supported by an access agent which provides the hardware connections to machines on the intranet and access to the Internet, such as TCP/IP couplings.

HTTP Hypertext transfer protocol

Hypertext transfer protocol. At the beginning of a URL "http:" indicates the file contains hyperlinks.

Hyperlink

A network address embedded in a word, phrase, icon or picture that is activated when you select the highlighted tidbit. Information about that item is currently retrieved to the client supporting a Web browser.

HyperText Markup Language (HTML)

HTML is the language used by Web servers to create and connect documents that are viewed by Web clients. HTML uses Hypertext documents. Other uses of Hypertext documents are described in U.S. Patents 5,204,947, granted April 20, 1993 to Bernstein et al.; 5,297,249, granted March 22, 1994 to Bernstein et al.; 5,355,472, granted October 11, 1994 to Lewis; all of which are assigned to International Business Machines Corporation, and which are referenced herein.

BACKGROUND OF THE INVENTIONS

The Internet is not a single network, it has no owner or controller, but is an unruly network of networks, a confederation of many different nets, public and private, big and small, that have agreed to connect to one another. An intranet is a network which is restricted and while it may follow the Internet protocol, none or only part of the network available from outside a "firewall" surrounding the intranet is part of the agreed connectionto the Internet. The composite network represented by these networks relies on no single transmission medium, bi-directional communication can occur via satellite links, fiber-optic trunk lines, phone lines, cable TV wires and local radio links. When your client computer logs onto the Internet at a university, a corporate office or from home, everything looks local, but the access to the network does cost time and line charges.

Until recently, "cruising or surfing" the Internet was a disorienting, even infuriating experience, something like trying to navigate without charts. The World Wide Web, a sub-network of the Internet, introduced about two years ago, made it easier by letting people jump from one server to another simply by selecting a highlighted word, picture or icon (a program object representation) about which they want more information -- a maneuver known as a "hyperlink". In order to explore the WWW today, the user loads a special navigation program, called a "Web browser" onto his computer. While there are several versions of Web browsers, IBM's example is the new WebExplorer which offers users of IBM's OS/2 Warp system software a consistent, easy to use desktop of pictorial icons and pull down menus. As part of a group of integrated applications available from IBM for OS/2 Warp called the IBM Internet Connection, lets users log onto the Internet.

To this point the World Wide Web (Web) provided by Internet has been used in industry predominately as a means of communication, advertisement, and placement of orders. As background for our invention there now exists a number of Internet browsers. Common examples are NetScape, Mosaic and IBM's Web Explorer. Browsers allow a user of a client to access servers located throughout the world for information which is stored therein and provided to the client by the server by sending files or data packs to the requesting client from the server's resources. An example of such a request might be something called GSQL (get SQL) which was a NCSA language and CGI server program developed to getting textual results for a client caller. Developed by Jason Ng at the University of Illinois, this document provided a way to map SQL forms against a database, and return the textual results to the client caller. This system is unlike the present invention, and presents difficulties which are overcome by our described system.

These servers act as a kind of Application Processing Agent, or (as they may be referred to) an "intelligent agent", by receiving a function request from a client in response to which the server which performs tasks, the function, based on received requests from a client in a distributed environment. This function shipping concept in a distributed environment was first illustrated by CICS as a result of the invention described in U.S. Patent 4,274,139 to Hodgkinson et al. This kind of function, illustrated by CICS and its improvements, has been widely used in what is now known as transaction processing. However, servers today, while performing many functions, do not permit the functions which we have developed to be performed as we will describe.

Now, "surfing" the Internet with the WWW is still a time consuming affair, and the information received is not generally useful in the form presented. Even with 14,400 baud connection to the Internet much line time is tied up in just keeping going an access to the Internet, and the users don't generally know where to go. Furthermore the coupling of resources available on a company's intranet and those available on the Internet has not been resolved. There is a need to reduce gateways, make better use of existing equipment, and allow greater and more effective usage of information which is resident in many different databases on many different servers, not only within a homogeneous network but also via the Internet and heterogeneous network systems.

The problems with creating access to the world via the Internet and still to allow internal access to databases has been enormous. However, the need for a system which can be used across machines and operating systems and differing gateways is strongly felt by users of the Internet today. Anyone who has spent hours at a WWW browser doing simple task knows how difficult it still is to navigate thorough arcane rules without knowing where to go and even if you know what you are doing spending hours doing routine tasks. Many needs exist. As one important instance, until now we know of no way to access data on multiple databases of different types using a single user request from a client.

This and other difficulties are solved by our invention.

SUMMARY OF THE INVENTIONS

In accordance with our invention needless user intervention is eliminate or greatly reduced with a Web server supports an HTTPD which is provided with the capabilities of our control program agent which organizes sub-agents supporting command file objects or capsules to perform tasks in support of a Web browser's request for service as programmable functions receiving parameters as input and providing as their output handled by the control program agent task completed results for reporting in accordance with the Web browser request in the form and to the location determined by a request and handling these request without needless user intervention.

In accordance with our invention, we have created a way to allow Web users to request information that is created by a data interpretation system (DIS) and then presented by a web server to the user of the web. Our solution provides a way of requesting and processing and presenting information on the Web. In the process, data is retrieved from multiple sources which may be located remotely and accessed via an intranet routing and via the Web Internet and processed by our decision support capsules. Now companies and universities, and other users that want to access data located on different databases, want that data processed and formatted, and presented in a form the user desires, such as a graphical format. Our solution permits users to access information from various sources and obtain information at a desired location as a result of a single request which is responded to by an organization of facilities and command file sub-agent decision support capsule objects by our command program agent. Users of the information can be internal to a company, or external. The result can be furnished to a user at a location which is internal or external to the company, and as specified at a specified location with a form and format desired. This allows a report to be managed by the web support services we provide, and in a form consistent with the request, but without requiring a consistent interface solution.

In order to create a way for Web users to request information generation we provide a web server with a control program agent which is linked to a decision support tool of a data interpretation system server, the application processing agent, and then have that server retrieve, process, and format information which is presented to the user on the Web by the Web server. In our preferred embodiment, we have provided a link between a Hypertext Markup Language (HTML) document using a common gateway interface, and open data interpretation system server (ODAS). As a result, Web clients can request DIS reports to be generated, specify the parameters to be used in generating the reports, and then view the report results on a Web home page. The DIS capsule can generate graphical information, such as colored pie charts, line graphs, bar graphs, and other forms of generated information. Since the Web server is capably of presenting the results in desired formats, the full capabilities of a DIS report are utilized.

Our invention provides a method and system for allowing a user of a client to access and assemble information structured and reported to the user in accordance with his desires, selecting information for disparate servers which are located within a network can be an intranet or internal network, such as a LAN or WAN not normally accessible to the Internet, or coupled to the Internet. In accordance with our invention one can access data on multiple databases of different types using a single user request from a client. We also allow the facility for providing specialized specific requests to be created for routine use, as well as the facility to formulate generalized or specialized ad hoc requests. In addition, we provide besides query and update capability, the ability to perform calculations with respect to any retrieved data, to format the information in text or in graphics, and the facility of presenting the results to the client for display or other use.

The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client in a text or graphics display as a location specified by the request.

Our invention of providing a web server with a control program agent allows organization of decision support functions to be executed by application processing agent servers located throughout the Internet to gather and supply information not presently available with any existing resources without the need of endless intervention on the part of a requesting user of the WWW; further enabling an ordinary user to take advantage of expertise which is provided by programmable sub-agents developed by those with particular expertise in a given area as well as enabling use of standard routines commonly needed.

These improvements are accomplished by providing for Web clients to request information from an application processing agent in which the application processing agent server performs tasks based on received requests from a client in a distributed environment by a web server supported by an access agent link and control program agent which in turn causes a decision support function to be executed by the application processing agent server. This is performed within the distributed environment by the application processing agent server which forms part of a network coupled to and under control of the control program agent. According to our invention the decision support function is provided by a data interpretation system which functions as part of the application processing agent and the decision support function is programmable and generated by a data interpretation system, DIS or other decision support element performing

similar functions, and provided in a form accessible to our control program agent which presents the output generated to be presented to the user on the Web who made the initial request. We have provided, in a preferred embodiment, a link between IBM's Hypertext Markup Language (HTML), the Common Gateway Interface (CGI), and the Open DIS Access Server (ODAS), all of which may be used on machines which are are commercially available from IBM. In order to write additional functions which develop our invention, the reader is referred to the Medaphor Data Interpretation System publication "Developing Applications with OpenDIS Access Service, Version 2.0, available from IBM, First Edition (September 1994) Part Number 315-0002-01 which is incorporated herein by reference.

Our improvements relating to our control program agent is in accordance with our preferred embodiment is normally installed on an IBM HTTPD which is an IBM OS/2 Web Server or other server having Hypertext Markup Language and Common Gateway Interface. In our preferred embodiment, the HTTPD incorporates our control program agent and is supported by an access agent which provides the hardware connections to machines on the intranet and access to the Internet, such as TCP/IP couplings. The hardware for the Web server is thus a workstation, such as IBM's PS/2 model 80 with OS/2. However, the HTTPD can be installed in PCs and upwardly also in machines which range across IBM's line of computers from powerful personal computers to mainframe systems which support MVS, IBM's operating system which enables multiple kinds of operating systems, including "UNIX" to coexist on a single platform. As a result of our invention Web clients can request DIS reports to be generated by the application processing agent specifying the parameters to be used in generating the reports, and then as a result of the request receive a result which is presented, as a visual display or otherwise, on a Web page for use by the requesting user. Our machine implementation allows a user having DIS access to generate graphical information such as colored pie charts, line graphs, bar graphs, etc. Since Web browsers such as IBM's Web Explorer are capable of displaying these formats, all the functions which can be created by a DIS capsule can be utilized by a user of our invention.

According to our improved method, an Internet World Wide Web user connects to a Web server through the use of a Web browser. In accordance with our preferred embodiment, we use HTML as the language used by Web servers to create and connect documents that are viewed by Web clients. HTML is an example of a hypertext language having the facility of clicking on a highlighted word, string of words, or image in order to move to another HTML document or invoke a program on the server. An example of a Web client would be a machine used by a person using IBM's Web Explorer product. In using our invention a user may click on the hypertext in a document to reference a function which will be provided by an application processing agent server. The user is able to connect to another document that may be on another Web server. HTML commands are used to reference other documents. HTML is used to reference programs available on a server, and pass parameters to those programs. The application processing agent server executes a program when it is referred to by a Web client via a control program agent resident, preferably, in a Web server.

The Web client selects the information that they wish to view by using the HTML created page, the Web server takes the client request and passes it to a C program implementation of our control program agent. Web servers, such as HTTPD for OS/2, with our control program agent are able to provide access to executable programs through the use of the Common Gateway Interface (CGI). When a program is referenced by the HTML, any parameters are passed to the program and it is executed. In our preferred embodiment we have used CGI to invoke programs that we have developed that will interface with the DIS product. CGI is an example of a software gateway from a Web server to programs outside the Web server application.

The control program agent that is called in this instance by the Web server through the CGI interface, passes the Web client request along to a data interpretation system DIS via a Open Dis Access Server (ODAS). ODAS is a feature of a data interpretation system DIS that allows programs to initiate DIS functions, such as invoking DIS capsules. Our control program agents interface with DIS through ODAS to submit DIS capsules for execution. DIS capsules are basically programs that DIS application programmers create with the DIS programming language. In accordance with our invention, we have written capsules which are executed as a DIS capsule on a server to gather data from one or more databases, process that data, and create a report in one of many formats, which we will describe by way of example. After the DIS capsule completes executing, in accordance with our preferred embodiment, the results that are generated during execution of a capsule are stored in a file on the application processing server.

After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet. Our control program agents using the CGI interface can create HTML commands dynamically. In this way a program can present information on a Web browser for the Web client.

After the DIS capsule has created the file containing the report request results, the control program creates HTML statements dynamically that display the report results to the Web browser.

Alternative means of presenting the data are shown by alternative routing. The user requesting the report may wish to have the report results sent to another location in addition to or instead of displaying the report results to the Web browser. This information is provided during the request phase. As a result of the alternative report request, and according to the parameters indicated therein, the report results can be sent by the control program via electronic mail, i.e. TCP/IP Sendmail facility and Lotus Notes, to one or more locations on the Internet. The report results can be sent as a file and as a note. The request can request a voice response, which can be routed to a voice response unit. Thus, with

a call to a translator, the text can be converted to voice, and even translated along the way. The report results can also be sent to a fax machine, or to a computer that has the capability of receiving fax data.

We use these report concepts to present report files created by DIS capsules on the Web client display.

These and other improvements are set forth in the following detailed description. For a better understanding of the invention with advantages and features, refer to the description and to the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 shows schematically an overview of the preferred embodiment and particularly shows a typical distributed computer system which has internal and external networks including the Internet to connect clients to World Wide Web servers and other servers within the system in which our invention is situate.

FIGURE 2 shows a inquiry screen (home page) which is displayed on a client after the client is coupled to its server (which may be an Internet server) by a Web browser.

FIGURE 3 is a next screen which illustrates how a request is made according to a users desires, making a request in accordance with our invention with an input screen shown.

FIGURE 4 is a sample result screen which is returned to the client after the requested service is provided by the computer system network in accordance with our invention formatted according to the specifications of a DIS capsule.

FIGURE 5 is a next screen which illustrates how a request is made according to a users desires, making a request in accordance with our invention by selection from a menu and through the use of image mapping.

FIGURE 6 is an example of a graphical result screen which is returned to the client after the requested service is provided by the computer system network in accordance with our invention.

FIGURE 7 illustrates a flowchart showing data flow between a web server and decision support system tool such as IBM's Data Interpretation System (DIS), and shows the coupling of a Web client to a Web server and the coupling of a request to execute a DIS capsule and the coupling within the Web server from ODAS to a distributed DIS LAN with heterogeneous connections to multiple databases.

FIGURE 8 illustrates as a flow chart the functions of the control program for the web server.

FIGURE 9 illustrates by way of example a DIS capsule that creates a text report file.

FIGURE 10 illustrates by way of example a DIS capsule that creates a graphical report file.

FIGURE 11 illustrates an alternative configuration of the network system as it may be employed for permitting access to information available through homepages and in data warehouses where access to the homepage or database may or may not be restricted by a firewall.

(Note: For convenience of illustration, in the formal drawings FIGURES may be separated in parts and as a convention we place the top of the FIGURE as the first sheet, with subsequent sheets proceeding down and across when viewing the FIGURE, in the event that multiple sheets are used.)

Our detailed description explains the preferred embodiments of our invention, together with advantages and features, by way of example with reference to the following drawings.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 illustrates a information delivery solution of a typical combination of resources including clients and servers which may be personal computers or workstations as clients, and workstations to mainframe servers as servers. The various elements are coupled to one another by various networks, including LANs, WANs, and other networks, which may be internal SNA networks or other like internal networks, and also providing access to the Internet, which couples the system to the world via Internet.

The Preferred Embodiment

Turning now to our invention in greater detail, it will be seen from FIGURE 1 that our preferred embodiment provides a Web browser 10, which is coupled to a Web server 11. Our Internet WWW browser is an intelligent computer system, such as an IBM PS/2, or other computer, an IBM ThinkPad, an RS/6000 works as well and connections are made to the network via OS/2 WARP Connect, an IBM product. The Internet Web browser in the intelligent computer system which performs the Web browser function has IBM Web Explorer, or NetScape or Mosaic installed thereon. This computer system 10 is bi-directionally coupled with the OS/2 WARP Connect facility over a line or via a wireless system to our preferred computer system which we call our Web server. This system is a PS/2 or RS/6000 or other similar system which includes our control program agent 73, which will be discussed below. Web server 11, in our preferred embodiment is coupled again bi-directionally via a line or wireless coupling to a computer system, such as a PS/2 or RS/6000 or other server which supports and performs the server function of ODAS server 12, which is coupled to the distributed DIS network, here shown as LAN 13. ODAS 12 may be located on the same server as the Web server 11 or be located at a separate service machine, such as an IBM Digital Server. The Web server is logically coupled to our

application processing agent server via a network. We call our application processing agent server a DIS File server 14 because is comprises a data interpretation system which supports the decision support functions we provide which is today most inexpensively provided by an IBM computer system which supports OS/2. In our preferred embodiment, the intranet network is a LAN. Thus the components of the DIS LAN 13 comprise a DIS File Server 14, a general purpose workstation 15 which can be used for capsule development, a local database server 16, a Capsule Server 17 for storing a plurality of DIS capsules ready for user, a Database Gateway Server 18 which performs the gateway functions to access databases which are linked to it, these databases include geographically distributed databases which can be located, for instance, in Chicago, New York, Dallas, Los Angeles, and each of which can be a different supported database, such as DB2 database 19, ORACLE database 20, Sybase database 21, Redbrick database 22. In our preferred embodiment all servers are coupled with a conventional LAN or WAN connection, with a preferred IBM token ring shown. Reference should also be had to our alternative preferred embodiment discussed below with respect to FIGURE 11.

Thus, in connection with the preferred embodiment of FIGURE 1 as well as with respect to FIGURE 11 it would be appreciated from the schematic overview illustrated by FIGURE 1 and FIGURE 11 that our invention may be employed in a distributed computer system environment which has internal or intranet networks represented in our preferred embodiment by the DIS Network 13 and external networks including the Internet to connect clients to World Wide Web servers and other servers within the system in which our invention is situate. Our invention makes use of the entire network. The Web browser 10 can make a request to the Web Server 11 for a report. The Web server 11 with the facilities we provide causes the application processing agent which includes our DIS server 14 and its supporting communication server, the database gateway server 18, to act as an agent to gather data from one or more of the multiple databases, including the local database 16, DB2 database 19, ORACLE database 20, Sybase database 21, Redbrick database 22. Further details with respect to the use of our invention for database retrieval of information from multiple databases are provided as to the actions of the application processing agent functions of the database server(s) 18 with reference to FIGURE 7.

Thus, returning to our simplified and preferred embodiment, FIGURE 2 shows a inquiry screen (home page) 29 in the form which is displayed on a client after the client is coupled to its server (which may be an Internet Web server 11) by a Web browser 10. The entire screen contains information and a plurality of objects. Once the home page is displayed, with appropriate descriptive guidance as illustrated by the FIGURE 2, the user can interact, for example, by clicking on image objects 30, 31, 32, 33, 34. As a example should the user want to make a special request in accordance with our invention, he could click on image 30. This would take the user to the next screen, illustrated by FIGURE 3. Alternatively the user could select by clicking on image 31 another menu screen, illustrated by FIGURE 5. At this point also, a specialized format could be selected by double clicking first on a format select image illustrated by image objects representing access to menu screens 32, 33, 34, one or more of which a gopher.

The use of selection of icon image object is a function provided by HTML and programmers knowing this language can readily create variants to the images and functions we have illustrated. Thus incorporated within the drawings are to be understood to be the variants that can thus be created using our examples, as well as extensions and combinations thereof.

When the user selected image 30 by clicking on the image 30, FIGURE 3 appears. FIGURE 3 is the next screen which illustrates how a request is made according to a users desires, making a request in accordance with our invention with an input screen shown. The content of FIGURE 3 is preformatted 40 except for the user entries which are to be entered in the data input fields 41. In this example the input field 41 is a userid. After a user has entered in field 41 an acceptable input, he would then click on instruction key 42. The instruction key illustrated is submit a request. At this point the Web server captures the information entered by the user, as described in FIGURE 7. It will be appreciated that the Web server captures the information entered by the user, including specialized input, as well as any "hidden" default information, which can include password authorizations, charge account identification, and other information that can be used by the system in responding to the request. Thus the system can assume that the "hidden" password is an authorization to perform some function, such as include information from confidential source, or exit to the Internet. The charge authorization can also be tracked and accumulated by the system as it parses through its functions to charge back chargeable usages. If a request is for an order of an item, the actual item requested can be shipped and billed with this information. Since these functions are "hidden" they do not appear in the FIGURE but included with a request. The return of the request is illustrated in FIGURE 4.

FIGURE 4 is a sample result screen which illustrates how a sample report conforming to the request results are presented to the client after the requested service is provided by the computer system network in accordance with our invention formatted according to the specifications of a DIS capsule which is illustrated by example in FIGURE 9. In this example, the return was a file, whose file name is displayed as P81484 at 43. Informative text accompanying the file is included as illustrated by the example information 44 The screen provides the content of file 43 in the requested form of preformatted text 50 in the form of a display of a text report generated by a DIS capsule stored in the DIS server 17. While we show text as the form the report results, the form of the request can be another form of presentation, as and image, a voice response, or other multimedia presentation.Reports can be returned translated into any desired lan-

guage based upon the request, as may be provided by DIS capsule calls to a translator. These features are included in the result 50 report.

When the user selected image made by clicking on the image 32 in FIGURE 2, FIGURE 5 appears. FIGURE 5 is a next screen which illustrates how a request is made according to a users desires. A user makes a request, in this instance for sales results within the organization for YTD Catalog Revenue in accordance with our invention by entering text data into the data entry areas 41 and 42 of the formatted screen with information as to type of data selected 40A which will be translated into specific report information created by a DIS capsule.

FIGURE 6 is a sample result screen which illustrates how the request results are presented to the client after the requested service is provided by the computer system network in accordance with our invention formatted according to the specifications of a DIS capsule. In this instance selection of the object 32 links to the the screen of FIGURE 5, which in turn with the DIS capsule created the output shown in FIGURE 6. DIS Capsules will be illustrated by examples in FIGURE 9 and 10. In this example the output of the DIS capsule illustrated in FIGURE 10 is presented on the screen shown by FIGURE 6. The screen comprises a file name identifier, descriptive information 61, and preformatted text 60 which is the display of the named file P555119. This is the display of a graphic report showing what might be deemed (but is not) Confidential information relating to Catalog Revenue for 1995 YTD, with revenue given in $M, and breakout as to HDW, SFW, PMV, MN and MNT from selected locations in Chicago, New York, Dallas, and Los Angeles, all of which are located on different systems, and which, as illustrated in FIGURE 1, may be on different databases such as DB2, Oracle, and Sybase relational databases. This report was generated by a DIS capsule which is illustrated in FIGURE 9. This example illustrates how multiple actions can be taken on information retrieved. In this example data was translated into image material by calculation and formatting in the form of a graphic pie shaped report. Other image data could also be displayed, as frames of selected images, or a sequence of images in the form of a moving picture display, which can be outputted from a server as will be described in FIGURE 11.

FIGURE 7 illustrates a flowchart showing data flow between a web server and decision support system tool such as IBM's Data Interpretation System (DIS). FIGURE 7 shows the coupling of a Web client 71 (corresponding to Web browser 10 in FIGURE 1) to a Web server 72 (corresponding to Internet WWW server 11) and the coupling of a request to execute a DIS capsule.

The Web browser 71 can make a request to the Web Server 72 for a report through the use of HTML. The HTML document refers to our control program agent 73, which may be implemented with the C language or other language which can provide run code for the particular Web server which is employed. We illustrate our preferred program according to the description provided in FIGURE 8. The Web Server 72 passes request data to and invokes our control program 73 through the use of the CGI in accordance with our invention. The control program uses ODAS 74 in ODAS server 12 to set DIS capsule parameters and initiates the execution of a DIS capsule located in this embodiment in DIS capsule server 17 according to our preferred examples illustrated in FIGURES 9 and 10.

After a DIS capsule completes execution, the file created by the DIS capsule contains the formatted report results requested by the user. Our control program 73 dynamically creates the HTML statements that present the file to the Web browser 10 screen. Figure 7 shows the coupling within the Web server from ODAS 74 to a distributed DIS LAN 75 with heterogeneous connections to multiple databases DB2, Redbrick, Sybase and Oracle. Other sources of data can be linked to the LAN.

Preferred Embodiment Interface between Server and DIS

Our preferred control program agent 73 in FIGURES 1 and 11 is illustrated in detail by way of the flowchart of FIGURE 8. In our preferred embodiment, this program can be written in C or other suitable language but for general appreciation of the details, we will describe the steps in detail. These steps can be implemented by programmers of ordinary skill in the art without undue experimentation after understanding the steps described below. The control program agent 73 is located in a Web server and provides an interface and execution functions. Thus in FIGURE 11 the function is provided between the Web Server 131 (corresponding to Internet WWW server 11 in FIGURE 1) and DIS which is located in a DIS server 133 (corresponding to server 14 in FIGURE 1) and for presentation of results according to the instructions of the Web browser 130 (corresponding to browser 10 in FIGURE 1) according to the request command, which in default is a return to the Web browser home page. This interface utilizes in our preferred embodiment the Web Server CGI and the DIS ODAS.

Before we proceed to the control program 73, it will be noted that in FIGURE 11 the Web Browser 130 will link to a Web Server 131 accessing it on the Internet though a unique ID called the uniform resource locater to access the node which we call the Web server 131. When that access takes place an HTML document is displayed by the Web server 131 to the Web browser 130, as shown in FIGURE 2. Now, the user makes his entries as described with respect to FIGURE 2. Next the HTML document refers to the control program agent 73 and the Web server 131 through the use of the CGI invokes our control program agent 73. The Web server 131 retrieves data entered by the user from the HTML document and passes that data to our control program agent 73 upon invocation.

The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running

on it and to pass input parameters to the control program agent 73 (FIGURE 8) that were returned from the Hypertext document of the Web Browser. It will be appreciated that while weillustrate for our preferred example a single Web Server 131, the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired. Once the control program 73 (FIGURE 8) is invoked, the steps programmed for the machine to follow begins with a step 110 illustrated in FIGURE 8. In reviewing this preferred control program agent it should be appreciate that steps 110 and step 111 are steps that are interchangeable in order and which obtain environment variable data from the HTML document return.

Thus step 110 obtains a PATH_INFO environment variable data. PATH_INFO contains data from the HTML document that referred the Web Server to our program. Specifically the data contains the name of the DIS capsule to call, the name of the file containing the HTML statements to use when building the HTML document that displays the DIS capsule results to the Web browser, and the type of file that the DIS capsule will create. All off this information is the variable data which is stored in a buffer environment in step 112, and which is used in subsequent steps.

Thus also, the control program proceeds with step 111 which may follow or precede or proceed in parallel with step 110 to obtain the QUERY_STRING environment variable data. QUERY_STRING contains data from the HTML document that referred the Web Server to our program. Specifically the data contains values selected by the user and / or default values selected by the HTML document designer. These values are set in the DIS capsule by our control program prior to execution of the DIS capsule.This information is used to set variables in the DIS capsule. All off this information is the variable data which is stored in a buffer environment in step 112, and which is used in subsequent steps.

Within the scope of the discussion of the control program agent illustrated by FIGURE 8 it should be appreciate that the steps 112 through 125 include the utilization of an API set that provides a method of invoking executable programs located in a service machine which we denote as a sub-agent which executes in step 122 object capsules from our sub-agent DIS file server 14. This provides functions such as queue and update functions for databases on multiple platforms and allows the processing of data retried from a database to be performed, including executing calculations, doing formatting, charging of accounts and the storing of results as a file accessible to the control program agent.During processing our control program agent 73 provides setups for API calls which occurs in steps WHAT ARE THESE STEPS. Thus the control program agent will proceed as with an API set with step 113.

With the variable information now stored in a buffer, in step 113 the control program retrieves from a store, all of the DIS capsules that are used and the variable names associated with each DIS capsule and loads into memory associated with the control program the DIS capsule names available and the variable names associated with each DIS capsule.

At that point in step 114 the control program is ready to and does initialize a connection between our control program and the ODAS through the use of an ODAS API. In other environments another API performing similar functions could be used.

At that point, if required for control by the decision support system, and as required by DIS, the control program would log onto the port or desktop for the assigned user. Thus, our control program agent 73 in step 115 logs onto a DIS "desktop", our DIS file server 14.

Once the DIS capsule information is loaded into control program memory, the control program can and does in step 116 retrieve from its memory the DIS capsule variable names associated with the DIS capsule name passed to our control program in the step 110 where PATH_INFO is provided.

Next, in step 117 the control program creates a data array stored in the control program memory containing the DIS capsule variable names and the values for them that were passed to our control program in the QUERY_STRING step. These two steps 116 and 117 should be done in order, even though steps 110 and 111 can have an arbitrary order. At this point in step 117 you are matching the DIS capsule variable names with the data that was passed to the control program in the QUERY_STING environment variables.

Next, in preparation for a report, in step 118 the program creates a unique filename which may include data originated by the HTML document's variables stored in step 112 (dotted line) to pass to the DIS capsule as a DIS variable for use in naming the report which will be created by the DIS capsule. As a result, the DIS capsule will create that file with the unique file name during its process.

In anticipation of DIS capsule execution, the values of variables used by the DIS capsule are obtained from the data array in the control program memory containing the DIS capsule variable names and the values for them that were passed to our control program in the QUERY_STRING step. This is done in step 119 using the ODAS API to set the DIS capsule variable values. At this point the capsule server 17 for the DIS server 133 attached to the Web Server 131 via network 132 will have a DIS capsule services queue. This queue is the queue of jobs being requested of the Dis Capsule Server 17. For the current job request (other like requests being perhaps still in the queue) we use the ODAS API to query the contents of the DIS Capsule Services queue. If the queue size is larger (>t) than a threshold level, then the process enters a wait state until the queue size is reduced to a tolerable level. The queue test of step 120 is a loop test which returns to test the queue size until a test answering "is the queue of a size that execution can proceed?" (<t)

is answered "YES".

Whenever the queue test is answered YES, at that point the ODAS API is used to submit a DIS capsule for execution in step 121.

After the ODAS API submits a DIS capsule for execution the particular request process being executed by the control program enters a wait state until completion of the DIS capsule execution. For this step of the process the control program uses the ODAS API to wait for completion of the DIS capsule execution performed by the DIS capsule execution 122. During a wait state other requests can be processed by the control program, as requests are fed through the control program as a pipeline, in this WAIT PIPE API step 123, so that the control program continually advances requests through the system.

During the wait state 123 the ODAS API looks for a completion signal. When that is received, the control program then in step 124 reads the file identified by the name passed to the control program in the first PATH_INFO step that contains the HTML statements which are to be presented with the DIS report results.

While in step 124 the control program reads the file identified, it dynamically creates new HTML statements to display the preformatted text to the Web browser.The new HTML statement include the information retrieved from the file in step 113 so that it can be displayed as a header 44 accompanying the report to be displayed, along with the filename 43.

At this point, in step 125 the control program tests for the kind of report to be created by obtaining information from stored variables and identifies output parameters, such as whether the report is to be a text report, or a graphical report. At this point the control program branches to the sequence applicable to the kind of report to be created. If the output is to be routed the the Web server 10, then the output is routed to the Web server in step 126.

If a text file report is created by the DIS capsule, that determines that a text display is to be reported and the the control program reads the file created by the DIS capsule and dynamically creates HTML statements to display the data lines to the Web browser.

If a graphics file is created by the DIS capsule, that determines that a graphics display is to be reported and the the control program dynamically creates the HTML statement to display the graphics file to the Web browser.

On the other hand, the control program agent allows alternative output direction, and if the output is another type, or an additional output, as for broadcast, it can be routed to another destination. In step 127, we illustrate how using the IBM Digital Server, output can be routed to a requestor selected resulting output selected from a group of possible output units, comprising fax, printer, retail or banking installations, or provided as a series of full motion videos or still frames which are can be transmitted to display devices, such as a TV set under control of end users with a set-top box cable control. These facilities are provided by providing the output of our control program agent from the web server to the alternative output device 127, In this case, the IBM Digital Server, which with an RS/6000 CPU, Network I/F Bus, DISKs, modems, and X.25 Data Switch provides the hardware to route the output to a variety of output devices, to fax, printer, retail, banking, TV or cable customers via the digital server service machine for full motion and still video, supplied with MPEG 2 and MPEG 1 protocol images respectively, to subscribers, Along the way, the output can be coupled to an auxiliary function, such as back-up or accounting processes 128 which allow for charging for system utilization and service charges for services and items requested. These processes will make use of hidden variables associated with the request, such as charge authorization. One of the hidden variables which may be associated with a request is a credit card number. The credit card number, is preferably encrypted, with a DES or RSA encryption utility, and this along with access authorization variables, will allow access to sensitive databases which reside behind firewalls. If selected data according to the request is permitted to the access authorized user at the location inside or outside the Internet, the data can be included in the results reported by our system to the Web browser.

Preferred Embodiment of text DIS capsule

In accordance with our invention, an HTML document, which is running on a web server, refers to the control program agent. The web server then invokes the control program agent. The control program agent has a to command files, which provide the preferred file command objects in the form of DIS capsule objects, or DIS capsules as they are known. The command file contains a list of available DIS capsules. Accordingly, there is not need for the HTML document to know how to get to the command file, as the control program supplies this access. A capsule object, as a DIS capsule, can call other routines which may be written in well known programming languages such as Visual Basic or C. These routines become part of the capsule object by the reference, and these routines perform such functions as account tracking, compression, calculation, handling specific custom outputs such as video, voice, translation, and enable programmability of the capsule objects. The capsule objects also have standard object capability, and we will illustrate these by way of the specific examples described.

It will be seen that the control program 73, described in detail in FIGURE 8 acts in concert with DIS capsule execution. The DIS capsule is an object program with executable additions which we have created to interact with the control program. It should be also understood that the DIS capsule object can perform programmable functions on data which is retrieved from databases. Not only can a DIS capsule get data, it can combine, reformat, and update, the data

retrieved. It can act on the data to create new data and basically act as a dedicated processor processing data gathered or created during a Web browser request to output the end result to the user under programmable parameters determined by the creator of the DIS capsule, as they may be selected, if desired, by the user as part of the request. Thus the user entered inputs as part of his request, either free form or by selection of variables in the menus afforded to the user as illustrated by way of example in FIGURE 5.

DIS capsule objects are like some other objects. For instance in Microsoft's products, an example being the Excel (trademark of Microsoft) spreadsheet, one can click on an object portrayed on the screen and link a succession of objects to perform a specific function, such as take data from a spreadsheet and reformat it into a variety of selectable formats, such as text or graphic illustration. The kind of action to be taken is illustrated by an object on the screen, and linking of routines is done by a succession of clicks on icons representing the object.

In accordance with our preferred embodiment, a DIS capsule is used to invoke system resources. This is done by providing a list of commands, which may be those provided by a DIS processor itself, or written in Visual Basic or C by the programmer. The result is a command file, like an exec or command file in OS/2 or like a *.BAT file in DOS. These capsules perform the specific functions that are requested by the user from his initiation session. The user further qualifies the execution of the DIS capsule by providing parameters which are used in the invocation.

Now the DIS server 133 supports DIS, the program processor which supports DIS capsules by processing commands contained in the DIS capsule, either directly, in the case of DIS functions, or by to other system or user supplied functions. The user supplied functions comprise mainly those DIS functions which are supplied by DIS and illustrated in the manual "Developing Applications with OpenDIS Access Service, Version 2.0 of the OPEN Access Service." For those not familiar with command files, this manual is fully incorporated herein by this reference as available at the USPTO. An example of a system supplied function would be the base support for SQL queries of a specific database, which are invoked by the DIS capsule program.

In illustrating the specific examples of our invention illustrated in FIGURES 9 and 10, both illustrate linked objects according to a specified flow sequence within a DIS environment. The DIS environment contains numerous functions, including the Internetwork routing functions which the DIS capsules can invoke. Thus, a DIS object which queries a database, as illustrated, invokes the Internetwork routing functions to query databases where they are located on the network. If the preferred example of DIS environment is not supplied, a similar environment with program environment means which supports reaching a destination on the Internet by a link between systems which routes data from one physical unit to another according to the applicable protocol should be supplied. The protocol will employ a URL address for Internet locations.

FIGURE 9 illustrates by way of example a DIS capsule that creates a text report file. Referring to FIGURE 9, it will be seen that the capsule, represented by a series of linked objects, is supported by Internetwork processor support environment means 90. Within this environment an integrated capsule creates a text report file as a result of the object 95, make text. This object result file is the file 43 according to FIGURE 3 which is displayed at the browser. In the illustrated example, the multiple DIS capsule data retrieval command file 91(a)..91(n) initiates as a first step multiple queries to different databases which are specified by the parameters of the request. In this example, multiple queries are initiates as SQL type search requests as multiple steps 91(a)..91(n) executed by the DIS capsule server 133 with the Database Gateway 134 to select data from DB26000 databases located inside the intranet 140 and on the Internet by Internetwork routing to database gateway 134' and its DB26000 databases by step 91(a). The data is stored in a DIS declared buffer. Similarly, in parallel or successively, additional steps 91(b), 91(c), 91(d), and 91(n) retrieve data and store in their object buffer data retrieved from Sybase, Oracle, Redbrick, and IBM's Data Warehouse databases. Thus object 91(a) will query DB26000 and bring data back to DIS. Object 91(b) will query Oracle and bring data back to DIS. Object 91(c) will query Sybase and bring data back to DIS. Object 91(d) (shown as a dot in FIGURE 9) will query Redbrick and bring data back to DIS, and so on. The nth object 91(n) will query IBM's data warehouse and bring data back to DIS. In a subsequent linked processing step 92 data from the database queries in the first step is joined by joining object command file 92 and stored in a buffer related to this object. Object 92 will joint the data from the n locations searched in step 91. Thereafter, in a subsequent processing step performed by calculation object command file 93 on the joined data in the joined database result buffer of step 92, desired calculations performed in accordance with the parameters indicated by the request are done on the joined data. Thereafter, in accordance with the request parameters text is formatted to space delimited text by the format object command file 94. The results are stored in a buffer associated with format object command file 94. Thereafter, a make text command file 95 causes the formatted text to be created as a text file for the WWW server 131 to be stored in a file which is accessible to and can be retrieved and displayed by the control program agent 73, or directly displayed by the control program agent 73 in the form illustrated in FIGURE 4 at the Web browser 130. It will be noted we have illustrated this process as object capsules in a DIS internetworking environment. These object capsules are a specialized form of a command file, which can encompass additional commands called by an object.

Preferred Embodiment of graphics DIS capsule

FIGURE 10 illustrates by way of example a DIS capsule that creates a graphical report file. For simplicity, data in this FIGURE is also shown in a DIS environment 90. Retrieval object command file 101 illustrates a step of retrieval of data from one or more databases as specified in the parameters of the request, performing these retrieval steps as did retrieval object command files 91(a)..91(n). Thereafter, this data is plotted with the make plot object command file 102, with the results being stored in a buffer. The final step of creating a result- to-be-presented file, in this instance in the form of a bitmap ready for display to a Web browser 130 is created by the make bitmap (BMP) object command file 103. The example of a preferred bitmap object command which would be employed with todays Internet environment is a GIF image. Others can be used as well. Again the results are provided to the Web browser 130, by the action of the program command agent 73 on the Web Server 131, the results being illustrated by the pie-chart of FIGURE 6 in accordance with the parameters of the request for generating the graphical report illustrated by FIGURE 6.

Alternative Preferred Embodiments

Figure 11 illustrates an alternative configuration of the network system as it may be employed for permitting access to information available through homepages and in data warehouses where access to the homepage or database may or may not be restricted by a firewall. In Figure 11, the web browser(s) 130 accesses an associated Web Server 131, 131', 131"either by a coupling or addressing with a uniform resource locator (URL) the Web Server 131 which may be selected with a Hyperlink. This can be a direct coupling or an indirect coupling, as via a node locatable in a common access medium, such as provided by Internet resources accessible via a web browser, e.g. supporting Web Explorer, or Mosaic, NetScape, node 131 located somewhere on the Internet which utilizes our control program agent 73. Now node 131 which functions as a Web server is coupled via a token-ring network, SNA network, or other suitable network 132 (one of the any which may be used on the Internet as a transmission medium) with the facilities provided within what we will call our intranet, those facilities which are "proprietary" to the owner and which may be protected by firewalls at the intranet boundary 140. Now note that our control program 73 is resident within the Web Server 131 and functions as described in FIGURE 8 to couple to a DIS server 133 located within the intranet 140, which is preferably located behind a firewall as indicated in FIGURE 11. This DIS Server 133 is in turn coupled to our Database gateway 134. This database gateway is configured as illustrated also in FIGURE 1 for gathering information from databases coupled to it and located on servers for DB2, Oracle, Sybase, and Redbrick, as well as one for information warehouse functions. In our preferred embodiments these database units are IBM mainframe systems, as available commercially today, but they could be AS400s, RISC/6000, RISC/6000 SP or other systems supporting the databases.

The DIS Server is a server which supports DIS or similar decision support functions and the functions provided by our DIS capsules illustrated by FIGURE 9 and 10.

Now our Web browsers 130 can not only access information within the intranet, but can reach outside the intranet to gather information located elsewhere via the Internet. We will describe two examples of our preferred couplings to elements on the Internet. One example couples the database gateway 134 to another (a second) database gateway 134' via the Internet and it Internetwork routing (INR) protocol available from IBM as part of its current DIS product which can make use of UALs. The second database gateway 134' is coupled to its own (second) DIS server 133'. At this point the Web browser 130 can access data not only intranet, but also via the Internet to gather data from a database supported by DIS server 133' located outside the intranet. The Database server 134' would be able to gather information from any database coupled to it, as illustrated, assuming access is public or accessible after processing of a hidden variable access authorization.

However,the web browser(s) 130 can also access via Web Server 131 (with our control program 73 illustrated in detail in FIGURE 8) another Web server 131' which implements our control program 73. This Web server, for example, Web server 131' can also be coupled via its own (second) network 132' (which supports functions equivalent to network 132 and as illustrated in FIGURES 1 and 11) to an associated DIS Server 133' as illustrated to perform tasks like those we are describing from a request sent via the second network from its Web server 131'.

However, as another alternative example, Web server 131' with an appropriate API can access a directly coupled database available to the server, such as Microsoft's Access 131a. Thus small databases which have not yet invested in being able to gather information from an intranet resource, can use their own direct resources, and also be interrogated by the Web browser(s) 130, or another web browser 136. Remember that browser's 130 can also communicate with the Web server 131' across the Internet, just as can a Web browser 136 located on the intranet 140 inside of the firewall illustrated by the intranet 140 dashed line shown in FIGURE 11. With a browser 136 in place at the Web Server 131' location, that browser 136 can make requests, if authorized across the intranet to the Web Server 131 which can then utilize the DIS capsules provided by the DIS Server 133.

Physically, the network 132 will have its own access server 135 preferably in the form of a TCP/IP server 135 to make the physical connection across the Internet. We illustrate in FIGURE 11 this other logical layer as located on the network. This TCP/IP server supports the physical connections which are needed by the other logical higher levels of

service supported on the network. The use of an InterNetwork Routing Protocol (INR) allows the logical coupling illustrated between a application processing server 134 to an external intranet application processing server 134'. On each network there can be one or more web servers. A Hypertext document request asking for a field to be seached, as by a Hyperlink, could index to a server directly, e.g. a second web server 134" on the same network which would have its own control program agent function duplicating the control program agent resident in web server 134. Thus at the request homepage a menu which say if "Art&Literature search", when selected in a Hyperlink setting, would index to a particular web server and a particular document within that web server's environment. This web server 134" besides being linked to its own application processing server 133" has a direct link, in the environment illustrated, to an MVS CICS, a transaction processing server for handling transaction processing. Such a solution allows CICS transaction processing to utilize the Internet to save transmission costs and still be located beneath a firewall for retention of data integrity. The outputs provided by the web server to the requested destination can be outside of the firewall, and in the form of results illustrated by the possible examples shown in FIGURES 3, 5 and 8.

While we have described our preferred embodiments of our invention, it will be understood that those skilled in the art, both now and in the future, may make various improvements and enhancements which fall within the scope of the claims which follow. These claims should be construed to maintain the proper protection for the invention first disclosed.

The following features - for themselves or in combination with other features - are also characteristics of the invention:

- The web browser system further comprises at least one intranet network supporting a plurality of web servers having a control program agent and having an access agent coupled to the Internet;
  at least one web server coupled to said web browser client for receiving and fulfilling a request as an agent of said client; and
  at least one intranet network supporting said web server and having an access agent coupled to the Internet;

- The web browser system further comprises a plurality of information access servers for acting as sub-agents for said web server during a process of fulfilling requests, said access servers having capsule objects which perform programmable functions which are executable upon a received command from said web server, at least one of said access servers being located on said intranet network; said web server including a control program agent for receiving a user request initiated at the web browser client for information and for transmitting said request to a sub-agent access server having capsule objects which execute upon command programmable functions requested by said web server.

Claims

1. A web browser system, comprising:
   a web browser, means for associating said web browser with a homepage by a coupling or addressing with a uniform resource locator,
   a control program agent node located somewhere on the Internet supporting a control program agent coupled to and supporting said homepage by a coupling or addressing with a uniform resource locator,
   said control program agent node being coupled via a network with facilities provided within an intranet for private owner facilities and which may be protected by firewalls at the intranet boundary, said control program agent being coupled to a command file server and said command file server being coupled to a database gateway for gathering information from databases coupled to said database gateway and located on different database servers, said command file server supporting a plurality of command file objects which are programmed to perform web browser service support functions at the request of a user of said web browser to access information within the intranet and to gather information located elsewhere via the Internet as a sub-agent of said control program agent.

2. A web browser system according to claim 1,
   wherein by submission of a request at a web browser a user can not only access information within an intranet, but can reach outside the intranet to gather information located elsewhere via the Internet.

3. A web browser system according to claim 1,
   wherein there are on the networking including an intranet and the Internet a plurality of database gateways, and at the command of a command file running within a command file server one database gateway is coupled to another database gateway via the network by an inter-network routing protocol.

4. A web browser system according to claim 1,
   wherein there are on the networking including an intranet and the Internet a plurality of database gateways, and at the command of a command file running within a command file server one database gateway is coupled to another

database gateway via the network by an inter-network routing protocol invoking coupling of database gateways by UALs.

5. A web browser system according to claim 1, wherein said second database gateway is coupled to its own command file server.

6. A web browser system according to claim 1, wherein a web browser initiated request is distributed via an intranet to the Intranet whereby access of data is obtained not only intranet, but also via the Internet to gather data from a database supported by a command file server located outside the intranet.

7. A web browser system according to claim 1, wherein a web browser initiated request is distributed via an intranet to the Internet whereby access of data is obtained not only internet, but also via the Internet to gather data from a database supported by a command file server located outside the internet and coupled to said command file server with public access or access obtained after processing of variable access authorization data provided thorugh said command file server.

8. A web browser system, comprising:
a web browser, means for associating said web browser with a homepage by a coupling or addressing with a uniform resource locator including a first control program agent node located somewhere on the Internet supporting a control program agent coupled to and supporting said homepage by a coupling or addressing with a uniform resource locator,
said second control program agent node being coupled via a network with facilities provided within an intranet for private owner facilities and which may be protected by firewalls at the intranet boundary,
a second control program agent node located somewhere on the Internet supporting a second control program agent by a coupling or addressing with a uniform resource locator,
said second control program agent node being coupled via network with facilities provided within an intranet for private owner facilities and which may be protected by firewalls at the intranet boundary,
said first control program agent being coupled to said second control program agent node located somewhere on the Internet supporting said second control program agent and coupled to and supporting a a command file server,
said command file server being coupled to a database gateway for gathering information from databases coupled to said database gateway and located on different database servers, said command file server supporting a plurality of command file objects which are programmed to perform web browser service support functions at the request of a user of said web browser to access information within the intranet and to gather information located elsewhere via the Internet as a sub-agent of said control program agent.

9. A web browser system according to claim 8, wherein said first control program agent resides on a first web server supporting said web browser and said second control program agent resides on a second web server which is coupled via its own network to an associated command file server to perform tasks requested by said web browser and communicated to said web browser after passing through multiple networks.

10. A web browser system, comprising:
a web browser, means for associating said web browser with a homepage including a first control program agent node supporting a control program agent coupled to and supporting said homepage and supporting an API to access a database available to said first control program agent node,
said control program agent and API enabling a user of said web browser to gather information from said database available to said first control program agent node and to gather information from an intranet resource and to provide access thereto in response to an interrogation initiated at a remote web browser.

11. A web browser system according to claim 10, wherein said remote web browser is also coupled to a second control program agent node located on the Internet, said second control program agent node supporting a second control program agent supporting an API to access a database available to said first control program agent node via said second control program agent said second control program agent and API enabling a user of said web browser to gather information from a database available to said first control program agent node via said second control program agent node and to gather information from an intranet resource and to provide access thereto in response to an interrogation initiated at said web browser. across the Internet by a coupling or addressing with a uniform resource locator to said second control agent node and from resources available on an intranet coupled to said second control program agent node.

12. A web browser system according to claim 10,
wherein said second control program agent node is coupled via a network with facilities provided within an intranet for private owner facilities and which may be protected by firewalls at the intranet boundary,
said second control program agent node located somewhere on the Internet supporting said second control program agent by a coupling or addressing with a uniform resource locator,
said first control program agent being coupled to said second control program agent node located somewhere on the Internet supporting said second control program agent and coupled to and supporting a a command file server,
said command file server being coupled to a database gateway for gathering information from databases coupled to said database gateway and located on different database servers, said command file server supporting a plurality of command file objects which are programmed to perform web browser service support functions at the request of a user of said web browswer to access information within the intranet and to gather information located elsewhere via the Internet as a sub-agent of said control program agent.

13. A web browser system according to claim 10,
wherein said web browser is at a web server location with said web server providing said control program agent node,
and browser requests, if authorized for access across said intranet, accesses a command file agent in a web server on said intranet providing said second command file agent node, which then utilize DIS capsules provided by a DIS Server functioning as a command file server.

14. A web browser system according to claim 10,
wherein the intranet network is provided with an access server to make physical connections across the Internet.

15. A web browser system according to claim 10,
wherein said DIS server invokes an InterNetwork Routing Protocol (INR) to create a logical coupling between an a application processing server to an external intranet application processing server.

16. A web browser system according to claim 10,
wherein a web browser request is a Hypertext document request asking for a field to be searched and indexing with a Hyperlink a web server providing a command file agent node on the same network which would have its own program control agent function duplicating the control program agent resident in said control program agent node.

17. A web browser system according to claim 10,
a web browser request homepage a menu topic, when selected in a Hyperlink setting, indexes index to a particular second control program agent node and a particular document within that second control program agent's environment.

18. A web browser system according to claim 10,
wherein said second control program agent node, besides being linked to its own application processing server has a direct link, to a a transaction processing server for handling transaction processing.

19. A web browser system according to claim 10,
wherein said transaction processing enables use of the Internet for authorized access and transmission of data, but said transaction processing server is located beneath a firewall for retention of data integrity.

20. A web browser system according to claim 10,
wherein said transaction processing server's output is provided to said second control program agent node server and from said second control program agent node server to a requested destination on the internet specified by the request initiated by said web browser.

21. A web server system for supporting a web browser, comprising
means for receiving from a world wide web browser a request to be fulfilled as an agent of the browser client,
a control program agent for organizing organizing distributed sub-agents as distributed integration solution servers on an intranet network supporting the web server which also has an access agent servers accessible over the Internet.

22. A web server system according to claim 21, further comprising, a plurality of distributed integration solution servers for executing selected capsule objects which perform programmable functions upon a received command from said web server control program agent.

23. A web server system according to claim 22, further comprising, a database gateway coupled to a plurality of database resources for supplying upon a singlr request made from a Hypertext document, requested information from multiple data bases located at different types of databases geographically dispersed.

24. A web server system according to claim 23, further comprising, command objects for performing calculations, formatting, and other services prior to reporting to the web browser or to other locations, in a selected format a requested result report selected from a set of result reports, including a display report, facsimile report, a printer report, a report to customer installations, and a report to TV video subscribers, with account tracking.

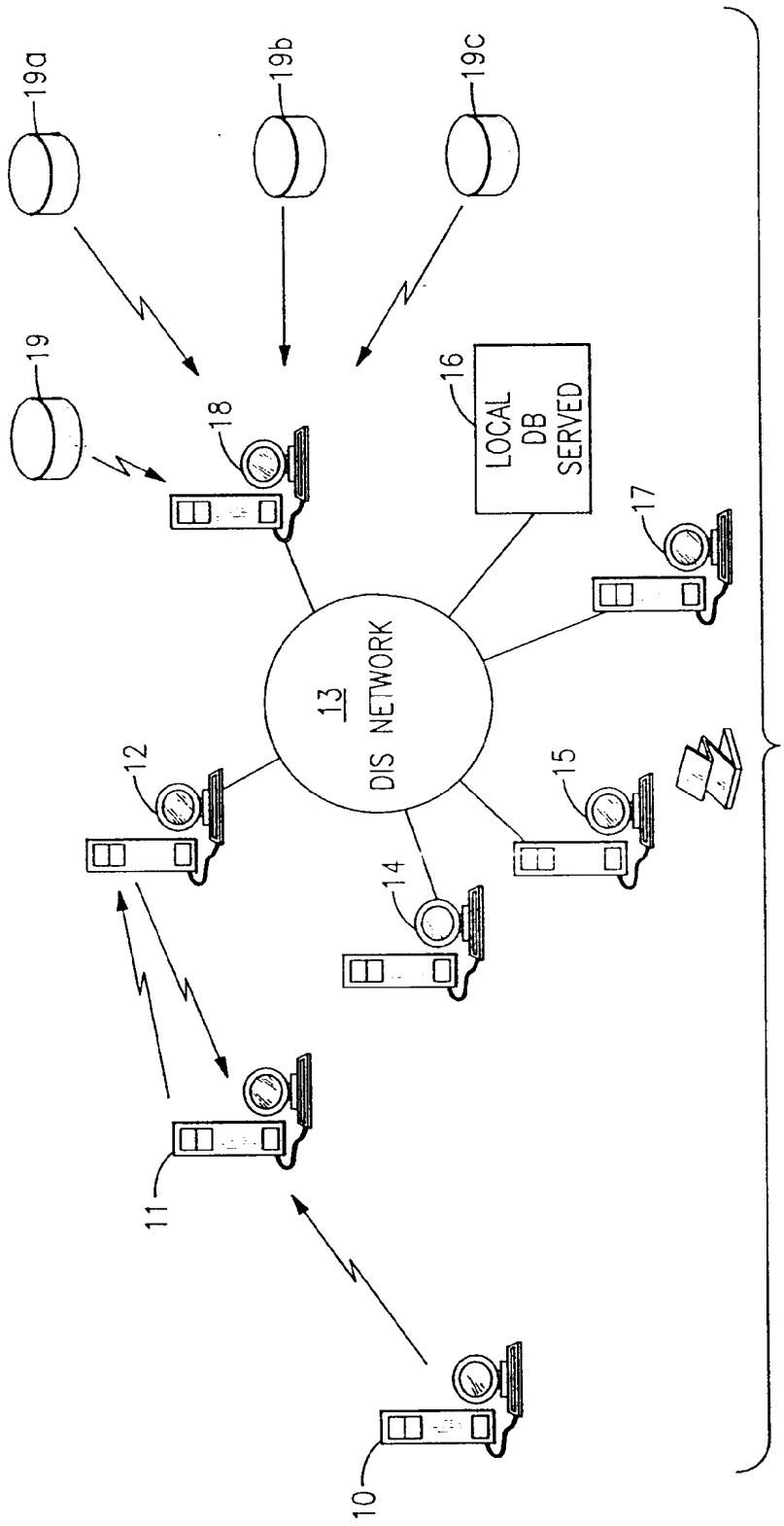


FIG.1

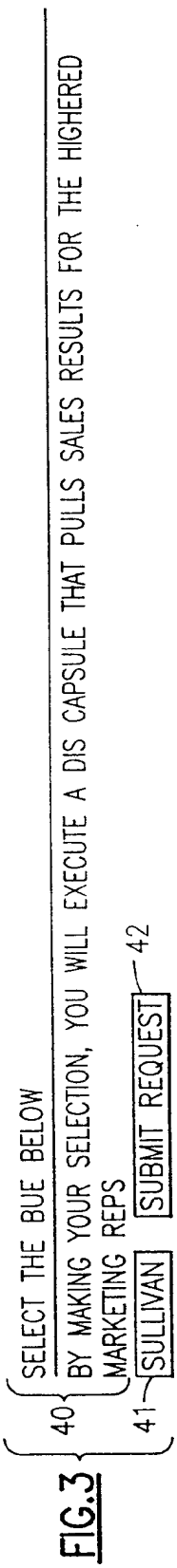


FIG.3

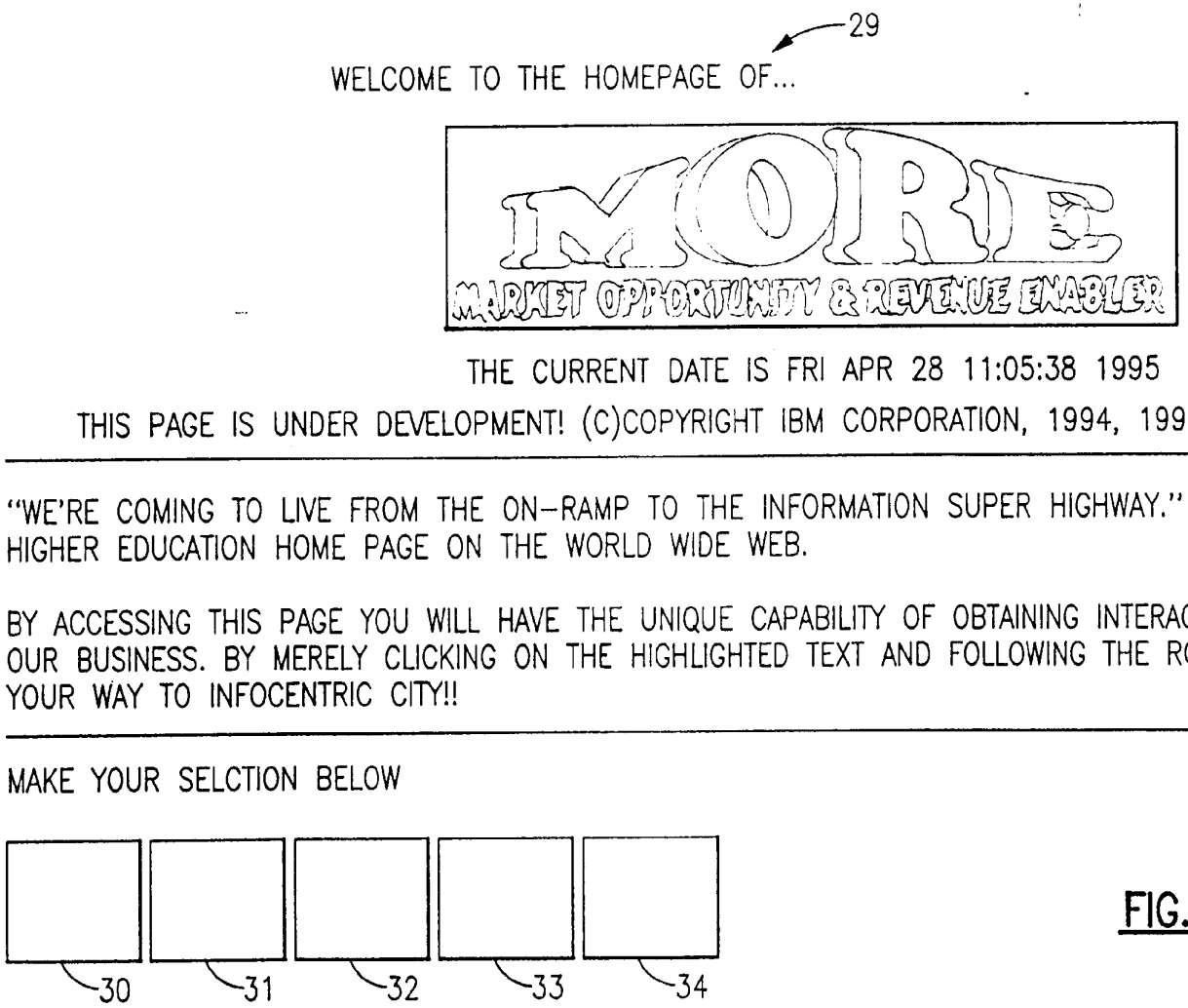

29
WELCOME TO THE HOMEPAGE OF...
MORE
MARKET OPPORTUNITY & REVENUE ENABLER
THE CURRENT DATE IS FRI APR 28 11:05:38 1995
THIS PAGE IS UNDER DEVELOPMENT! (C)COPYRIGHT IBM CORPORATION, 1994, 1995. ALL RIGHTS RESERVED.
"WE'RE COMING TO LIVE FROM THE ON-RAMP TO THE INFORMATION SUPER HIGHWAY." THIS IS THE IBM U.S. HIGHER EDUCATION HOME PAGE ON THE WORLD WIDE WEB.
BY ACCESSING THIS PAGE YOU WILL HAVE THE UNIQUE CAPABILITY OF OBTAINING INTERACTIVE INFORMATION RELATING TO OUR BUSINESS. BY MERELY CLICKING ON THE HIGHLIGHTED TEXT AND FOLLOWING THE ROAD-SIGNS YOU'LL BE WELL ON YOUR WAY TO INFOCENTRIC CITY!!
MAKE YOUR SELCTION BELOW
30
31
32
33
34

FIG.2

FILE NAME IS P81484. ←43

CATCH IT ON THE NET!

44 YOU JUST RAN A DATA INTERPRETATION SYSTEM CAPSULE!
THE OUTPUT IS THE TEXT TOOL TRANSFERRED AS A TEXT.

CUSTOMERS WHO MIGHT WANT TO KNOW ABOUT THE
DIS WWW GATEWAY---THIS DATA IS FROM THE IBM MARKET PLANNING DATA SYSTEM--
A DB2 MVS DATABASE

CUSTOMER NAME	SW DESCRIPTION	INST	CITY STATE
AC NIELSEN CO	DIS 1.3.6 DIS ENTERPRISE SE	9501	CHERRY HILL NJ
AC NIELSEN CO	DIS 1.3.6 DIS ENTERPRISE SE	9501	GREEN BAY WI
AC NIELSEN CO	DIS 1.3.6 DIS ENTERPRISE SE	9501	MINNEAPOLIS MN
AC NIELSEN CO	DIS 1.3.6 DIS ENTERPRISE SE	9501	WILTON CT
ADVANTIS	DIS 1.3.6 DIS ENTERPRISE SE	9501	SCHAUMBURG IL
ALTA BATES MEDI	DIS 1.3.6 DIS ENTERPRISE SE	9410	BERKELEY CA
ALTA BATES MEDI	DIS 1.3.6 DIS ENTERPRISE SE	9503	BERKELEY CA
AMERICAN PRESID	DIS 1.3.6 DIS ENTERPRISE SE	9501	OAKLAND CA
ANHEUSER BUSCH	DIS 1.3.6 DIS ENTERPRISE SE	9501	ST LOUIS MO
ANHEUSER BUSCH	DIS 2.0 OPENDIS ACCESS SERV	9501	ST LOUIS MO
50 ASHLAND OIL INC	DIS 1.3.6 DIS ENTERPRISE SE	9502	LEXINGTON KY
ASHLAND OIL INC	DIS 2.0 OPENDIS ACCESS SERV	9502	LEXINGTON KY
BELLSOUTH CELLU	DIS 1.3.6 DIS ENTERPRISE SE	9501	FT LAUDERDALE FL
BELLSOUTH CELLU	DIS 2.0 OPENDIS ACCESS SERV	9501	FT LAUDERDALE FL
BELLSOUTH COMMU	DIS 1.3.6 DIS ENTERPRISE SE	9501	ATLANTA GA
BELLSOUTH COMMU	DIS 1.3.6 DIS ENTERPRISE SE	9501	BIRMINGHAM AL
BELLSOUTH TELEC	DIS 1.3.6 DIS ENTERPRISE SE	9501	ATLANTA GA
BRIO TECHNOLOGY	DIS 1.3.6 DIS ENTERPRISE SE	9501	MOUNTAIN VIEW CA
BRISTOL MYERS S	DIS 1.3.6 DIS ENTERPRISE SE	9410	PLAINSBORO NJ
BRISTOL MYERS S	DIS 2.0 OPENDIS ACCESS SERV	9502	NEW YORK NY
BROADWAY DEPT S	DIS 1.3.6 DIS ENTERPRISE SE	9410	LOS ANGELES CA
BROOLYN UNION	DIS 1.3.6 DIS ENTERPRISE SE	9410	BROOKLYN NY
CHESEBROUGH PON	DIS 1.3.6 DIS ENTERPRISE SE	9501	GREENWICH CT
COLGATE-PALMOLI	DIS 1.3.6 DIS ENTERPRISE SE	9410	IRVINE CA
COLGATE-PALMOLI	DIS 1.3.6 DIS ENTERPRISE SE	9410	MORRISTOWN NJ
COLGATE-PALMOLI	DIS 1.3.6 DIS ENTERPRISE SE	9410	NEW YORK NY

FIG.4

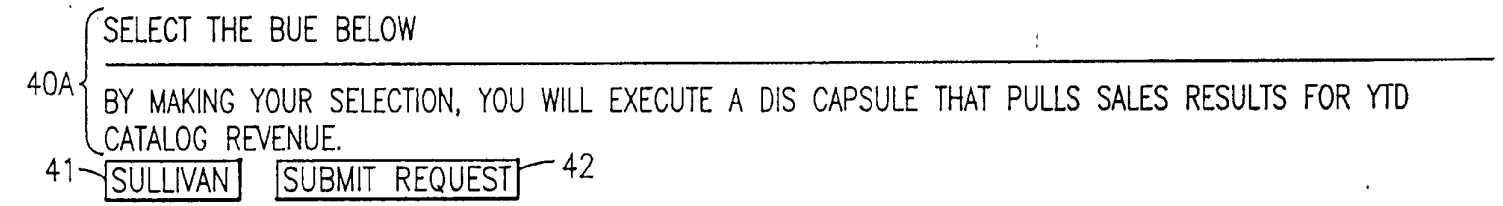


FIG.5

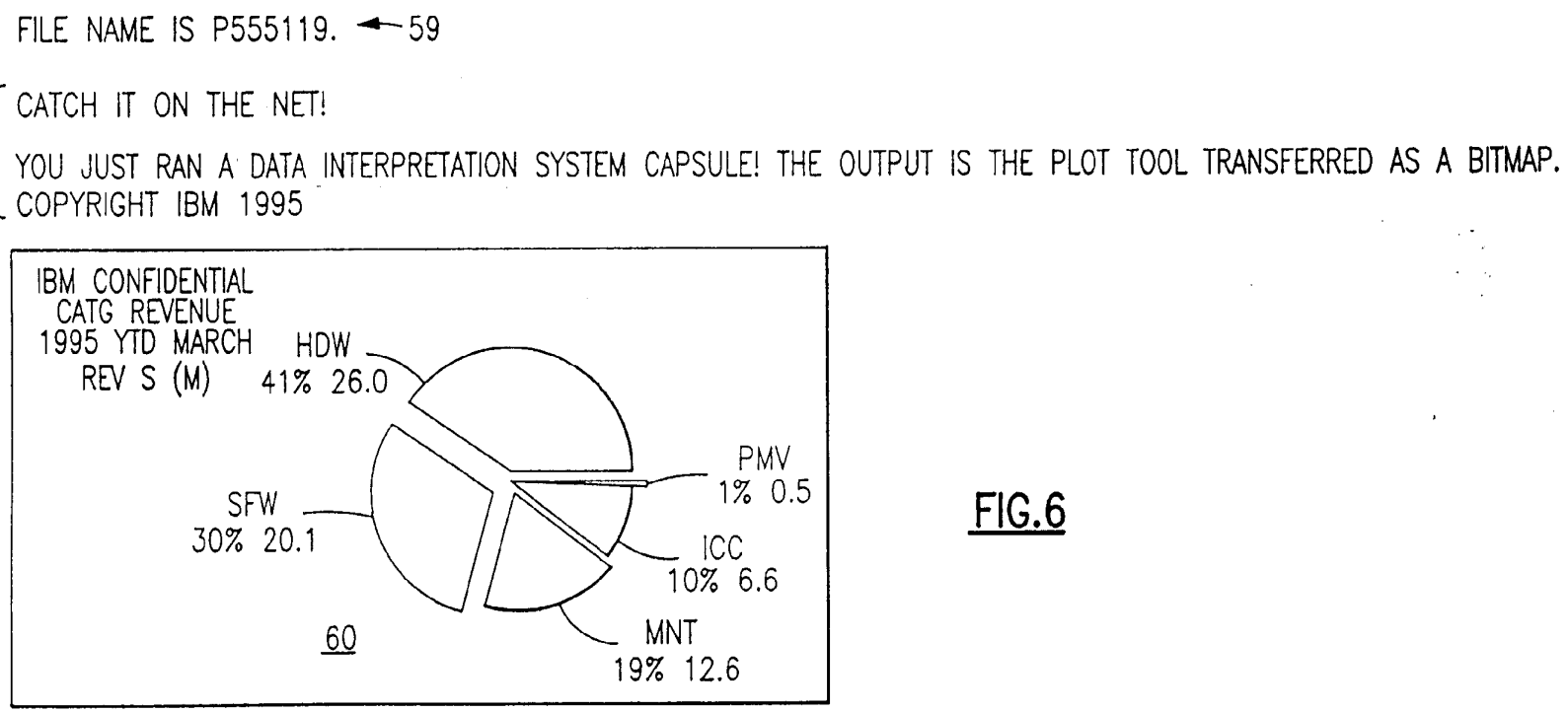


FIG.6

EP 0 747 842 A1

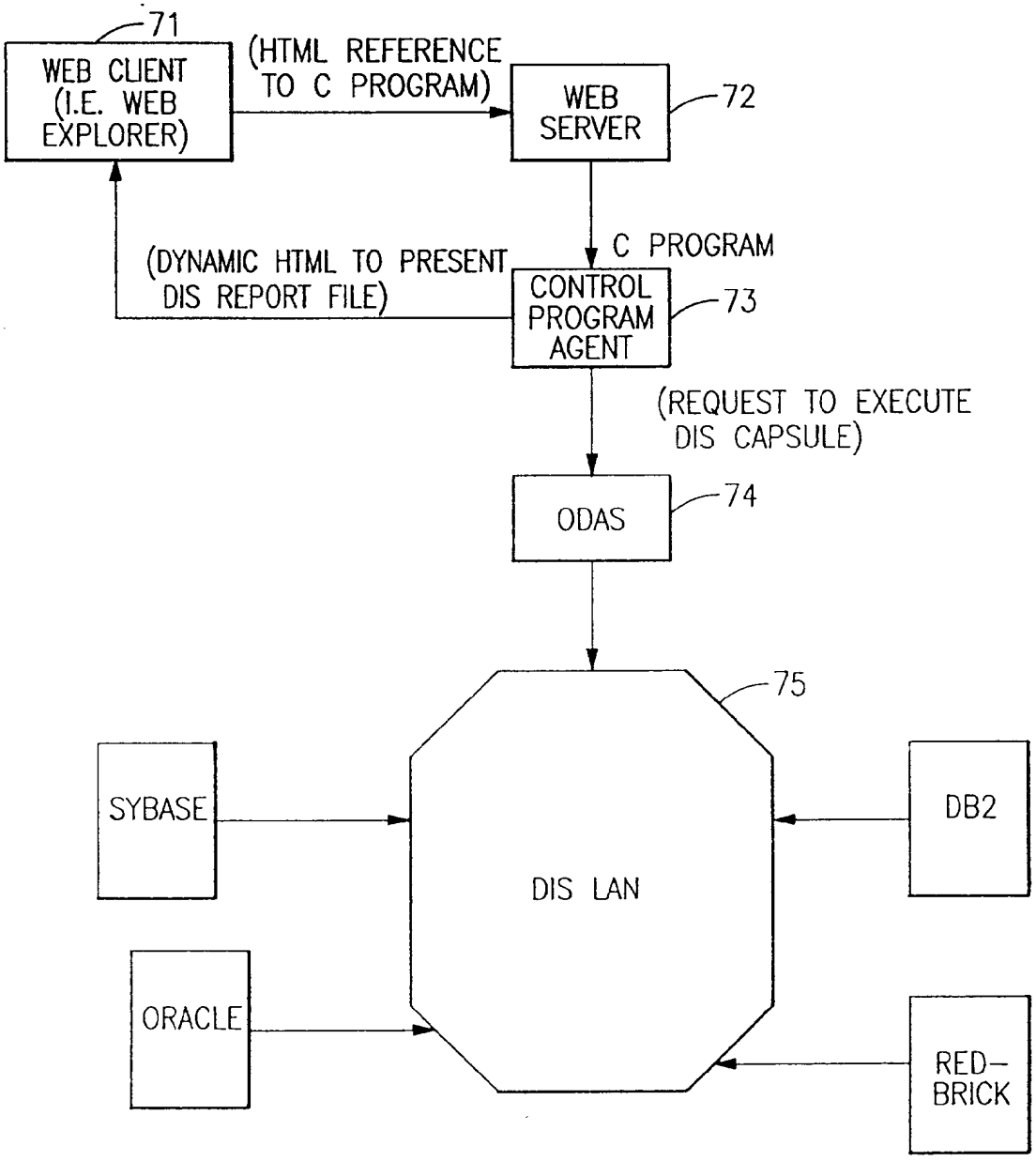


FIG.7

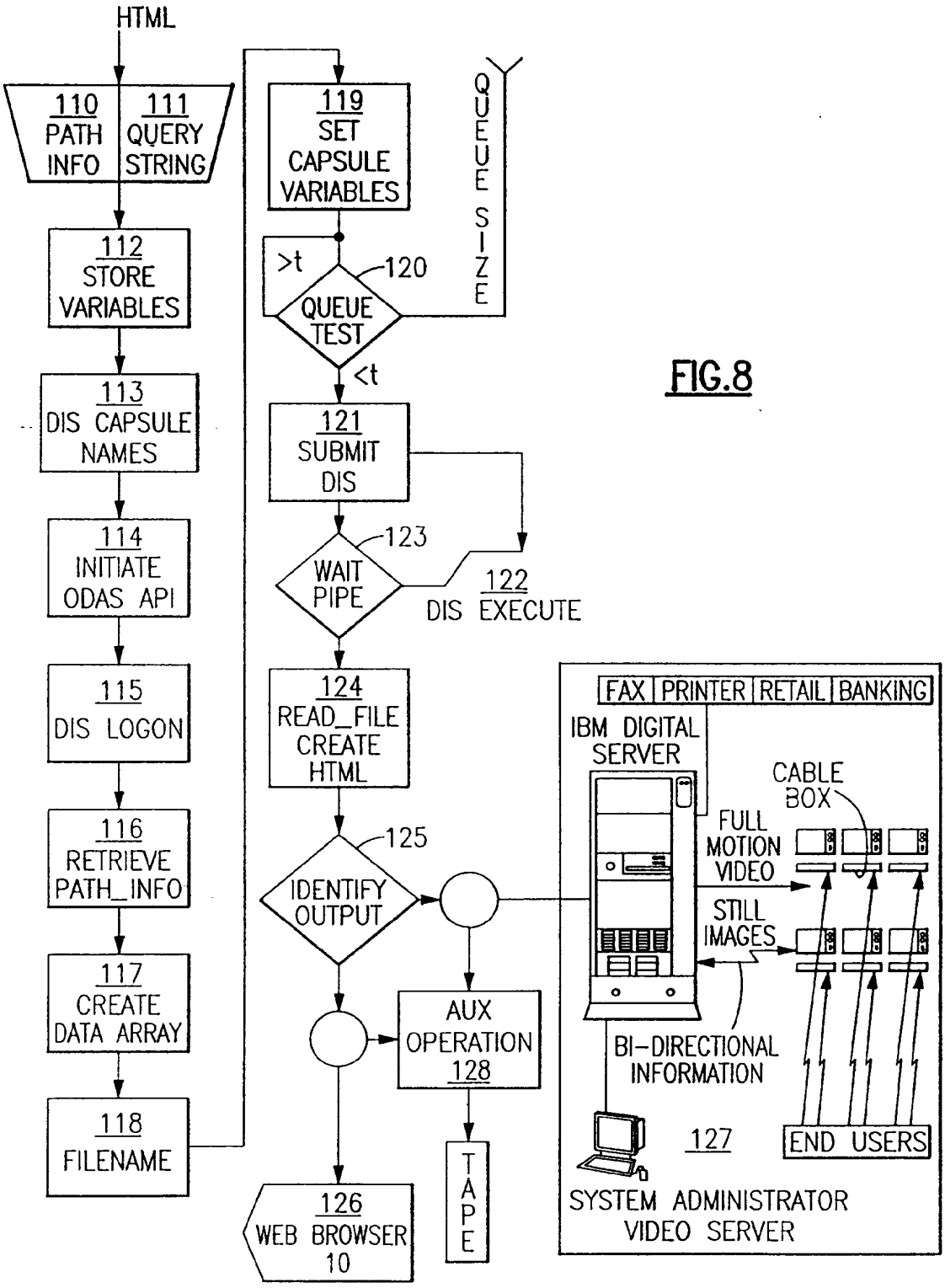


FIG.8

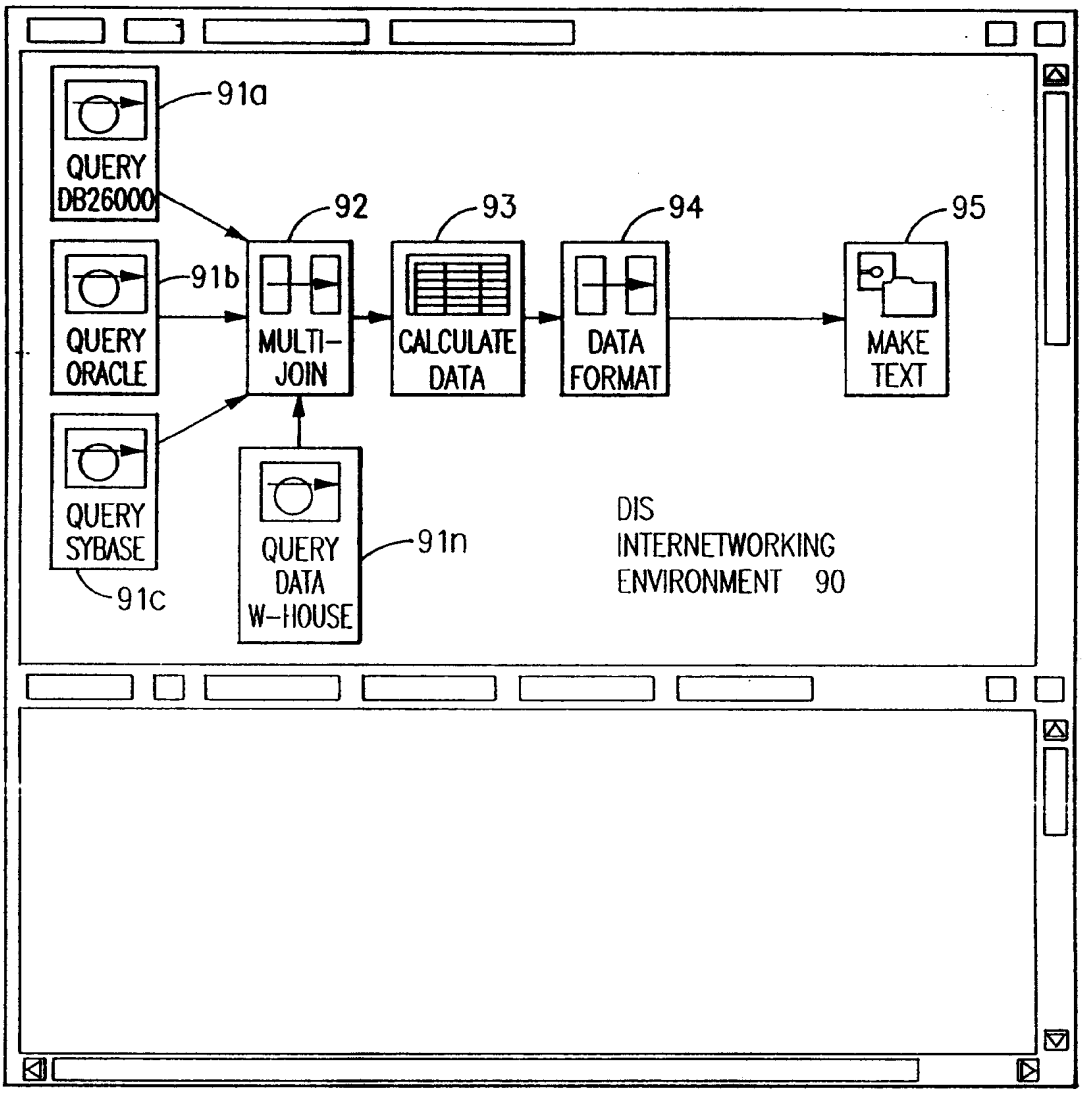


FIG.9

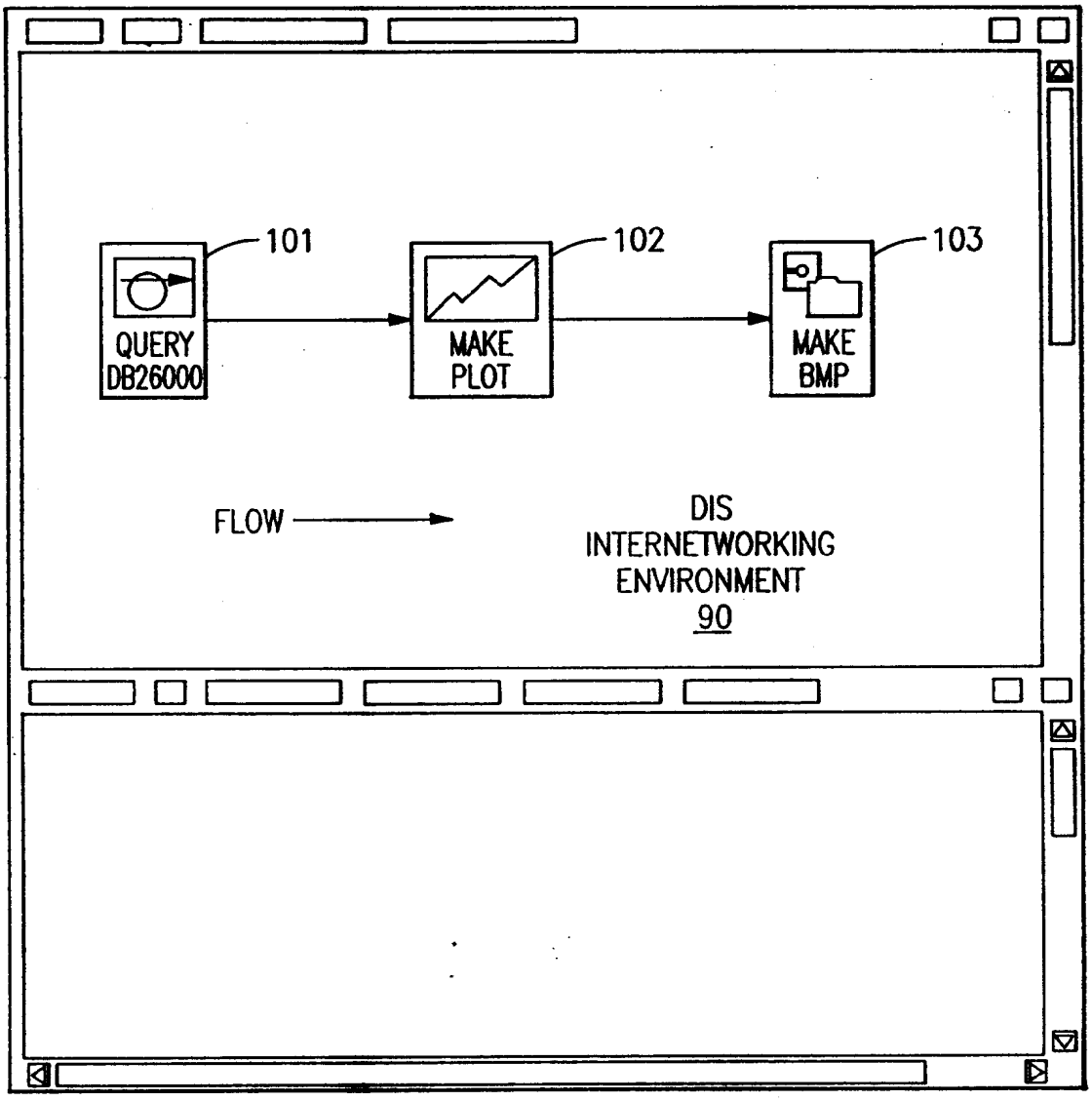


FIG.10

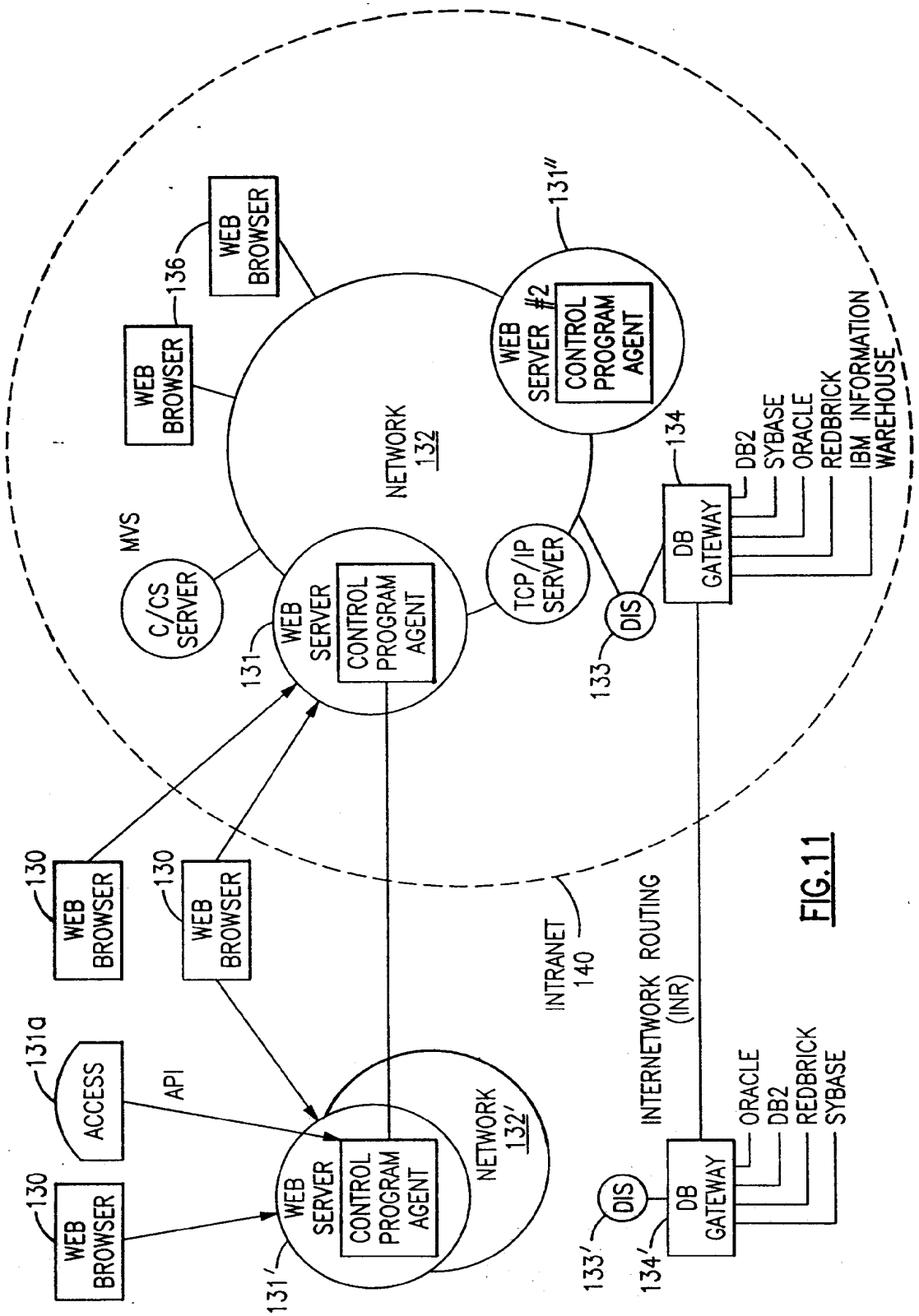


FIG.11

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 96 10 8976

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	SCALING NEW HEIGHTS IN TECHNICAL COMMUNICATION, BANFF, SEPT. 28 - OCT. 1, 1994, 28 September 1994, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 192-197, XP000510688 LAU T: "BUILDING A HYPERMEDIA INFORMATION SYSTEM ON THE INTERNET" * abstract * * page 192, line 1 - page 193, line 3 * ---	1-24	G06F17/30
D,A	US-A-4 274 139 (HODGKINSON SUSAN D ET AL) 16 June 1981 * abstract * * column 1, line 1 - column 3, line 18 * -----	1	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	11 September 1996	Katerbau, R

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 782 085 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
02.07.1997 Bulletin 1997/27

(51) Int. Cl.[6]: **G06F 17/30**

(21) Application number: **96308817.4**

(22) Date of filing: **05.12.1996**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **28.12.1995 US 581300**

(71) Applicant: **International Business Machines Corporation**
Armonk, N.Y. 10504 (US)

(72) Inventor: **Steele, David Aaron**
Cupertino, California 95014 (US)

(74) Representative: **Ling, Christopher John**
IBM United Kingdom Limited,
intellectual Property Department,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Video browsing on the world wide web

(57) A system and method are provided for supporting video browsing over a communication network such as the Internet/World Wide Web. A graphical user interface is provided through a client software tool such as a Web browser. A client/user selects a video data object stored at a remote server. A set of points (52) within the object are displayed at the client's graphical user interface display, as representations, preferably thumbnail images, of the points within the object. The user selects an interval (56) defined by the representations, preferably the interval between two temporally adjacent representations. Responsive to this selection, a new set of points, falling within the selected interval, are chosen, and representations of those points are generated and displayed. By doing so repeatedly, the user can easily browse through the video data object, and quickly and easily zero in on a desired portion of the video data object.

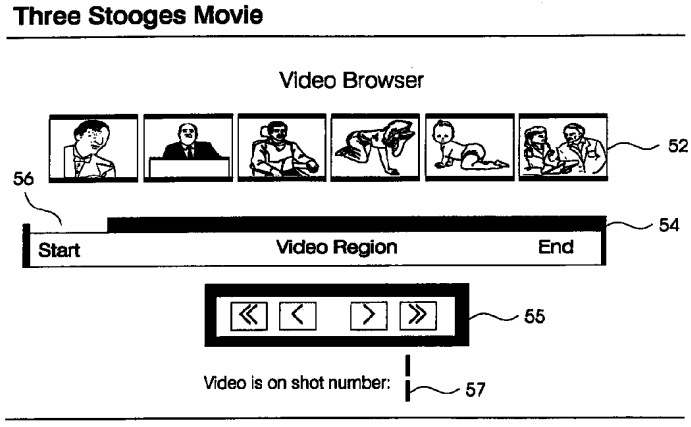


FIG. 7

EP 0 782 085 A1

Description

Field of the Invention

The invention generally relates to the fields of computer systems and multimedia communications, and more particularly relates to the fields of video storage and compression, and of interactive video playback and browsing.

Glossary of Terms Used

While dictionary meanings are also implied by certain terms used here, the following glossary of some terms may be useful.

Internet ("the Net"): The connection system that links computers worldwide in a network.
TCP/IP: Transmission Control Protocol/Internet Protocol. A packet switching scheme the Internet uses to chop, route, and reconstruct the data it handles, from e-mail to video.
World Wide Web (WWW, "the Web") : The Internet's application that lets people seeking information on the Internet switch from server to server and database to database by clicking on highlighted words or phrases of interest. An Internet Web server supports clients and provides information.
Home page: A multi-media table of contents that guides a Web user to stored information on the Internet.
Server: A machine (computer) which performs a task at the command of another machine ("client"). In the context of the present invention, a server's primary function is to facilitate distribution of stored information over the Web.
Client: A machine which provides commands to a server, and is serviced by the server. Typically, a client machine is operated by an end user, and functions responsive to user commands.
Web Browser: A program running on a user-operated client computer. When a user "surfs" the Web using a browser, the browser acts as an Internet tour guide, allowing the client machine to display pictorial desktops, directories and search tools supported by the server.
URL: Universal Resource Locater, a Web document version of an e-mail address, in character string form, which uniquely identifies a document, application, or tool available over the Web.
Hyperlink: A network addressing tool embedded in a user-understandable displayed and/or highlighted item, such as a word, phrase, icon or picture. A URL can be accessed by means of its corresponding Hvperlink. When a user on a client machine selects the highlighted hvperlink through the user interface, the underlying item is then retrieved to the client supporting a Web browser.
HTTP Hypertext transfer protocol: Hypertext transfer protocol. The character string "http:" at the beginning of a URL indicates that the document or file designated by the URL contains hyperlinks defined according to the HTTP.
HyperText Markup Language (HTML): HTML is the language used by Web servers to create and connect documents that are viewed by Web clients. HTML uses Hypertext documents. Other uses of Hypertext documents are described in the following U.S. Patents:

Bernstein et al., 5,204,947, issued April 20, 1993;
Bernstein et al., 5,297,249, issued March 22, 1994; and
Lewis, 5,355,472, issued October 11, 1994;

all of which are assigned to International Business Machines Corporation, and which are referenced herein.

Background Art

In recent years, the technologies of video data compression, storage, and interactive accessing have converged with network communications technologies, to present exciting prospects for users who seek access to remotely stored multimedia information.

In the area of network communications technologies, particluarly exciting has been the recent prominence of the Internet and its progeny, the World Wide Web. The Internet and the Web have captured the public imagination as the so-called "information superhighway." Accessing information through the Web has become known by the metaphorical term "surfing the Web."

The Internet is not a single network, nor does it have any single owner or controller. Rather, the Internet is an unruly network of networks, a confederation of many different networks, public and private, big and small, whose human operators have agreed to connect to one another. The composite network represented by these networks relies on no single transmission medium. Bi-directional communication can occur via satellite links, fiber-optic trunk lines, phone lines, cable TV wires and local radio links.

To this point the World Wide Web (Web) provided by the Internet has been used in industry predominately as a

means of communication, advertisement, and placement of orders. The World Wide Web facilitates user access to information resources by letting people jump from one server to another simply by selecting a highlighted word, picture or icon (a program object representation) about which they want more information, a manoeuvre known as a "hyperlink". In order to explore the WWW today, the user loads a special navigation program, called a "Web browser" onto his computer.

There are a number of browsers presently in existence and in use. Common examples are NetScape, Mosaic and IBM's Web Explorer. Browsers allow a user of a client to access servers located throughout the world for information which is stored therein. The information is then provided to the client by the server by sending files or data packets to the requesting client from the server's storage resources.

Part of the functionality of a browser is to provide image or video data. Web still image or video information can be provided, through a suitably designed Web page or interface, to a user on a client machine. Still images can also be used as Hypertext-type links, selectable by the user, for invoking other functions. For instance, a user may run a video clip by selecting a still image.

However, video data objects are very large, or, to put it more precisely, the quantity of data per unit time in a real-time viewing of a video data object is large. As a consequence, access by a user to a desired video data object is subject to data throughput constraints. The present state of the art makes it impracticable to provide more than a few tens of seconds of real-time video over the Internet with a response time that will be satisfactory to a user.

Therefore, multimedia and communication systems for providing users with access to video data objects, for browsing, searching, etc., must grapple with the problem of providing video data in a manner which best utilizes the available throughput to provide video data in a form which is most useful to the user.

With this design objective in mind, let us now consider the state of the art in the technologies of video data compression, storage, and interactive accessing. Recent work has been done to make video material more available and usable over the Web. For instance, an article in the August 1995 issue of ADVANCED IMAGING, by Amy T. Incremona, titled "Automatically Transcribing and Condensing Video: New Technology is Born", describes a method for providing video having an accompanying textual index, such as audio narration or closed caption text. Still images are presented, along with a transcription of audio text that accompanies the images (illustration on page 60). This information is provided in HTML format. Thus, a user can take advantage of the temporal correspondence between video shots and narration or closed caption text. To find a desired point in the video corresponding with a known point in the text, the user performs a key word search for the known point in the text. The result of this key word search is that the desired point in the video is reached.

Additionally in Shahraray et al., "Automatic Generation of Pictorial Transcripts of Video Programs", SPIE Vol. 2417, pp. 512-518, there is described an automatic authoring system for the generation of pictorial transcripts of video programs which are accompanied by closed caption information. The system employs a table having a series of rows, each row containing a pointer to a location of an image, and another pointer to the beginning of a text segment related to the image. A viewing window for a GUI display is shown in FIG. 4 of Shahraray et al., and reproduced herein in simplified form as FIG. 1 of the present patent application. FIG. 1 shows a video image 2, a closed caption text subtitle area 4, and a basic user control area 6. The basic user controls include a "Seek" slider 8.

Accordingly, the state of the art allows for user access to video information based on associated text. However, a more general method for accessing video, not provided by the prior art, would sever the tie between video images and accompanying audio narration or closed caption text.

Disclosure of the Invention

It is therefore an objective of the present invention to provide a system and method for allowing convenient user access to a stored video object, for viewing and browsing, through a communication medium having a client-server architecture, such as the World Wide Web.

It is a further objective of the present invention to provide such convenient user access to a stored video object without requiring that access to the video object to be keyed with any text or other accompanying indicia, not actually part of the video itself.

To achieve these and other objects, there is provided, in accordance with the invention, a method for displaying, on a user terminal, video data object information pertaining to a stored video data object, the stored video object including a temporal beginning point, a temporal end point, and temporal intermediate points therebetween.

The method comprises the following steps:

First, a sequence of representations of points within the video data object, preferably still images, is provided to the user through a user interface. Each representation or still image that is provided corresponds with one of the temporal points in the video data object. In a typical environment, in which the user is an Internet/World Wide Web user, and the video data object is stored in a remote repository accessible through a server, the step of providing the representations includes performing suitable operations to identify points within the video data object, for which representations are to be provided. A preferred technique is to detect scene cuts.

Then, a user interface is provided, including means for allowing a user to select an interval between first and second ones of the temporal points. The user interface is preferably a graphical user interface (GUI), as is commonly made available through computer operating systems such as IBM's OS/2 and Microsoft's Windows operating systems. Also, suitable user interface equipment, such as a video screen and a mouse, are preferably used.

Finally, responsive to user selection of such an interval, a subsequence of representations is provided, each representation of the subsequence corresponding with a respective temporal point in the video data object, each of the respective temporal points falling between the first and second temporal points selected as discussed above.

Therefore, by providing this capability, the invention allows a user to iteratively home in on the portion of a lengthy, large video object that he/she is interested in. The invention advantageously eliminates the need to use text accompanying the video object as a crutch, as was done in the prior art discussed above.

While the invention is primarily disclosed as a method, it will be understood by a person of ordinary skill in the art that an apparatus, such as a conventional data processor, including a CPU, memory, I/O, program storage, a connecting bus, and other appropriate components, could be programmed or otherwise designed to facilitate the practice of the method of the invention. Such a processor would include appropriate program means for executing the method of the invention. Also, an article of manufacture, such as a pre-recorded disk or other similar computer program product, for use with a data processing system, could include a storage medium and program means recorded thereon for directing the data processing system to facilitate the practice of the method of the invention.

Brief Description of the Drawings

The invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

FIG. 1 is an illustration of a prior art graphical user interface for video browsing;

FIG. 2 is a system block diagram of a typical distributed computer system which has internal and external networks, including the Internet, to connect clients to World Wide Web servers and other servers, the client systems being capable of incorporating the invention;

FIG. 3 is a high-level flowchart showing the method of the invention;

FIG. 4 is a flowchart showing a more detailed implementation of step 22 of the flowchart of FIG. 3;

FIG. 5 is a flowchart showing a still more detailed implementation of step 28 of the flowchart of FIG. 4;

FIG. 6 is an illustration of a graphical user interface according to the invention;

FIG. 7 is an illustration of a graphical user interface according to the invention; and

FIG. 8 is a flowchart showing a more detailed implementation of step 44 of the flowchart of FIG. 3.

Description of the Preferred Embodiment

FIG. 2 illustrates a typical communication and processing system, including nodes (computers, processors, etc.) and communication media therebetween, making up an information delivery solution of a typical combination of resources including clients, servers, information repositories, and communication links or networks. Various nodes, shown typically as 9, are also coupled to one another by various networks, shown generally as 7, which may include local area networks (LANs), wide area networks (WANs), etc.

The discussion of FIG. 2 which follows will focus on certain individual components for clearer illustration and description of the novel and non-obvious features of the present invention. In the system of FIG. 2, a client machine 10 includes a computer or other device (as discussed above), running a Web browser program. The program run by the client 10, which incorporates the invention, may be provided to a general purpose computer by means of a commercial computer program product, such as a pre-recorded floppy disk 11 or other suitable computer-readable recording medium. In such a computer program product, the novel and non-obvious features of the invention, implemented as program code, are pre-programmed onto the disk 11 for convenient sales and marketing, and for directing operation of the client machine 10 in accordance with the invention.

The client 10 is coupled, through a communication network 12 such as the Internet, to a Web server 14. Physical access to the Internet is provided in conventional fashion. The general command protocols, etc., for exchanging messages between the client 10 and the server 12, are also conventional.

The client machine 10 may preferably be an intelligent computer system, such as an IBM PS/2 computer, an IBM ThinkPad laptop computer, or an IBM RISC System/6000 workstation. Communication coupling made to the network via a suitable communication interface software tool, such as IBM's OS/2 WARP Connect software product. (IBM, PS/2, ThinkPad, RISC System/6000 and OS/2 are trademarks of IBM Corporation.)

The Web browser in the client machine 10 may preferably be the IBM Web Explorer software product, or equivalent software tools such as the NetScape or Mosaic tools. This computer system 10 is bi-directionally coupled with the OS/2 WARP Connect facility over a line or via a wireless system to the server machine 14. The server machine 14 may preferably be another IBM PS/2 computer, an IBM RISC System/6000 workstation, or other similar system.

The program run by the server 14, which incorporates the invention, may be provided to a general purpose computer on a pre-recorded medium, as discussed above, such as a floppy disk 15, in which case the novel and non-obvious features of the program code are pre-programmed onto the disk 15 for convenient sales and marketing, and for directing operation of the server machine 14 in accordance with the invention.

FIG. 3 is a high level flowchart showing the operation of the method of the invention. A preferred embodiment of the invention employs a client machine supporting a user interface utilizing HTML format, and further employs the World Wide Web. Much of the functionality of the invention may be carried out at the server. However, the description which follows will be readily understandable from the vantage point of the user at the client machine.

Initially, the user selects a video object (step 20). This may be done in any suitable way. The Web provides numerous formats, techniques, etc., for selecting objects. In the preferred embodiment, selection of the video object by the user causes a request to be sent over the Web from the client machine to the server machine. In particular, a Web page, provided by the server 14 to the client 10, may include a graphical interface, such as Hypertext links, for allowing the user to select an object. A preferred way of facilitating this user selection is by means of a graphical interface such as the Web page shown in FIG. 6 (discussed below). The Web page of FIG. 6 may be designed and implemented using the known techniques such as those given in the references discussed above.

In step 22, a first set of representations of selected points within the video data object are displayed, responsive to the user request. In the preferred embodiment employing the World Wide Web, the selection of the selected points in step 22 takes place at the server. A more detailed description of the server activity implementing step 22, for this preferred embodiment, is given in FIG. 4.

Referring to FIG. 4, the server initially receives a user request for a video object (step 24). The server accesses the video object (step 26), and selects a set of temporal points in the object (step 28, described in still more detail in FIG. 5). The server then produces representations of the selected temporal points for display (step 30). In a preferred embodiment, employing the Web and HTML links, the representations of the temporal points are thumbnail images corresponding with the temporal points in the video object. Finally, this information is provided for display (step 32), preferably by transmitting, from the server to the client machine, a display such as a Web page. A preferred implementation is that of the Web page shown in FIG. 6 (discussed below).

Before the display is illustrated and described in detail, a more detailed description will be made of step 28 (selecting temporal points). This may be done in any suitable fashion. A few possible ways would be to select temporal points at random, or evenly spaced, within the video object.

However, it is preferred that the temporal points be selected at scene cut points in the video object. In a preferred embodiment of the invention, step 28 is implemented as shown in more detail in FIG. 5.

Referring to FIG. 5, a scene change measure is computed for adjacent frames (step 34). In a particular preferred embodiment, the measure is computed by calculating a normalized correlation between the two frames. (For discussion purposes, the frames will be referred to as the "image" I and the "model" M, without necessarily implying any temporal order or sequence between them.)

It is possible to compute the scene change measure using straight correlation, in which case, each pixel of the image I is multiplied with a corresponding pixel of the model M, and a running sum of the products is accumulated. In such a scheme, correspondence might be established by means of row-and-column coordinates or other suitable methods.

However, a particularly preferred approach is as follows: For each pixel of the image I, a neighbourhood of the corresponding point in the model M is considered. For calculating the frame change measure, a point within the neighbourhood of the model M whose value is closest to the value of the point in the image I, preferably closest in terms of the grey scale value, is multiplied by the value of the point in the image I.

It has been found that using the neighbour point having the nearest value, rather than the exact corresponding point, improves response to motion in areas of high frequency image data. The set of neighbour points in the Model, for a given point in the Image, may be established in any suitable manner, such as by identifying the Image point according to a system such as row-and-column coordinates, and defining the Model neighbourhood as the points whose row and column coordinates matched those of the Image point, or were either one above or one below the Image point coordinate values.

Also, a normalized version of correlation is used, in order to reduce sensitivity to changes in illumination between images of essentially the same scene.

A preferred formula, incorporating both the correlation function and the pixel-neighbourhood search described above, has been found to work well. Specifically, it is easy to threshold, because it is already normalized. That formula is as follows:

$$r = \frac{n\sum_i I_i M_i - \sum_i I_i \sum_i M_i}{\sqrt{(n\sum_i I_i^2 - \sum_i I_i^2)\,(n\sum_i M_i^2 - \sum_i M_i^2)}}$$

In this expression, r is the scene change measure, and n is the number of active pixels used in the measure calculation. Depending on the particular circumstances or the preference of the user, n can be the total number of pixels in the frame, or a subset of the total number used for the calculation.

I_i is the value (such as the grey scale value) for the i-thy one of the pixels in the image I. M_i is the value for one of the pixels in the model M which is to be paired up with I_i for the above-discussed scene change calculation.

The subscript i, for distinguishing between pixels, is a simplified representation for the purpose of the present discussion. Any suitable arrangement for distinguishing between pixels, which would be understood or deemed appropriate to a particular implementation, may be used. For instance, if the pixels are a rectangular array, then the subscript might be an ordered pair, such as row and column counts. A polar coordinate system, or other systems which would be understood to be suitable for a particular implementation, could also be used.

Note that, for this purpose, M_i is either the i-thy pixel, or a pixel in the neighbourhood of the i-thy pixel. That is, if, for instance, a row-and-column coordinate system were used, and a neighbourhood were defined as given above, then the Image point $I_i = I_{r,c}$ and the Model point $M_i = M_{(r+1),c}$ might be paired up, if $M_{(r+1),c}$ had the closest value to $I_{r,c}$ of any of the Model points in the defined neighbourhood.

Once the scene change values have been determined for the various pairs of adjacent frames in the video object, they are compared with a frame change threshold (step 36 in FIG. 4). As stated above, this expression produces a normalized scene change value, the value being 0 for two identical frames. It is convenient to work with a scene change threshold having a value between 0 and 1. However, any suitable measure of the threshold may be employed.

Then, a test is made (step 38) to determine whether the value is greater than the threshold (alternatively, greater than or equal to the threshold; whichever is considered suitable according to the particular implementation). The result of this test determines whether the two frames are treated as a scene change (steps 40, 42).

A preferred graphical implementation of the user interface is shown in FIGs. 6 and 7. In a preferred embodiment, to be used in an environment such as the World Wide Web, the graphical interfaces are implemented using HTML. Details of this implementation will be omitted, since they would be known to persons skilled in the fields of HTML and other graphical user interfaces.

Referring first to FIG. 6, a selection menu is provided for the user. The menu lists video objects which are available. The listing may include thumbnail images 46 which illustrate the content of the video objects, text captions 48 naming or describing the video objects, or other suitable descriptors. Also, the menu may include other information which may be of use to the user, such as the characteristics of the stored file. For instance, FIG. 6 shows a legend "Energy 1 mpeg file" 50, which informs the user that the video data object contains image information which is compressed according to the MPEG data compression standard. The listed video objects are preferably provided in HTML format, so that the user can select a video object (step 20 of FIG. 3) by a suitable means such as mouse-clicking on the thumbnail image or other identifier.

Responsive to the user command, the server 14 obtains the video object and, as per FIG. 4, responds to the client machine 10. In step 44 of FIG. 3, a user interface is provided for viewing and browsing the video object. A preferred embodiment, for use on the Web, is illustrated in FIG. 7.

FIG. 7 shows a graphical image that would be displayed on the client machine for the user's viewing. In accordance with the invention, representations of several temporal points in the video object are shown. Preferably, these representations are shown as thumbnail images 52. For ease of comprehension, the thumbnail images 52 are preferably shown in temporal order, in a user-intuitive arrangement such as from left to right, forming a "storyboard" that lets the user scan his/her eyes from left to right, and get a sense of the sequence of images or events in the video object.

To further assist the user in getting a sense of where the displayed thumbnail images occur in the video object, a representation of the video object in its entirety, and of the portion covered by the displayed thumbnail images, is given. In FIG. 7, this is shown as a slider bar 54. The slider bar 54 is shown horizontally, but can be in any suitable configuration, preferably a configuration which comports well with user intuition. The entire horizontal length of the slider bar 54 represents the total duration of the video object, and a marker 56 represents the relative location, within the video object, of the displayed thumbnail images. Preferably, the marker 56 has a width, relative to the width of the entire slider bar 54, which reflects the portion of the entire video data object covered by all of the displayed representations.

The marker 56 may be moved using the technique, familiar to users of slider bars in conventional GUI window applications, of moving a cursor onto the marker 56 with a mouse, holding the mouse button down, and dragging and dropping the marker 56 by moving the mouse. Alternatively, a GUI control panel 55 may be provided, having buttons such as the left, right, fast left, and fast right button shown. By positioning the cursor on one of these buttons, or other suitable control buttons, and pressing a mouse button, the user causes the marker 56 to scroll along the slider bar 54.

Regardless of what particular technique the user uses to move the marker 56 along the slider bar 54, representations of points within the video data object appear and disappear to represent the key frames in whichever part of the video data object the marker 56 is covering at that moment. Thus, by moving the marker 56, the user selects a portion of the video data object which he or she wishes to view in more detail.

Also in accordance with the invention, identifying indicia such as consecutive numbers may be assigned to the key frames, to assist the user in keeping track of which key frames have been examined, or where, within the video data object as a whole, a given image appears. This information may be provided in the user interface. A shot count 57 displays the indicia, in the form of consecutive numbers. For instance, in FIG. 7, the marker 56 is at the left end of the slider bar 54. Let us say, then, that the first six thumbnails in the video data object being viewed are displayed. Accordingly, the shot count 57 displays a value of 1, indicating that the left-most thumbnail image represents the first thumbnail (i.e., the earliest temporal point) of the video data object. If a user were then to use the controls 55 or the marker 56 to move through the video data object, then, as the marker 56 moved to the right and different thumbnails 52 appeared for later points in the video data object, the shot count 57 would likewise ascend in value. As a result, a user might identify a shot of particular interest, and remember it for easy future access by taking note of the value of the shot count.

Step 22 of FIG. 3 is illustrated, in a preferred embodiment, by the thumbnail images 52 and the slider bar 54, discussed above.

Also in accordance with the invention, the user interface includes means for selecting an interval within the data objects. The interval is selected in terms of the displayed representations. Preferably, the user selects one of the representations, and the interval is selected in terms of an interval between the selected representation and another one of the representations, such as the temporally subsequent one of the representations. The method step 44 of FIG. 3, in which a user interface is provided for facilitating viewing and browsing, is implemented by providing the means for selecting.

In accordance with the invention, the user interface provided in step 44 includes means for selecting an interval between two of the key frames. Preferably the means for selecting an interval operates in conjunction with the graphical user interface GUI of the system according to the invention. That is, a user uses a GUI input device, such as a mouse, to select a representation of a desired interval from several representations of intervals shown on the display.

A preferred way of performing step 44 is shown in FIG. 8.

Referring to FIG. 8, a user is initially presented with a set of representations of frames of the video object, such as the sequence 52 of thumbnails shown in FIG. 7. The user views the thumbnails and decides that a portion of the video object which he/she would like to examine in more detail falls between two of the thumbnails. The user selects the interval (step 58) through the graphical user interface. In the preferred embodiment of FIG. 7, for instance, the user selects one of the thumbnail images. The thumbnail image is implemented as a hypertext link, and the user's selection of the thumbnail indicates that he/she is interested in the interval between that thumbnail and an adjacent thumbnail (preferably the thumbnail which follows the selected thumbnail, temporally).

The user's selection of the thumbnail is processed in known manner as a hypertext selection, and a message is sent to the server. The server examines the portion of the video object between the selected frame and the next frame. In a manner similar to that of step 28 in FIG. 4, the server selects a new set of key frames falling between the selected frame and the next frame (step 60).

The selected new frames are then displayed (step 62), essentially in the same manner as before (steps 30 and 32 of FIG. 4). Also, the slider bar 56 reflects the selection of the interval by moving and/or changing in width.

In many cases, the user will want to perform several iterations of this process, in order to narrow down to the particular small area of interest within the video object. Accordingly, a repetition loop is shown in FIG. 8 (step 64 and the loop back to step 58).

Claims

1. A method for facilitating user examination and browsing of video data object information pertaining to a stored video data object (Object), the Object including a beginning point, an end point, and points therebetween, the method comprising the steps of:

providing (30), to a user interface, a sequence of representations, each representation corresponding with one of the points in the Object;

a user utilizing a user interface to select (60) an interval of the Object in terms of a first one of the points; and

responsive to user selection of such an interval, providing (62) a subsequence of representations, each representation of the subsequence corresponding with a respective point in the Object, each of the respective points falling within the selected interval.

2. A method as claimed in claim 1, wherein the step of providing (30) a sequence of representations includes communicating, through a network communication medium (7), with a video repository (14) having video data objects, including the Object, stored therein.

3. A method as claimed in claim 1, wherein the step of providing a sequence of representations includes displaying (32) still images representative of content of the Object at respective ones of the points in the Object.

4. A method as claimed in claim 1, wherein the step of providing a sequence of representations includes:

generating thumbnail images based on content of the Object at respective ones of the points in the Object; and

providing thumbnail images representing still frames of the video object at the respective points therein.

5. A method as claimed in claim 1, wherein the step of providing a user interface includes:

displaying a representation of the Object in its entirety; and
displaying a representation of a portion of the Object covered by the displayed representations of points in the Object.

6. A method as claimed in claim 5, wherein:
the step of displaying the representation in its entirety includes displaying a slider bar (54); and
the step of displaying a representation of a portion of the Object covered by the displayed representations of points in the Object includes displaying a marker (56) on the slider bar (54).

7. A method as claimed in claim 6, wherein:
the step of selecting (60) an interval includes moving the marker (56) along the slider bar (54) from a first position thereon to a second position thereon; and
the step of displaying the sequence of representations (52) includes the step (62), responsive to movement of the marker along the slider, of displaying representations of points within the Object corresponding with the second position of the marker, in place of previously displayed representations of points within the Object corresponding with the first position of the marker.

8. A method as claimed in claim 1, wherein the step of selecting an interval includes selecting one of the representations of still images which forms a boundary of the interval to be selected.

9. A method as claimed in claim 8, wherein the step of selecting an interval further includes selecting one of the representations of still images which forms a first boundary of the interval to be selected, a next one of the representations forming a second boundary of the interval.

10. A system for facilitating user examination and browsing of video data object information pertaining to a stored video data object (Object), the Object including a beginning point, an end point, and points therebetween, the system comprising:

a display unit;

an interface for coupling the display unit with a network communication medium (7), the medium being coupled to a video repository (14) having video data objects, including the Object, stored therein;

means for sending, through the interface and over the medium, a request to view the Object;

means, responsive to receipt of a response to the request to view, for displaying a sequence of representations (52), each representation corresponding with one of the points in the Object;

means (56) for selecting an interval in terms of a first one of the points; and

means, operable responsive to operation of the means for selecting an interval, for providing a subsequence of representations, each representation of the subsequence corresponding with a respective point in the Object, each of the respective points falling within the selected interval.

11. A system as claimed in claim 10, wherein:
the network communication medium (7) includes the Internet;
the interface includes an Internet interface; and
the video repository (14) includes a server on the Internet.

12. A system as claimed in claim 10, wherein the beginning point, the end point, and the points are temporal points within the Object.

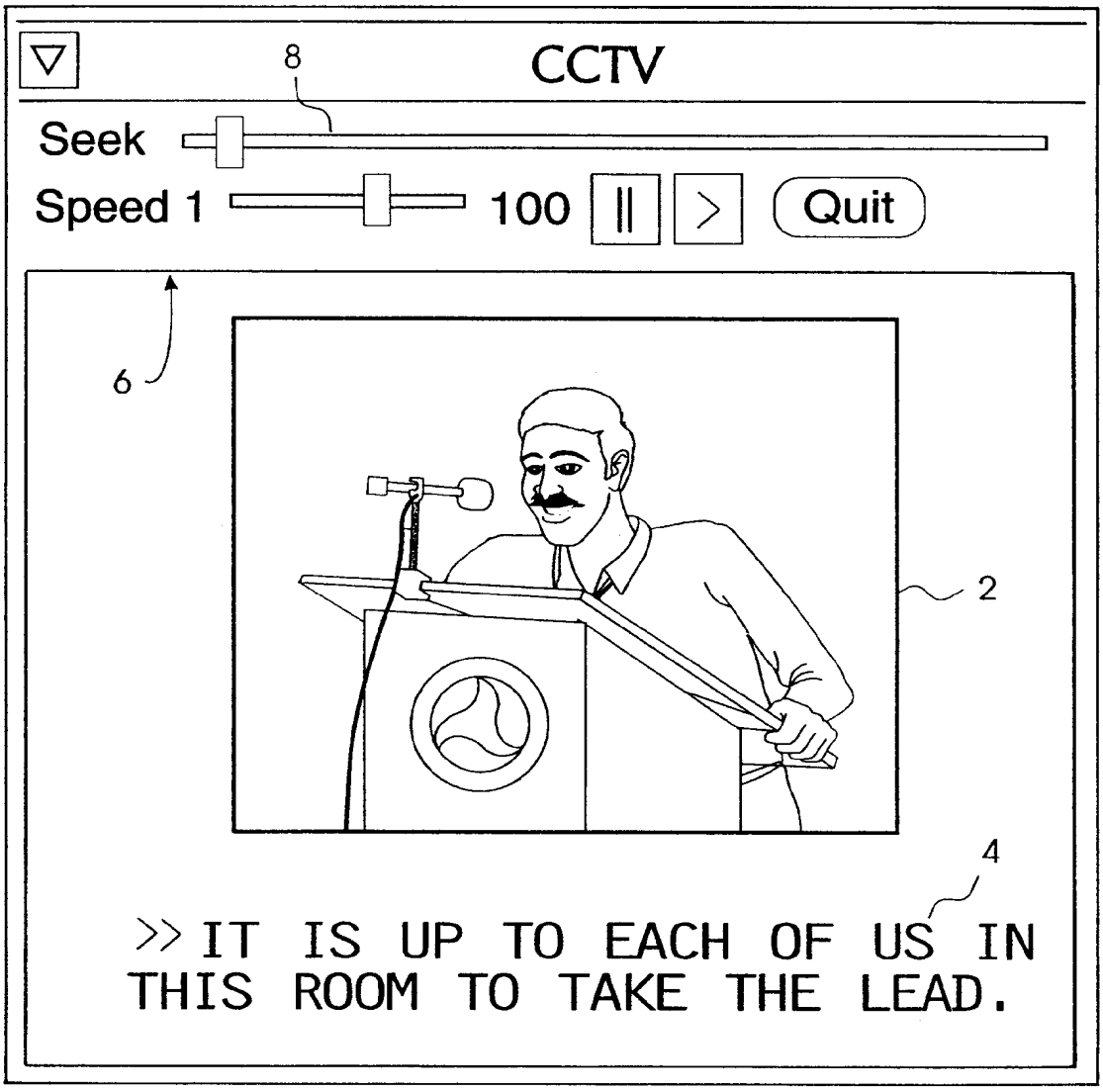


FIG. 1
(PRIOR ART)

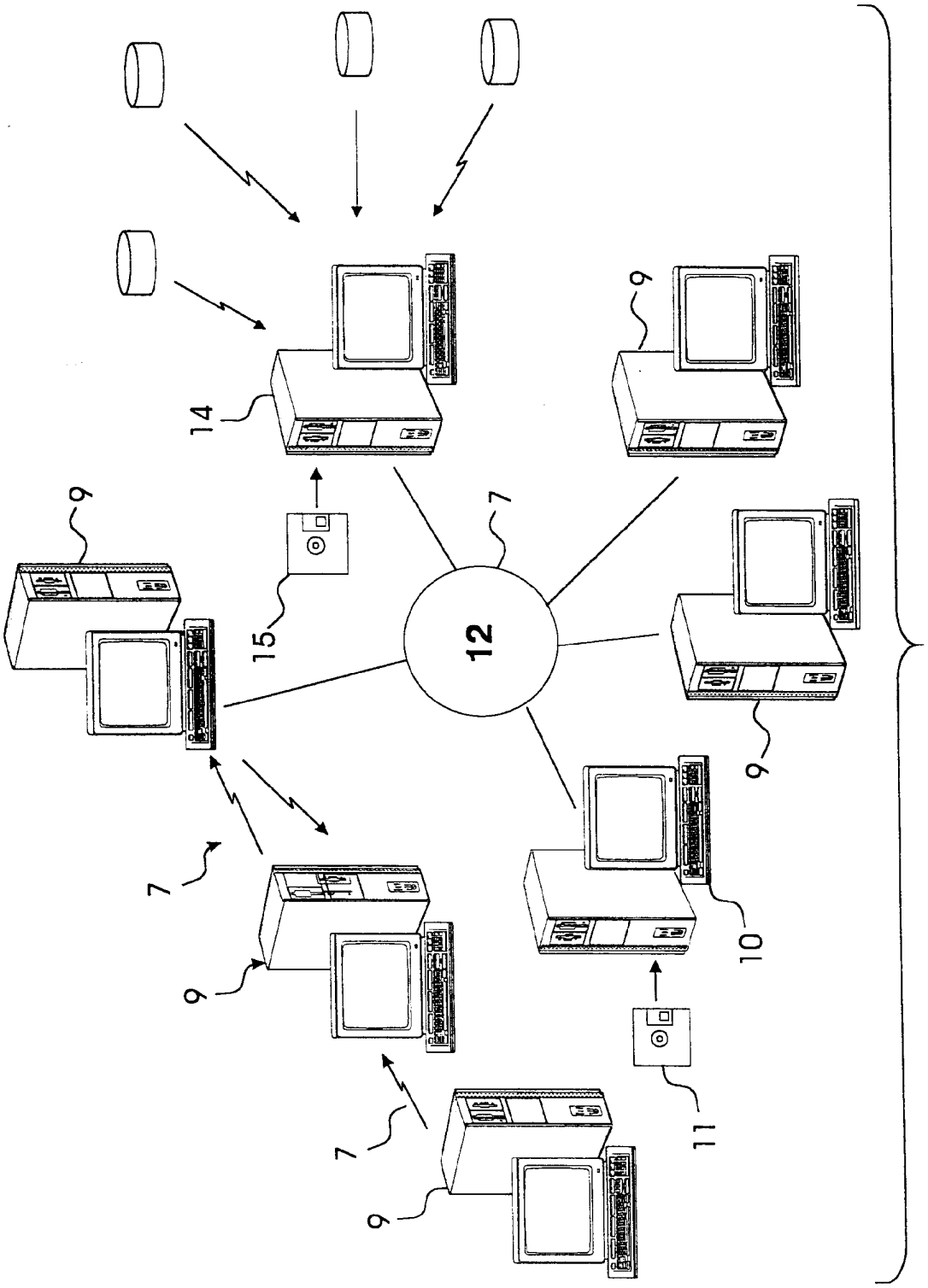


FIG. 2

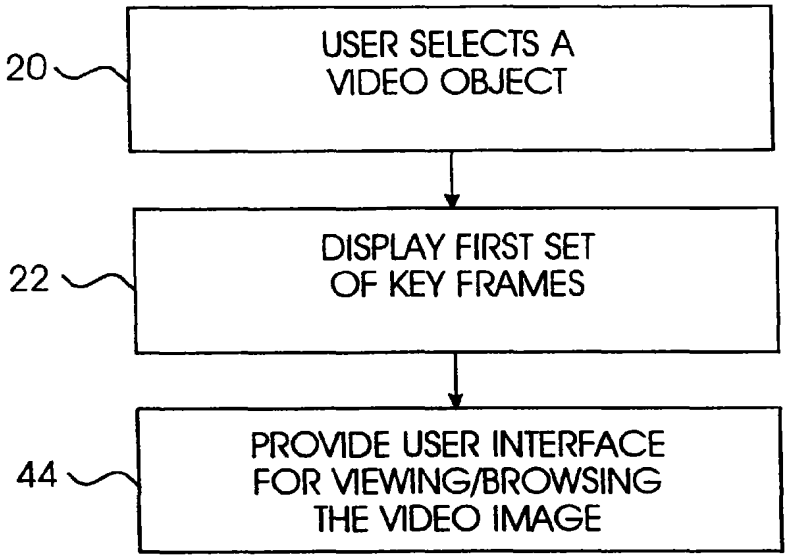


FIG. 3

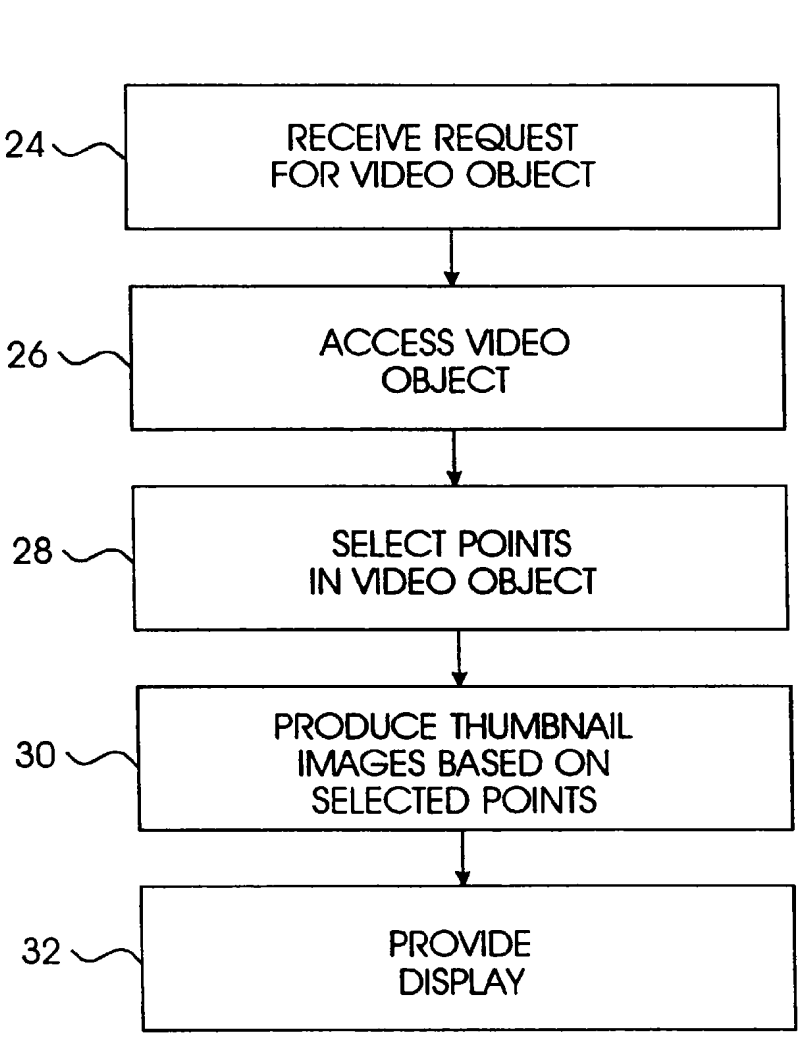


FIG. 4

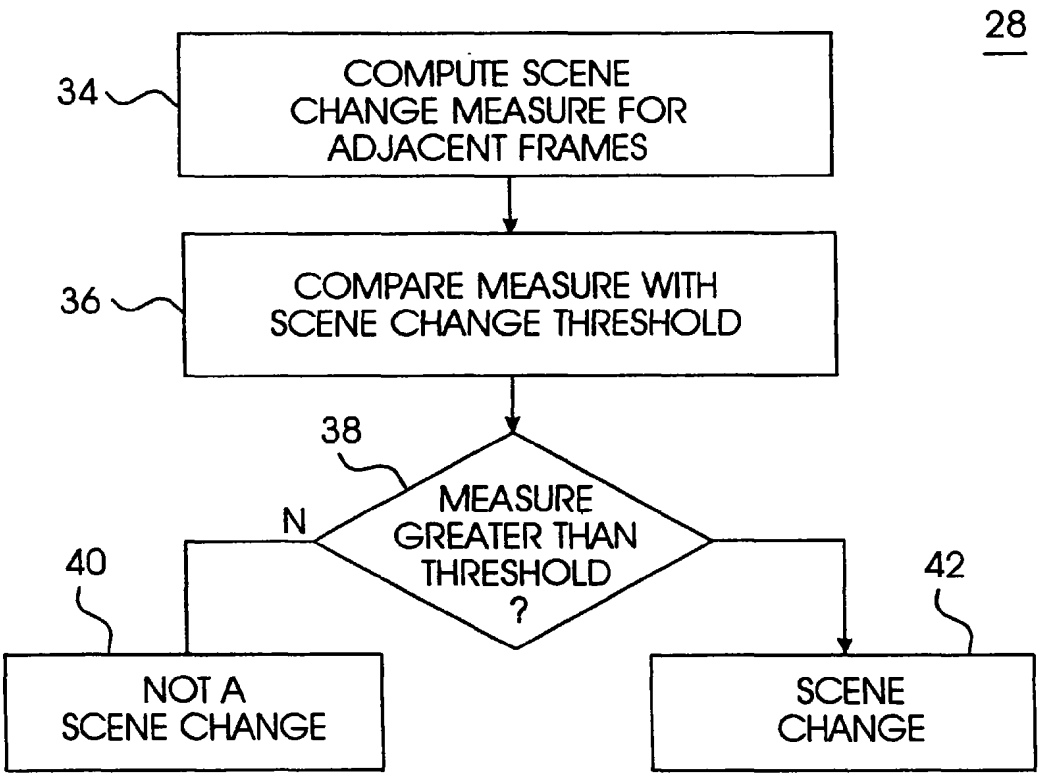


FIG. 5

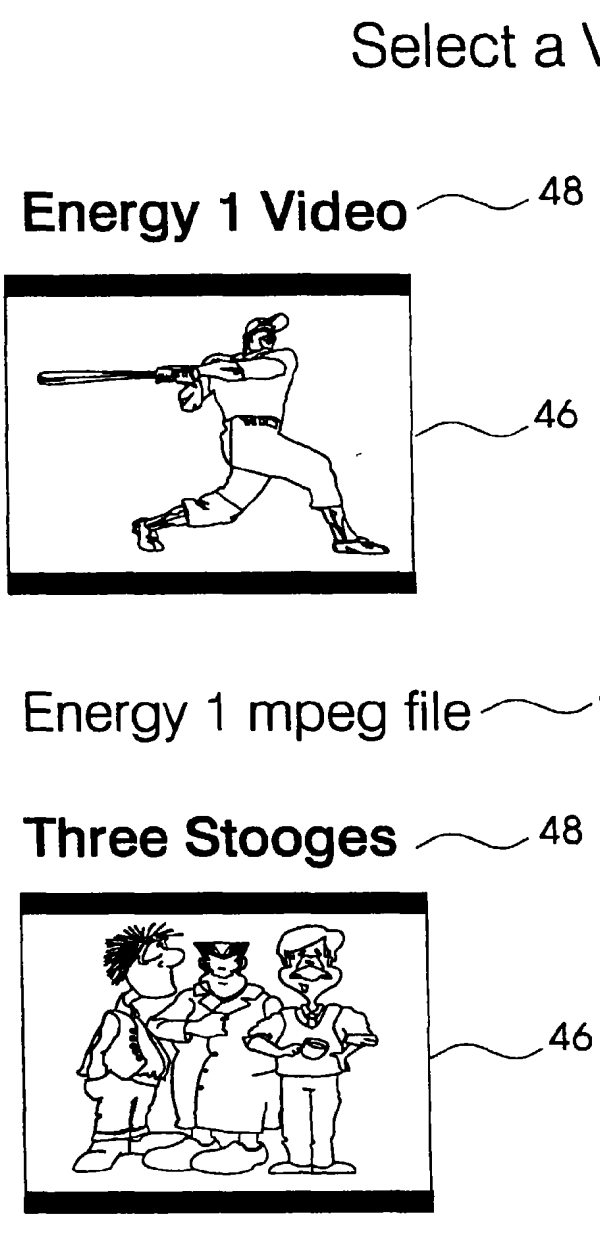


FIG. 6

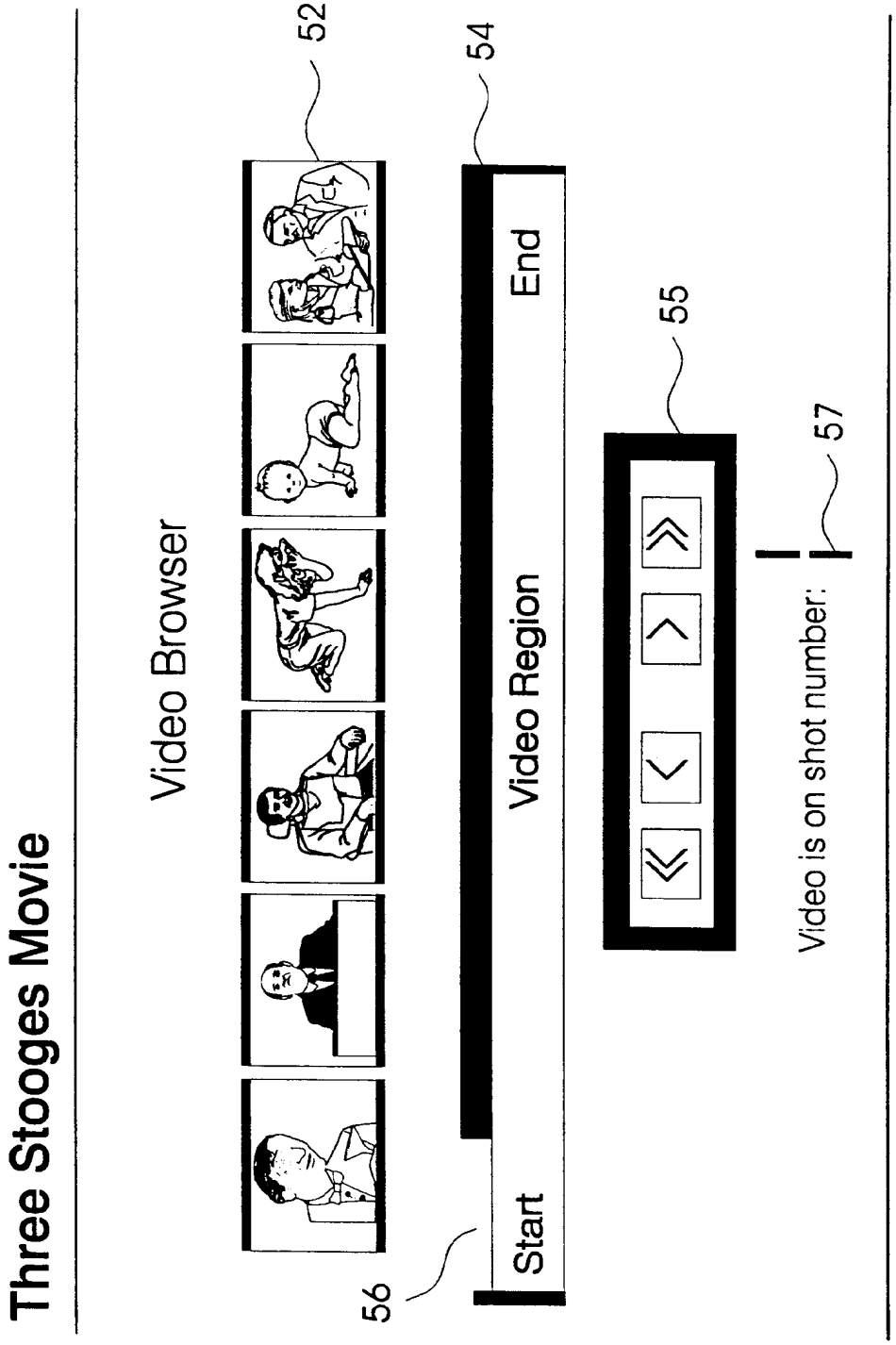


FIG. 7

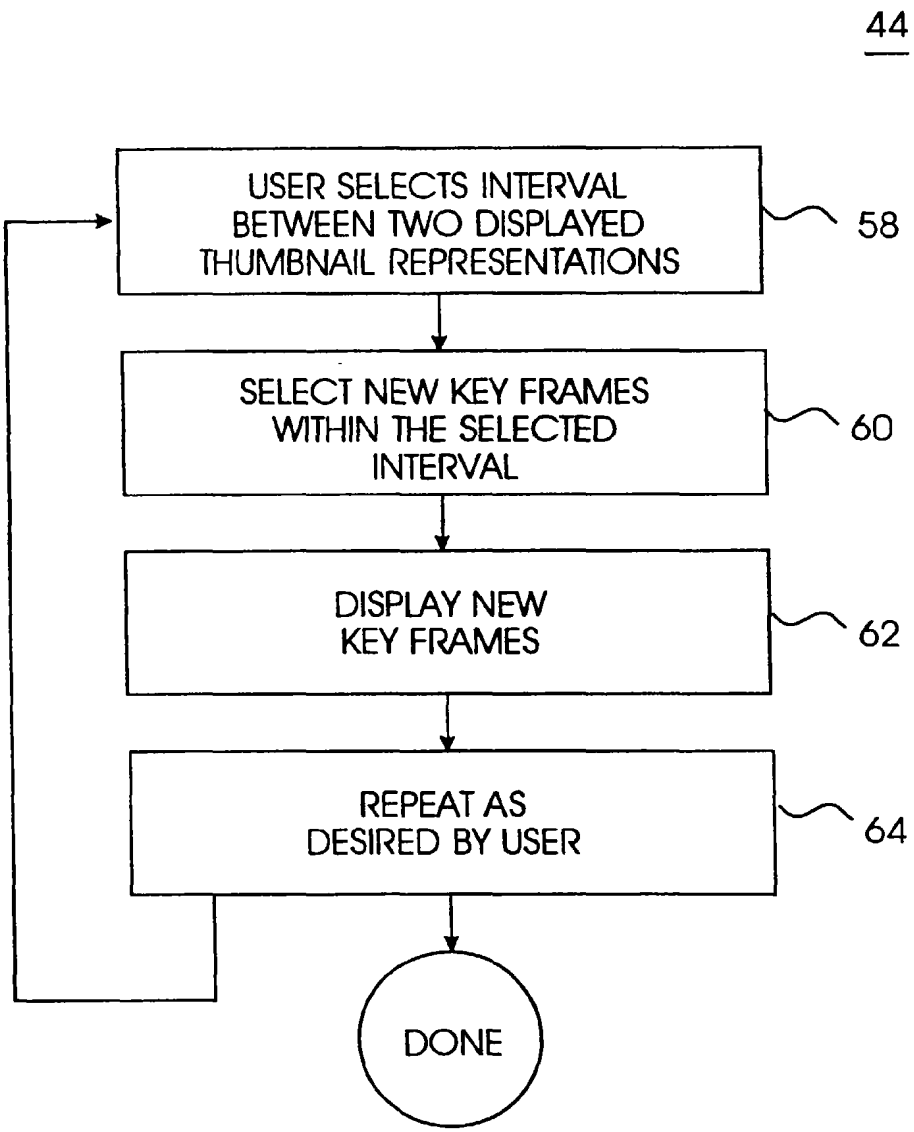


FIG. 8

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 96 30 8817

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	CHI'92 CONFERENCE PROCEEDINGS , MONTEREY, MAY 3 - 7, 1992, no. -, 3 May 1992, BAUERSFELD P;BENNETT J; LYNCH G, pages 93-98, XP000426811 MILLS M ET AL: "A MAGNIFIER TOOL FOR VIDEO DATA" * the whole document * ---	1-12	G06F17/30
X	EP 0 619 550 A (IBM) 12 October 1994 * abstract; claims; figures 3,5 * ---	1-10,12	
A	PROCEEDINGS OF THE REGION 10 ANNUAL INTERNATIONAL CONFERENCE (TENCO, SINGAPORE, 22 - 26 AUG., 1994, vol. 2, 22 August 1994, CHAN T K Y (ED), pages 852-856, XP000528240 SMOLIAR S W ET AL: "INTERACTING WITH DIGITAL VIDEO" * page 854, right-hand column, paragraph 2.3 - page 855, right-hand column, paragraph 4 * ---	1-12	
A	IEEE MULTIMEDIA, vol. 2, no. 1, 21 March 1995, pages 12-25, XP000500083 WEISS R ET AL: "COMPOSITION AND SEARCH WITH A VIDEO ALGEBRA" * page 22, right-hand column, line 1 - page 24, left-hand column, line 6; figures 2,9,10 * -----	1,10,11	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)
G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	9 April 1997	Fournier, C

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 818 907 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
14.01.1998 Bulletin 1998/03

(51) Int. Cl.[6]: **H04L 29/06**, H04N 7/173, H04L 12/14, H04M 15/00

(21) Application number: **97111872.4**

(22) Date of filing: **11.07.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV RO SI

(30) Priority: **12.07.1996 US 678915**

(71) Applicant: **AT&T Corp.**
New York, NY 10013-2412 (US)

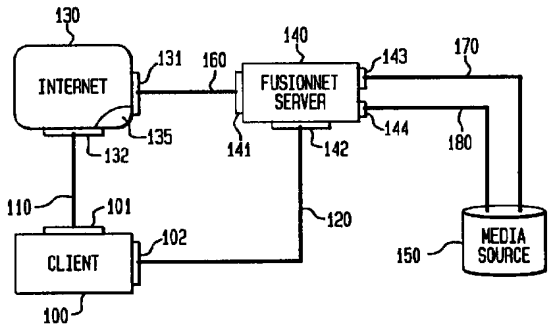
(72) Inventor: **Civanlar, Mehmet Reha**
Red Bank, New Jersey 07701 (US)

(74) Representative:
KUHNEN, WACKER & PARTNER
Alois-Steinecker-Strasse 22
85354 Freising (DE)

(54) Improved client-server architecture using internet and guaranteed quality of service networks for accessing distributed media sources

(57) An improved client-server architecture of the present invention utilizes the advantages of known QOS networks to provide guaranteed quality of service, security, and a charge mechanism for handling requests initiated over a packet network, such as the Internet, for access to distributed media sources. Such media sources may be independent of the QOS network provider and may be located by browsing the Internet. A method of operating a client-server network enables the system level merger of the Internet and a guaranteed QOS network, such as the public switched telephone network, in order to provide the users with a complete information superhighway today. It will appear to the average user that the Internet and QOS network are fused together. Thus, when a user, connected to the Internet, selects an application that requires functionalities offered by the telephone network, such as guaranteed QOS delivery of media information or customized billing, the Internet-resident application will communicate information to a server, which will in turn initiate a session over the QOS network for delivery of the required information to the client using client information transmitted from the client (or from the application) to the server over the established Internet session. The client information may include a client account number, login and password, and/or phone number to enable the server to establish the switched network connection to the client. Accordingly, media sources which are separate and independent from the QOS network provider may be accessed using a secure, guaranteed QOS network in a manner providing for ease of identification and billing.

FIG. 1



EP 0 818 907 A2

Description

The present application is related to U.S. Application Serial No. 08/402,664 entitled Client-Server Architecture Using Internet and Public Switched Networks, filed Mary 13, 1995, and to U.S. Application Serial No. 08/648,260 entitled Multimedia Information Service Access, filed on May 15, 1996. These three applications are co-pending and commonly assigned.

Technical Field

This invention relates to the Internet network and, more particularly, provide an improved client-server architecture utilizing a packet network, such as the Internet, and a guaranteed quality of service network, such as the public switched network, for access to distributed media sources.

Background of the Invention

The Internet's global and exponential growth is common knowledge today. The Internet is implemented using a large variety of connections between millions of computers. Internet access is readily available to individuals across the globe. Various on-line service providers, such as America Online, CompuServe, Prodigy, Netcom, etc., use PSTN with modems or ISDN adapters for client connections. These on-line service providers maintain servers on the Internet providing client access to the Internet.

The recent developments on the World Wide Web user interfaces and information navigation software such as the Netscape web browser, coupled with a continuously growing number of public access providers, are making the Internet a fundamental component of the information age, if not the information super highway itself. World Wide Web sites on the Internet are typically accessible through a browser program which interprets scripts written in Hyper Text Markup Language (HTML). Users may browse the World Wide Web for virtually any kind of information, including information having content derived from one or more media, such as words, sounds or images. It is desired to enable the on-demand access to a variety of sources of media information using the Internet.

On the other hand, it is also well known that the current Internet is deficient, as compared to other existing networks, in the following three fundamental functionalities expected from a complete information network: 1) quality of service (QOS), 2) security, and 3) an easy and flexible mechanism to charge for the information and transmission services. These deficiencies discourage not only media providers from providing real-time on-demand access to media source information over the Internet, but also users who might otherwise desire to purchase real-time on-demand access to information held by such media sources. Although recent developments in the Internet Protocol (IP) and in Internet applications show promise for providing tools for secure transmission and real-time support, without a globally-agreed and implemented billing mechanism, guaranteed QOS over a global network is not possible. Consequently, none of these mechanisms can provide reliable, real-time global transmission employing the Internet today. One such service which may not be adequately provided by the Internet would be a request for delivery of a long video or audio segment. It may not be possible to have a guaranteed real-time delivery of such a long video and audio segment because one or more of the routers or computer links in the Internet may be busy handling other requests.

The existing telephone networks, on the other hand, have been offering the capabilities of guaranteed quality of service, security, and an easy and flexible mechanism to charge for the information and transmission services for a long time; the level of QOS, security and ease and flexibility of billing provided by existing telephone networks clearly surpasses the levels of these functions afforded by the Internet. What is desired is a practical way to utilize these advantages to provide guaranteed quality of service, security, and a charge mechanism for handling requests to access media sources that may be located by browsing the Internet.

Pending U.S. Application Serial No. 08/402,664 and U.S. Application Serial No. 08/648,260 have recognized these shortcoming of the Internet and the advantages of using a QOS network, such as the public switched telephone network, in tandem with the Internet to provide guaranteed quality of service, security and an easy and flexible billing mechanism. The present invention utilizes these advantages to enhance access to distributed media sources that are unbundled from or separate from the server architecture.

Summary of the Invention

An object of the improved client-server architecture of the present invention is to utilize the advantages of known QOS networks to provide enhancements in functions, such as quality of service, security, and billing, for handling requests initiated over a packet network, such as the Internet, for on-demand access to distributed media sources. Such media sources may be independent of the QOS network provider.

In accordance with the present invention, a method of operating a client-server network enables a system level merger of a packet network, such as the Internet, and a guaranteed QOS network, such as the public switched network, in order to provide the users with a complete information superhighway today. As the present invention will make it appear to the average user that the packet network and the QOS network appear to be fused together, the present invention will be referred to from time-to-time as "FusionNet".

When a user engaged in an Internet session

selects an application that requires enhanced functionalities offered by the telephone network, such as guaranteed QOS delivery of media information or customized billing, the Internet-resident application will communicate information to a FusionNet server, which will in turn initiate a communications session for on-demand delivery of the required information to the client over the QOS network using client information transmitted from the client (or from the application) to the server. The client information may include a client account number, login and password, and/or phone number to enable the server to establish the switched network connection to the client. Billing for service and transmission may proceed as in a normal or collect telephone call. There is no security problem or complicated identification mechanism. Also, because of the built-in security of the telephone networks, FusionNet's guaranteed QOS path does not require a firewall between the client an the outside world in order to avert unauthorized access.

Accordingly, media sources which are separate and independent from the QOS network provider may be accessed using a secure, guaranteed QOS network in a manner providing for ease of identification and billing.

Brief Description of the Drawings

FIG. 1 shows an illustrative client-server architecture using both the Internet and a guaranteed QOS network in accordance with an embodiment of the present invention.

FIG. 2 shows another depiction of an illustrative client-server architecture using both the Internet and a guaranteed QOS network connected to a media source through a gateway in accordance with the present invention.

FIG. 3 shows a variation of the client-server architecture in which a client connects to the Internet and to the FusionNet server through a service provider in accordance with the present invention.

Detailed Description

The Internet is a dynamic packet network consisting of millions of interconnected computers which could run several applications, such as the World Wide Web. Web browsers such as the one by Netscape are programs using a graphical user interface that provides a user easy access to various services over the Internet. The present invention enables the above-identified functionality of a QOS network, such as the public switched network, to be combined with the Internet functionality to enhance services provided by existing Internet applications and to create new ones.

The present invention provides a seamlessly integrated system that makes it possible to use a packet network, such as the Internet, together with a QOS network that offers functions such as 1) quality of service, 2) security and 3) an easy and flexible charging mechanism at an enhanced level (as compared with the packet network). The QOS network may be a known switched network, such as the public switched telephone network (PSTN), the integrated services digital network (ISDN), an asynchronous transmission network (ATM), a cable network used for video or television programs, etc., or it may also be private or corporate communication or data network as long as it provides enhancements in at least one of the three functionalities described above.

A high level block diagram for a FusionNet services architecture is shown in FIG. 1. With reference to FIG. 1, the system is based on a client-server architecture, where the client apparatus 100 may be either a personal computer (PC), a single workstation or two such systems which can be operated in a coordinated manner (e.g., physically close systems, users who talk over the telephone, etc.). The key factor is that each client station 100 requires two logical network connections or interfaces, e.g., 101 and 102. One of these interfaces 101 provides a connection to the Internet 130 which can be made through one of a number of known channels, e.g., a local area network (LAN) connection, a Serial Line Internet Protocol (SLIP) or Point-to-Point Protocol (PPP) connection over a modem or over an ISDN port. These connections are made through network 110 which may be either a private connection or the public switched telephone network. The second interface 102 provides a connection to guaranteed QOS network 120, e.g., a public or private switched telephone network, which may also be made through a modem, an ISDN port, or via a connection to a special LAN such as an ATM LAN or a LAN that offers bandwidth reservations.

It should be noted that these two interface connections 101 and 102 are defined at the logical level, that is, client apparatus 100 may have a single physical connection (i.e., modem, ISDN adapter, etc.) that can be used to attach it to both the Internet 130 and the guaranteed QOS network (e.g., public switched network) 120. For example, a modem with "Call Waiting" functionality, an ISDN line used as two separate B channels, an ATM LAN connection where the PSTN connection can be accomplished through a bridge, or a video telephony connection conforming to the CCITT H.320 standard which provides Internet connectivity over its data channel can provide the required two logical connections over a single physical connection.

Similarly, a FusionNet server 140 has access to both Internet 130 and the QOS network 120 via logical network interfaces 141 and 142. A FusionNet server 140 may be a computer or other software-driven machine having processing capability for carrying out the server functions herein described, or it may be a software package that includes those sever functions and that operates on a computer or other software-driven machine. Generally, the system of FIG. 1 enables a plurality of client computer-based apparatuses 100 to access a plurality of FusionNet servers 140 via Internet 130 and QOS network 120. Typically, access to the

Internet is made through a logical network interface 132. Typically, access by a client apparatus 100 to FusionNet server 140 via the Internet 130 utilizes the HyperText Transport Protocol (HTTP) which is accessed using a browser application program available at the client apparatus 100.

A general overview of the operation of the present invention is as follows. A user at a client apparatus 100 connects to Internet 130 and requests an access to media information requiring enhanced functionalities not available on Internet 130, but offered by QOS network 120. Illustratively, an enhanced service request would be one requesting a guaranteed QOS delivery, security, or customized charging. Providing such enhanced Internet services would enable the delivery over the switched network of 1) real-time video transmission, 2) real-time, high quality audio transmission, or 3) immediate access to sensitive data such as stock market data, as just three examples. The user may locate such desired media information by, for example, browsing the Internet and locating references to one or more media sources at a variety of World Wide Web sites. World Wide Web sites on the Internet are typically accessible through a browser program. As an example, a World Wide Web site could contain information about one or more media sources, such as the type of media (e.g., music, text, images, or movies), available selections (e.g., a list of movies by title and lead performers), and pricing information. Alternatively, other information sources available over the Internet (e.g., a USENET newsgroup posting) could provide information about a media source.

Upon locating a desired media source 150 by browsing a World Wide Web site 135, the user selects access to the media source 150 by clicking on a reference point contained in Web site 135, which in turn initiates a program to establish a communications session between Internet 130 and FusionNet server 140 using network 160 and logical network interfaces 131 and 141. During this process a program running from Web site 135 may request information about the client, for example the client's telephone number, a login name and password, or an account number. The client information along with information identifying the selected media source is then passed along to FusionNet server 140 over communications path 160. Functions such as these may be implemented using the well-known Common Gateway Interface (CGI) for which public software implementations are available for HTTP-compatible servers using common computer communications interfaces such as sockets. The CGI is a publicly-documented open interface with a specification that can be readily obtained from the Internet. For example, an Internet World Wide Web site maintained by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign contains a description of CGI and its specification as well as a CGI program archive.

FusionNet server 140 then undertakes to establish a communications session with the selected media source 150, utilizing network 170 and logical network interface 143, for purposes of controlling the selection and delivery of media information available through the media source. An address for the media source, if necessary, may be provided by the Web site 135 or stored by the FusionNet server 140. Communications path 170 can be any one of a number of known communications channels, e.g., a private communications network, the PSTN, or even the Internet. Once a communications session is established between FusionNet server 140 and selected media source 150, FusionNet server 140 could then query the selected media source for availability of the desired media information.

After FusionNet server 140 establishes the communications session with selected media source 150 over communications path 170, the server 140 then initiates a communications session over a QOS network 120 with client 100. This communications session may be established, for example, by initiating a call connection over the public switched network using a telephone number provided by the client 100 as part of the service request. Several programs for automatic dialing of PSTN or ISDN numbers are available for various computers (e.g., PictureTel™ PCS series) and can be used for this purpose. Client apparatus 100 would include a program that accepts incoming phone calls. Such software is also available from many vendors (e.g., PictureTel™ PCS series). Such a call may be billed by any of the standard call billing methods, e.g., collect or third party billing, etc. The QOS network 120 chosen for the communications session provides the billing for the service and transmission, as well as enhanced security, and complicated identification mechanisms are not needed. In a typical application or service request between a client and server, both the Internet and QOS network connection may coexist, or the Internet session may be discontinued. Another possibility is to use an already established client account for billing purposes.

The actual transaction, e.g. security checks and sending a file or audio/video data over the QOS connection, can then be handled by the same process or yet another process. FusionNet server 140 completes the QOS connection between the client 100 and the selected media source 150 by establishing communications session over path 180 between the server 140 and media source 150. Communications path 180 may also be made using the PSTN, or ISDN network, or another QOS network. FusionNet server 140 then causes the media source to transfer media information to client 100 over the OQS connection, and server 140 can retain control over the transfer process by causing media source 150 to, e.g., stop, pause, or retransmit the requested media information.

Typically, QOS network connections 120 and 180 will be of a common type (e.g., each PSTN) such that FusionNet server 140 may simple implement a switched

connection of the two QOS network connections providing therefore a single connection between client 100 and media source 150 to permit transfer of media information from the source 150 to client 100. It will be obvious to one skilled in the art that the media information could be transferred, subject to the control of FusionNet server 140, from media source 150 to client 100 over a QOS network directly connected between source 150 and client 100 without routing the media information through FusionNet server 140.

Where the transfer of information through network 120 and network 180 requires more than just a simple switching operation, e.g., data conversion or data reformatting is required, a FusionNet gateway 100 as shown in FIG. 2 operates to transfer media information from source 150 to client 100. All of the other elements in FIG. 2 are the same as those in FIG. 1, and therefore bear the same reference numerals. Once the QOS gateway connection is established, FusionNet server 140 instructs media source 150 to transmit the desired media information to client 100. The FusionNet gateway 200 may, if necessary, translate or reformat the information as it is transferred from source 150 to client 100. Gateway 200 may, alternatively, adjust to the client's information protocol. At the end of the transaction, the QOS gateway connection may be dropped. FusionNet server 140 may then complete any remaining billing procedures for the transfer.

A variation of the architecture described above may be derived by shifting the client connection functionality to a FusionNet service provider as shown in FIG. 3. The client makes a single connection to the FusionNet service provider which, in turn, would implement the two logical connections for the Internet and the QOS network connection to the FusionNet server, respectively. This has the advantage of providing a client access to the FusionNet architecture by making only a single physical connection which may reduce the complexity of client apparatus as well as remove the need for the client to coordinate two logical connections. Referring to FIG. 3, a user at client station 300 establishes a connection 310 with FusionNet service provider 320. Connection 310 is a point-to-point connection made using logical interfaces 301 and 321 through a QOS network that may be one of the known channels such as the PSTN, or ISDN or a private telephone network.

In this variation of the client-server architecture, FusionNet service provider 320 has the capability of establishing communications sessions with the Internet and with a FusionNet server that are functionally equivalent to the sessions among the client, the Internet and the FusionNet server described above with reference to FIGS. 1 and 2. In FIG. 3, a communications session is established between the FusionNet service provider and the Internet using communications path 330 and logical interface 322, which correspond to path 110 and logical interface 101 shown, respectively, in FIGS. 1 and 2. Similarly, in FIG. 3 a communications session between the FusionNet service provider and the FusionNet server is established using path 340 and logical interface 323, which correspond to path 120 and logical interface 102 shown, respectively, in FIGS. 1 and 2. FusionNet service provider 320 also has the capability of obtaining and supplying the client information, and transferring the desired media information obtained from the selected media source to the client. Since the user is connected to the FusionNet service provider through a point-to-point dedicated connection, the advantages of using a QOS network as described above can be preserved.

The remaining steps in obtaining access to the selected media source proceed as described above with reference to FIGS. 1 and 2, with the FusionNet service provider 320 shown in FIG. 3 substituting in place of client 100 that is shown in FIGS. 1 and 2. Thus, whereas the client is the requesting party in the embodiment shown in FIGS. 1 and 2, the FusionNet service provider becomes the requesting party (on behalf of the client) for access to a media source in the variation shown in FIG. 3.

In summary, an improved client-server architecture for access to distributed media sources locatable over the Internet has been described which utilizes the advantages of known QOS networks, such as the PSTN, to provide guaranteed quality of service, security, and a charge mechanism for handling such media access requests. The architecture has the advantage of allowing access to media sources that may be independent from the QOS network provider. Where necessary, the architecture may include a gateway for transferring the requested information from the media source to the client. A variation of the architecture provides for the client functionality to be included within a service provider so that a user may obtain the benefits of the FusionNet architecture while making only a single connection to the service provider.

Claims

1. A method linking a requesting party (100) to a selected one of a plurality of media sources (150) locatable over a packet network (130) in response to a client request, characterized by the steps of:

   a. receiving information about the client and the selected media source at a server (140) over a first communications session established using the packet network (130);

   b. establishing a second communications session between the server and the selected media source (150); and

   c. transferring information over the second communications session to cause the selected media source to transfer media information,

subject to the control of the server, to the requesting party over a third communications session established using a network (120) having at least one functional quality different from a functional quality of the packet network.

2. The method according to claim 1, characterized in that the requesting party is the client.

3. The method according to claim 1 or 2, characterized in that the packet network is the Internet.

4. The method according to claim 1, characterized in that the client information includes a communication number for use in establishing the third communications session with the requesting party.

5. The method according to claim 1, characterized in that the client information includes a login and password from which the server determines a communication number for use in establishing the third communications session with the requesting party or includes an account number from which the server determines a communication number for use in establishing the third communications session with the requesting party.

6. The method according to claim 1, characterized in that the client information includes a communication number for use in billing the client or includes a login and password from which the server determines a communication number for use in billing the client or includes an account number from which the server determines a communication number for use in billing the client.

7. The method according to claim 1, further characterized by the step of before transferring information over the second communications session to cause the media source to transfer media information to the requesting party over the third communications session, interrogating the selected media source using the second communications session to determine the availability of the requested media information.

8. The method according to claim 1 or 7, characterized in that the third communications session is established among the server, the requesting party, and the selected media source.

9. The method according to claim 8, characterized in that the third communications session is established using a modem or using an ISDN adapter or using the public switched telephone network or using a private telephone network or using a high speed data transmission link or using a cable network employed for video or television.

10. The method according to claim 8 or 9, further characterized by the step of transferring additional client information from the requesting party to the server over the third communications session for use in billing the client.

11. The method according to claim 10, characterized in that the additional client information includes an account number for use in billing the client.

12. The method according to claim 1, characterized in that the third communications session is established by linking each of the selected media source and the requesting party to a gateway.

13. A system for linking a requesting party (100) to a selected one of a plurality of media sources (150) locatable over a packet network (130) in response to a client request, characterized by:

a. a server (140) for communicating with a selected media source;

b. a first interface (141) for receiving client information at the server using the packet network;

c. a second interface (143) for establishing a communications session between the server and the selected media source; and

d. a controller causing the selected media source to transfer media information to the requesting party over a communications session established using a network (120) having at least one functional quality different from a functional quality of the packet network.

14. The system according to claim 13, characterized in that the requesting party is the client.

15. The system according to claim 13, characterized in that the packet network is the Internet.

16. The system according to claim 13, characterized in that the first interface includes a mechanism used to transmit information according to the Internet Protocol or includes means responsive to the client request for identifying the selected media source in order to establish the communications session between the server and the selected media source.

17. The system according to claim 13, further characterized by a third interface for establishing a communications session among the requesting party, the server, and the selected media source.

18. The system according to claim 17, characterized in that the third interface includes a mechanism used

to establish communications session employing the public switched telephone network or employing a private telephone network or

employing a high speed data transmission link or through a cable network employed for video or television.

19. The system according to claim 17, characterized in that the third interface includes a modem or an ISDN adapter.

20. The system according to claim 13, further characterized by a gateway for linking the requesting party and the selected media source.

FIG. 1

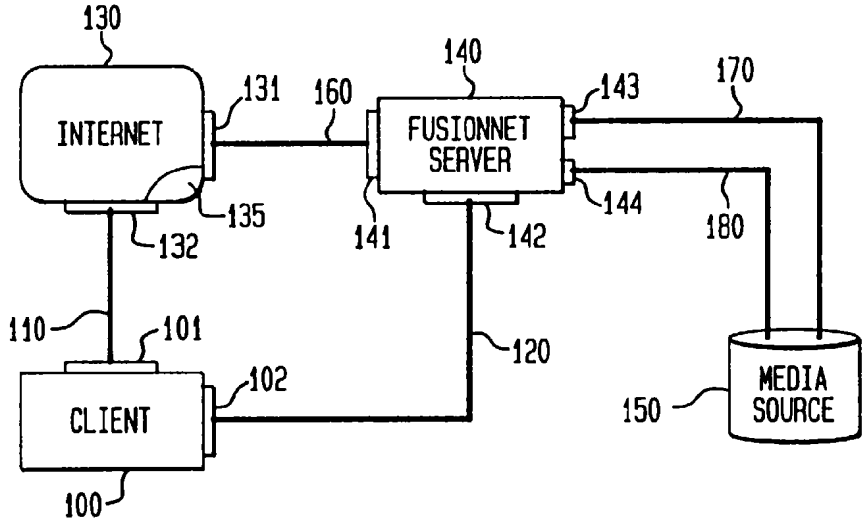


FIG. 2

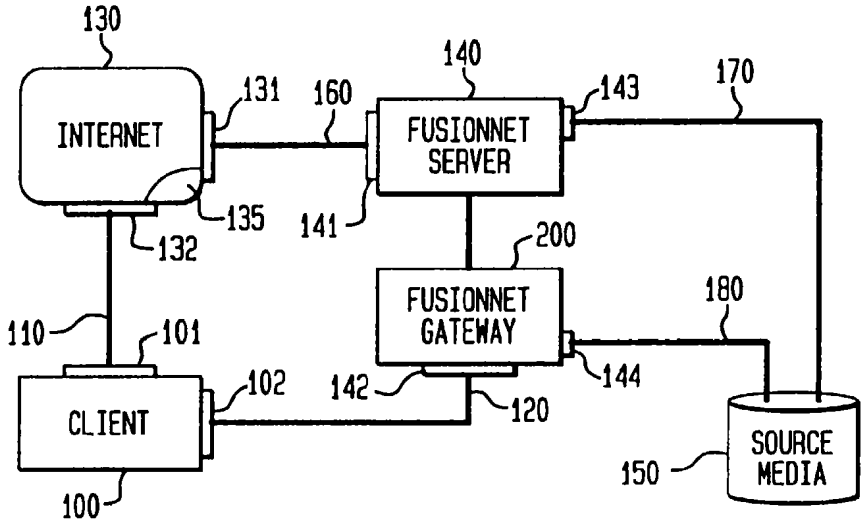
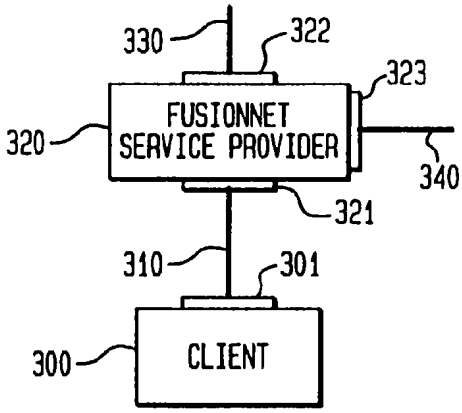


FIG. 3



(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 843 276 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
20.05.1998 Bulletin 1998/21

(51) Int. Cl.[6]: **G06K 9/20**, G06F 17/21

(21) Application number: **97304451.4**

(22) Date of filing: **24.06.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV RO SI

(30) Priority: **18.11.1996 US 751676**

(71) Applicant:
Canon Information Systems, Inc.
Costa Mesa, CA 92626 (US)

(72) Inventors:
- **Tyan, Ching-Yu**
  Irvine, California 92612 (US)
- **Huang, Hung Khei**
  Lake Forest, California 92630 (US)
- **Niki, Toru**
  Irvine, California 92614 (US)

(74) Representative:
Beresford, Keith Denis Lewis et al
BERESFORD & Co.
2-5 Warwick Court
High Holborn
London WC1R 5DJ (GB)

(54) HTML generator

(57) Automatic generation of HTML files based on bitmap image data, which faithfully preserves layout information of an original document from which the bitmap data was obtained. Generally, multi-column document layouts result in automatic generation of HTML files that use HTML "table tags" to display each of the different columns. More particularly, a bitmap image is obtained such as by scanning or retrieval of a pre-existing image, and the bitmap image is segmented into blocks. The location of each block is determined, each block is analyzed in preparation for insertion of appropriate data into an HTML file, and layout analysis is performed to identify layout relationships between the blocks based on the relative locations of the blocks in the bitmap image. Based on the layout relationships, a block type is determined for each block, column span and row span data for each block is determined, blocks are re-ordered if needed, and an HTML file is generated in which blocks are tagged as data elements in a row of an HTML "table tag" based on block type and based on column and row span information for the block.

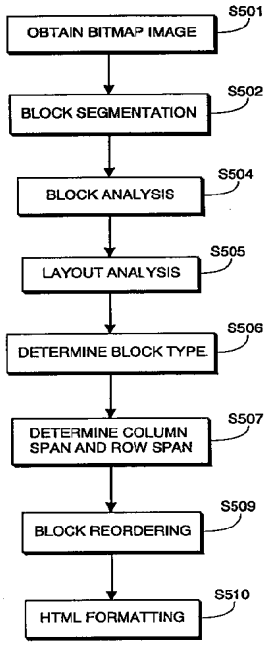


FIG. 5

EP 0 843 276 A1

Description

The present invention concerns generating an HTML file based on an input bitmap image, and is particularly directed to automatic generation of an HTML file, based on a scanned-in document image, with the HTML file in turn being used to generate a Web page that accurately reproduces the layout of the original input bitmap image.

In recent years, the popularity of the internet has grown dramatically. One reason for such growth has been the widespread adoption of HTML (HyperText Markup Language), which is a language for describing document appearance, document layout and hyperlink specifications. It defines the syntax that describe the structure and the content of a document including text, images, and other supported media. The language also provides connections among documents and other Internet resources through hypertext links and other hyperlinks. Using HTML, a Web page can be created which contains, in addition to bitmap images, graphic images, and text of various styles and sizes, hyperlinks which permit a viewer of the Web page to easily jump to another point within the page or to a completely different Web page, even one that is provided by a different server.

Once an HTML file is made available on the World Wide Web via a server, any client connected to the World Wide Web can access the page merely by typing the page address in the appropriate field of his browser. After the address has been entered, the browser requests the server to send the HTML file, which can contain text, references to graphic and bitmap image files, and formatting and hyperlink information for the entire page. Upon receipt of the HTML file, the browser automatically requests the graphic and bitmap image files referenced in the HTML file from the identified source.

To display the HTML file and the downloaded image files, the browser relies on HTML commands embedded in the HTML file. These commands are referred to as "tags". The tags indicate features or elements of a page and cause the browser to perform various functions, such as a particular type of formatting. HTML tags can be identified in HTML files by their syntax. That is, the tags are surrounded by left and right angle brackets, such as "〈P〉". In this case, "〈" indicates the start of the HTML tag, "P" is the tag itself (here a tag indicating a new text paragraph), and "〉" indicates the end of the tag. Often, tags come in pairs so as to indicate the start and end of a special function. The beginning tag initiates a feature (such as heading, bold, and so on), and the ending tag turns it off. Ending tags typically consist of the initiating tag name preceded by a forward slash (/). For example, 〈strong〉and 〈/strong〉, surrounding text, will display the surrounded text more strongly that other text. Any additional words in a tag are attributes, sometimes with an associated value after an equal sign (=), which further define or modify the tag's actions.

HTML 3.0 is presently the de-facto World Wide Web standard that defines permissible tags and nesting of tags. Approximately 100 different tags are permitted and defined.

Because of the complexity of HTML 3.0, as well as its cumbersome usage requirements, considerable effort is expended by the Web designer when authoring visually appealing and useful Web pages. For example, assume that an organization had good existing written marketing materials which it wanted to reproduce identically on a Web page. Even this seemingly simple task has typically required that a specialist spend a significant amount of time authoring HTML instructions by hand in an attempt to reproduce the layout and appearance of the written materials.

Several systems have been proposed that would automate this job of authoring HTML files from written documents. Xerox Text Bridge Pro and Caere Omni Page Pro are examples of systems which scan in written documents and generate HTML files based on the scanned-in document image. Where these systems fail is in producing HTML files that accurately represent the layout, tables and images of the original written document. In particular, a major problem has been automatically generating HTML instructions for the case where the written document is arranged in columns, or, more generally, when regions in the original document are horizontally adjacent. The term "horizontally adjacent," when used with respect to two image blocks, means a situation where the vertical extent of the two blocks overlap, or, equivalently, where a horizontal line can be found which will intersect both blocks. Similarly, the term "vertically adjacent," when used with respect to two image blocks, means a situation where the horizontal extent of the two blocks overlap, or, equivalently, where a vertical line can be found which will intersect both blocks.

A typical example of the problems associated with such systems is illustrated by reference to Figures 1 and 2. Figure 1 depicts an original printed document 10 to be converted into HTML format. As shown in Figure 1, the original document, has, among other features: title 1 in the upper left corner, subtitle 2 in the upper right corner, text columns 4, 5 and 6, picture 7 in the lower left corner, and footer 9 in the lower right corner.

Figure 2 illustrates how a Web page 20 would be displayed on display 23 by a Web browser based on the HTML file generated by an existing system for converting bitmap images into HTML. Elements in Figure 2 corresponding to those in Figure 1 are numbered similarly to those in Figure 1. Thus, after processing by the existing system, title 1 is reproduced as title 11. However, subtitle 2, rather than being reproduced in the upper right corner, is instead reproduced as element 12 in the upper left corner, just below title 11. Similarly, the entire text column structure of the original document has been eliminated, and picture 17, rather than being in the lower left corner, occupies the entire width of the page and is interposed between lines of text. Finally, footer 19 is reproduced in the lower left corner, instead of the lower right corner.

The above comparison shows the complete failure of commercially available systems to capture many layout and stylistic elements of the original document. Other known available systems also too frequently miss important layout features. Accordingly, the problems caused by the complex and cumbersome nature of HTML are not adequately addressed by commercially available systems.

It is therefore an objective of the present invention to address the foregoing problems by providing a means by which an HTML file can be automatically generated based on a bitmap image, which HTML file can be used to display a Web page which preserves layout information of the original bitmap image. In particular, according to the invention, multi-column layouts are faithfully preserved by automatic generation of HTML files that use HTML "table tags" to display columns.

According to one aspect of the invention, an HTML file is generated based on a bitmap image by obtaining two horizontally adjacent blocks in separate vertical columns of the bitmap image, and then generating an HTML file in which the blocks are placed inside table cells by being tagged as data elements in a row of an HTML tagged table.

According to another aspect of the invention, an HTML file is generated based on a bitmap image. The bitmap image is first segmented to obtain image blocks. It is then determined where in the bitmap image each of the blocks is located. Next, positional relationships between the blocks are identified based on their relative locations. Finally, an HTML file is generated. In the HTML file, the blocks are tagged as data elements in an HTML tagged table, the tags being determined in accordance with the identified positional relationships.

This brief summary has been provided so that the nature of the invention may be understood quickly. A more complete understanding of the invention can be obtained by reference to the following detailed description of the preferred embodiments thereof in connection with the attached drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 depicts an original printed document.

Figure 2 depicts an HTML page displayed by a Web browser based on an HTML file generated by a commercially available system.

Figure 3 is a perspective view showing the outward appearance of a workstation.

Figure 4 is a block diagram of the workstation depicted in Figure 3.

Figure 5 is a flow diagram illustrating a method for generating an HTML file based on a bitmap image.

Figure 6 is a diagram illustrating output of block selection analysis performed on the printed document of Figure 1.

Figure 7 is a view for explaining data output by block segmentation analysis. Figure 8 is a view for explaining output of block analysis. Figure 9 is a flow diagram for explaining layout analysis and generation of layout data based on block analysis.

Figures 10A-1 through 10A-14 are views for explaining iterative processing of layout analysis on the blocks depicted in Figure 6. Figures 10B-1 through 10B-14 are views for explaining generation of layout data for each corresponding iteration of Figures 10A-1 through 10A-14.

Figure 11A is a flow diagram illustrating identification of block type according to the present invention.

Figure 11B illustrates two cases where block reordering is required.

Figure 12 is a view for explaining output of block type determination based on layout data.

Figures 13A-13D are flow diagrams illustrating the generation of HTML instructions for each block in a bitmap image.

Figure 14 is a view showing an HTML page displayed by a Web browser based on an HTML document automatically generated in accordance with the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure 3 is a view showing the outward appearance of a computer workstation for implementing the present invention. Shown in Figure 3 is a workstation 30, such as a Macintosh or an IBM PC or PC-compatible computer having a windowing environment, such as Microsoft Windows. Provided with the workstation 30 is a display screen 31, such as a color monitor, a keyboard 32 for entering user commands, and a pointing device 34, such as a mouse, for pointing to and for manipulating objects displayed on the screen 31. The workstation 30 includes a mass storage device such as a computer disk 35 for storing data files. A scanner 37 is provided for generating a bitmap image from an input printed document.

Figure 4 is a detailed block diagram showing the internal construction of the workstation 30. As shown in Figure 4, the workstation 30 includes a central processing unit (CPU) 41 interfaced with a computer bus 42. Also interfaced with the computer bus 42 is a scanner interface 44, a display interface 45, a keyboard interface 46, a mouse interface 47, a main memory 49, and a fixed disk 35. Disk 35 stores a windowing operating system, such as Microsoft Windows, various Windows applications, an HTML Conversion Program for generating an HTML file based on a bitmap image

according to the present invention, previously scanned bitmap images, previously generated HTML files, and a Web browser for displaying HTML files. The main memory 49 interfaces with the computer bus 42 so as to provide random access memory storage for use by the CPU 41 while executing stored process steps such as those of the HTML Conversion Program. More specifically, the CPU 41 loads those process steps from the disk 35 into the main memory 49 and executes those stored process steps out of the main memory 49.

[Generation of HTML]

Figure 5 is a flow diagram comprised of process steps stored on disk 35 for generating an HTML file based on a bitmap image according to the present invention. Briefly, according to Figure 5, a bitmap image is obtained, such as by scanning or retrieval, and segmented into blocks. The location of each block is determined, each block is analyzed in preparation for insertion into an HTML file, and layout analysis is performed to identify layout relationships between the blocks based on the relative locations in the bitmap image. Based on the layout relationships, a block type is determined for each block, column span and row span for each block is determined, blocks are reordered if needed, and an HTML file is generated based on block type and column and row span information for the blocks.

[Obtaining A Bitmap Image]

More particularly, in step S501 a bitmap image is obtained. Typically, this is accomplished by scanning a printed document using scanner 37. However, the bitmap image may also be retrieved from a file stored on disk 35 or in any other manner.

[Block Identification and Segmentation]

Step S502 comprises process steps by which blocks in the image data are automatically detected and their locations with respect to the image are automatically identified. A "block" is a logically-related group of image data, such as a region of consecutive paragraphs of text image data, a region comprising title image data, or regions comprising non-text image data such as graphical image data, line drawing image data, picture data, or tabular image data. Preferably, in addition to automatic detection and identification of blocks and their locations, block segmentation according to step S502 also generates hierarchical tree data by which the logical relationship of each block with respect to other blocks is identified. For example, text image data is often found within non-text image data, such as in the case of text labels within a line-art graphic. In such a situation, the text labels would be identified in the hierarchical tree as a child node of the parent block that contains the graphic. Hierarchical Tree data such as this is useful in subsequent processing steps so as to determine the relationship of each of the blocks.

Suitable block segmentation techniques are described in the literature, and also in co-pending patent application serial number 07/873,012 filed April 24, 1992 and entitled "Method And Apparatus For Character Recognition"; and co-pending patent application serial number 08/338,781 filed November 10, 1994 entitled "Page Analysis System." The contents of these two patent applications are incorporated herein by reference as if set forth in full. Generally speaking, such block segmentation techniques work by analyzing individual pixel data within the image so as to identify connected components of the image, analyzing the connected components so as to determine whether a connected component is a text connected component or a non-text connected component, and thereafter grouping the connected components into blocks. The text and non-text blocks are then analyzed so as to determine attributes of each block. For example, in the case of text blocks, the blocks are analyzed to determine whether they are titles, plain text, captions to figures, and the like. In the case of non-text blocks, the blocks are analyzed to determine whether they are line-art images, half tone images, color images, tabularly arranged data, and the like. At the same time, a hierarchical tree is generated showing, through parent-child nodal relationships, the relationships of the text and non-text blocks within the image data.

Figure 6 is an illustration of the results of block segmentation according to step S502. As shown in Figure 6, and with respect to original image data illustrated in Figure 1, a first block 51 (BLK1) corresponds to the first line of title data of Figure 1. Likewise, a second block 52 (BLK2) corresponds to the second line of title-data of Figure 1. A fifth block 55 (BLK5) corresponds to the first line of the subtitle. An eighth block 58 (BLK8) corresponds to the large "S" in the article's caption, whereas a ninth block 59 (BLK9) corresponds to the remainder of the text in the article's caption. Blocks 53 (BLK10, BLK11, BLK14 and BLK15) correspond to text in the article itself. Block 54 (BLK12) corresponds to the half-tone image in the article whereas block 56 (BLK13) corresponds to the textual caption for that image. Other blocks shown in Figure 6 have meanings that can readily be appreciated from the foregoing.

Figure 7 is a view for explaining data that is stored for each block, including attribute data and hierarchical tree structure data. Specifically, and as shown in Figure 7, for each block, block data 150 is stored, and includes at least block number identification 151, coordinate information 152 (such as upper-left and lower-right coordinates of a circumscribing rectangle for the block), attribute data 153 which stores whether the block is text or non-text, together with any

other attributes determined during block segmentation, and hierarchical tree structure 154 which includes pointers to parent and child nodes for the block.

Summarizing, at the end of block segmentation in step S502, image data has been analyzed so as to automatically detect blocks of logically-related image data and to automatically identify locations of those blocks. In addition, hierarchical tree structure showing the logical relation from one block to another has been stored for each such block.

[Block Analysis]

Block analysis in step S504 comprises process steps by which each of the blocks identified in step S502 is analyzed so as to determine whether it is a block that affects document layout, and, if so, to extract needed information for HTML generation.

Specifically, for each block identified in step S502, block analysis according to step S504 determines whether the block affects document layout by inspecting the hierarchical tree to determine whether the block is a text block that is not a child of a non-text block. The purpose of this test is to eliminate from further processing any text block that is within a non-text block, inasmuch as such a text block does not affect document layout. For example, a textual label within a line-art graphic does not affect document layout; rather, it is the histogram that affects document layout. Accordingly, by inspecting the hierarchical tree generated in step S502, it is possible to distinguish between blocks that affect document layout from blocks that do not affect document layout.

On the other hand, non-text blocks, such as line-art drawings, half-tone images and tabular data, presumptively affect document layout and are therefore targeted for further analysis in step S504.

In the context of the Figure 6 example, it will be appreciated that all blocks shown there affect document layout. Accordingly, no blocks are eliminated from further processing. This would not be the case, for example, in the situation where a document image included textual data within an image or a table. In those situations, the block containing such textual data would be eliminated from further processing, since such a text block does not affect document layout but rather the image block affects document layout.

All blocks so identified as affecting document layout are thereafter processed in accordance with image attributes 153 in preparation for HTML generation. Specifically, if image attribute 153 identifies the block as non-text data such as a half-tone image, then block analysis in step S504 generates a suitable image file for the image. Preferably, the image file is stored in a commonly used Web browser format, such as .GIF or .JPEG format. On the other hand, for text-type blocks, the block is subjected to optical character recognition analysis so as to obtain a file of computer readable character codes such as ASCII character codes for subsequent use in HTML generation, and also to determine the number of text lines in each text block.

Figure 8 is a view for explaining the output of block analysis in step S504. Particularly, as shown in Figure 8, for each block that affects document layout, data 160 is stored containing at least block number 161, coordinate data 162, attribute data 163 which is the same attribute data as attribute data 153 of Figure 7, hierarchical tree data 164 which is the same as hierarchical tree data 154 of Figure 7 with the exception that child node information may be omitted, and block analysis results 165. Block analysis results 165 may include, for example, an image file name in the case where attribute data 163 indicates that the block is non-text data, or may include optical character recognition results and number of text lines in each text block in the case where attribute data 163 indicates that the block is text-type data.

Summarizing, block analysis according to step S504 results in identification of those blocks that affect document layout, and for each of those blocks, an image file or optical character recognition results.

[Layout Analysis]

Step S505 of Figure 5 comprises process steps by which the blocks identified in step S504 are analyzed so as to provide layout data for the document. These process steps are illustrated in Figure 9. Generally speaking, the process steps of Figure 9 test each possible combination of pairs of vertically arranged blocks or groupings of blocks to determine whether the smallest rectangle which circumscribes the pair of blocks overlaps onto any other blocks, and if the combination does not overlap, the pair is combined into a single grouping. Likewise, each possible combination of pairs of horizontally arranged blocks or groupings of blocks are tested to determine whether the combination overlaps onto any other blocks, and if there is no overlap the pair is combined into a horizontal grouping. Vertical combinations are tested first since most english language documents are arranged into vertical layouts; accordingly, in situations where it can be expected for documents to be arranged in horizontal layouts, such as in some Japanese language documents, horizontal groupings should be tested first. The process of testing vertical groupings and then horizontal groupings is repeated until there are no further groupings that can be made and the entire document has been combined into a single grouping, whereupon the layout data is complete.

Figure 9 will now be explained in more detail, in connection with the illustrative example of Figure 10 which is based on the document of Figure 1. Thus, in step S901, a combination of two vertically arranged "groupings" is first tested.

Groupings are either individual blocks or blocks that have previously been combined into a single grouping. "Vertically-arranged" groupings are groupings that differ in location vertically; thus, groupings that do not differ in the vertical direction, as measured based on coordinate data 162, are not tested in this step S901.

An illustrative example of a first such combination is shown in Figure 10A-1. As shown in Figure 10A-1, blocks BLK1 and BLK2 are vertically-arranged in the sense that they differ in vertical locations. A dotted line 171 represents a circumscribing rectangle that encloses these two blocks. It is this combination as represented by the circumscribing rectangle that is tested in step S902. For illustrative purposes, a small gap is shown between the rectangle and the enclosed blocks. However, it should be noted that the actual grouping border will generally coincide with the block borders.

Specifically, step S902 tests to determine whether the combination of vertically-arranged groupings overlaps onto any other grouping. A combination would overlap onto another grouping if, when making the combination, the combination would include all or any part of any other block or grouping. If the combination does not overlap onto any other grouping, then flow branches to step S903 in which the groupings are combined vertically into a vertical grouping. On the other hand, if the combination does overlap onto any other grouping, then flow advances directly to step S904 which determines whether there are any more vertical combinations that need testing. If more vertical combinations need testing, then flow returns to step S901 until all vertically-arranged groupings are suitably tested.

Thus, reverting to Figure 10A-1, since circumscribing rectangle 171 does not overlap onto any other groupings (step S902), blocks BLK1 and BLK2 are combined vertically into a single vertical grouping (step S903). This is illustrated graphically in Figure 10B-1 which shows layout data depicting the combination of blocks BLK1 and BLK2 into a single vertical grouping.

Referring now to Figure 10A-2, since more vertical combinations exist (step S904), those vertical combinations are tested. Thus, in Figure 10A-2, a circumscribing rectangle 172 represents the combination of the vertical grouping that is comprised of blocks BLK1 and BLK2 as well as vertically-arranged block BLK3. Since the combination does not overlap onto any other grouping, the combination is combined vertically into a new grouping as illustrated in the layout data of Figure 10B-2.

In Figure 10A-3, block BLK4 is not tested since, based on coordinate data 163, it is not a vertically-arranged grouping. Rather, the next block that is tested is block BLK5 since it is a vertically-arranged grouping. The circumscribing rectangle for the combination is illustrated with dotted line 173. As can be appreciated, the combination includes block BLK4. Thus, the combination overlaps onto another grouping, and a vertical combination is not made. Thus, as illustrated in Figure 10B-3, the layout remains the same.

As can readily be appreciated, any other combination of the vertical grouping of blocks BLK1, BLK2 and BLK3, when combined with any other block of the document, will result in an overlap onto some other grouping. Accordingly, when each test is made in step S901, step S902 will determine that the combination overlaps onto some other grouping, and no further vertical combinations will be made with this grouping.

Referring now to Figure 10A-4, combinations of vertical groupings with the next permissible block or grouping, here block BLK4, are now made. Circumscribing rectangle 174, which shows a combination of block BLK4 with block BLK5, clearly shows that an overlap exists with block BLK3. Thus, since the combination overlaps onto another grouping, a vertical combination is not made and the groupings remain as shown in the layout data of Figure 10B-4. Likewise, it can be understood that combinations with any other block with BLK4 will result in an overlap, and no further vertical combinations are made.

Referring now to Figure 10A-5, combinations of vertically-arranged groupings with block BLK5 are tested. Dotted line 175 represents a circumscribing rectangle for the combination of blocks BLK5 and BLK6. Since no overlaps onto any other grouping exist, blocks BLK5 and BLK6 are combined vertically as illustrated in the layout data of Figure 10B-5.

Likewise, in Figure 10A-6, the grouping of blocks BLK5 and BLK6 are then tested in combination with block BLK7, as illustrated by circumscribing rectangle 176. Since the combination does not overlap onto any other grouping, the groupings are combined vertically as illustrated in the layout data of Figure 10B-6.

The process of testing vertical combinations of groupings continues until there are no more permissible vertical combinations that can be tested. In connection with this testing, it will be appreciated that the only further groupings that can be combined are blocks BLK12 and BLK13 and blocks BLK14 and BLK15. This situation is illustrated in Figures 10A-7 and 10B-7.

Since no further vertical combinations are available for testing, flow then advances to step S905, in which similar combinations and testing of combinations are made in the horizontal direction. Thus, step S905 tests combinations of two horizontally-arranged groupings, step S906 determines whether the combination overlaps onto any other groupings, step S907 combines groupings horizontally in a case where step S906 determines that the combination does not overlap them to any other grouping, and step S908 determines whether there are any more horizontal combinations that need testing.

Specifically, reverting to Figure 10A-8, the grouping of blocks BLK1, BLK2 and BLK3 is tested in combination with

horizontally-arranged block BLK4. As seen in Figure 10A-8, circumscribing rectangle 177 does not overlap onto any other groupings and the combination is therefore made. This is illustrated graphically in the layout data of Figure 10B-8.

No other tests are made with respect to the new horizontal grouping of blocks BLK1 through BLK4 since, as will be appreciated, no other blocks or grouping of blocks is horizontally arranged. Likewise, no block or grouping of blocks is horizontally arranged with the vertical grouping of blocks BLK5 through BLK7, and no testing for these blocks is made.

Accordingly, the next block tested for combinations of horizontally arranged grouping is block BLK8 and the first such combination is with block BLK9. Inasmuch as circumscribing rectangle 178 does not overlap onto any other block, the horizontal combination is made as illustrated in the layout data of Figure 10B-9.

The new grouping of blocks BLK8 and BLK9 is not tested with block BLK10 because block BLK10 is not horizontally arranged. Block BLK11 is horizontally arranged, but as shown in Figure 10A-10, horizontal combination of the two overlaps onto BLK10 and accordingly no combination is made. Accordingly, the groupings remain as shown in the layout data of Figure 10B-10.

All other combinations of horizontally-arranged groupings also overlap onto other blocks and/or groupings. Accordingly, when formed in step S905 and when tested in step S906, no further horizontal combinations are made.

Flow then advances to step S909 to determine whether all blocks in the image have been incorporated into a single overall group. So long as blocks remain not grouped into one overall grouping, the process steps of S901 through S908 are iterated until all blocks are contained within one grouping.

Thus, as illustrated with respect to Figure 10A-11, the horizontal grouping containing blocks BLK1 through BLK4 is tested in combination with the vertical grouping containing the blocks BLK4 through BLK6, as illustrated by circumscribing rectangle 179. Since no blocks overlap, a combination is made vertically as illustrated in the layout data of Figure 10B-11.

Likewise, in Figure 10A-12, the horizontal grouping containing blocks BLK8 and BLK9 is tested with respect to block BLK10, as illustrated by circumscribing rectangle 180. Since no blocks are overlapped, a vertical grouping is made as illustrated in the layout data of Figure 10B-12.

In this iteration of steps S901 through S904, no further vertical combinations are overlap-free. Accordingly, no further vertical combinations are made.

Figure 10A-13 illustrates that, with respect to this iteration of horizontal steps S905 through S908, one further horizontal grouping can be made as illustrated at circumscribing rectangle 181. Accordingly, the groupings are as shown at Figure 10A-13 and in the layout data of Figure 10B-13.

Further vertical and horizontal iterations are made of steps S901 through S908, resulting in a final situation that is shown at Figures 10A-14 and 10B-14. It will be appreciated that the document has been arranged into one overall grouping, in this case a vertical grouping, which encompasses the entire document. Layout analysis thereupon concludes with storage of the layout data, together with storage of coordinates that identify circumscribing rectangles for each and every block or grouping of blocks and preferably also storage of width and height of each block. The layout data that is stored for the document of Figure 1 is reproduced below as an illustrative example:

```
VRT: (l, t, r, b, w, h) = (48, 72, 2247, 3167, 2200, 3096)
  VRT: (l, t, r, b, w, h) = (48, 72, 2247, 763, 2200, 692)
    HOZ: (l, t, r, b, w, h) = (52, 72, 2247, 423, 2196, 352)
      VRT: (l, t, r, b, w, h) = (52, 72, 1311, 415, 1260, 344)
        BLK1: (l, t, r, b, w, h) = (52, 72, 1311, 187, 1260, 116)
        BLK2: (l, t, r, b, w, h) = (56, 192, 827, 295, 772, 104)
        BLK3: (l, t, r, b, w, h) = (52, 316, 679, 415, 628, 100)
      BLK4: (l, t, r, b, w, h) = (1520, 380, 2247, 423, 728, 44)
    VRT: (l, t, r, b, w, h) = (48, 524, 2247, 763, 2200, 240)
      BLK5: (l, t, r, b, w, h) = (52, 524, 2247, 587, 2196, 64)
      BLK6: (l, t, r, b, w, h) = (48, 612, 2247, 675, 2200, 64)
      BLK7: (l, t, r, b, w, h) = (52, 696, 2091, 763, 2040, 68)
  HOZ: (l, t, r, b, w, h) = (48, 896, 2247, 3167, 2200, 2272)
    VRT: (l, t, r, b, w, h) = (48, 896, 1487, 3159, 1440, 2264)
      HOZ: (l, t, r, b, w, h) = (48, 896, 1487, 1819, 1440, 924)
        VRT: (l, t, r, b, w, h) = (48, 896, 727, 1815, 680, 920)
          HOZ: (l, t, r, b, w, h) = (56, 896, 551, 1059, 496, 164)
            BLK8: (l, t, r, b, w, h) = (56, 908, 131, 1035, 76, 128)
            BLK9: (l, t, r, b, w, h) = (144, 896, 551, 1059, 408, 164)
          BLK10: (l, t, r, b, w, h) = (48, 1084, 727, 1815, 680, 732)
        BLK11: (l, t, r, b, w, h) = (812, 900, 1487, 1819, 676, 920)
      VRT: (l, t, r, b, w, h) = (48, 1888, 1487, 3159, 1440, 1272)
        BLK12: (l, t, r, b, w, h) = (48, 1888, 1487, 2919, 1440, 1032)
        BLK13: (l, t, r, b, w, h) = (48, 2940, 1427, 3159, 1380, 220)
    VRT: (l, t, r, b, w, h) = (1568, 904, 2247, 3167, 680, 2264)
      BLK14: (l, t, r, b, w, h) = (1572, 904, 2247, 2499, 676, 1596)
      BLK15: (l, t, r, b, w, h) = (1568, 2524, 2247, 3167, 680, 644)
```

wherein "l", "t", "r" and "b" refer to left, top, right and bottom block coordinates, and wherein "w" and "h" refer to the width and height of the corresponding block.

Also in this step the block order is determined according to the order of the blocks in the layout data when read from top to bottom. Thus, in the present example the block order is: BLK1, BLK2, BLK3, BLK4, BLK5, BLK6, BLK7, BLK8, BLK9, BLK10, BLK11, BLK12, BLK13, BLK14, BLK15.

[Determining Block Type]

Step S506 shown in Figure 5 will now be discussed in more detail with reference to Figure 11. Generally according to Figure 11A, a type is determined for each block by successively performing a series of tests and either designating the block type when a test is first passed (the designation depending upon which test was passed) or designating the default block type in the event that none of the tests is passed.

More particularly, in step S1101 it is determined whether the current block is the first block encountered. It is also determined whether the current block is included, somewhere up the tree structure, in a horizontal grouping, and if so, whether it is the first block under its highest-level horizontal grouping. If either of the foregoing are answered affirmatively, then in step S1102 the block is designated "single column". If not, processing proceeds to step S1104.

In step S1104, a four-part test is applied. First, it is determined whether the previous block is a non-text image. Second, a determination is made whether the previous block's left edge is to the left of the current block's left edge. For purposes of describing these tests, the "current" block refers to the block for which a type presently is being determined, and the "previous" block refers to the block immediately preceding the current block in the ordered list identified in step S505.

The third part of the step S1104 test is to determine whether the right edge of the current block is close to the left edge of the next block. In the preferred embodiment, the answer to this part is yes only if the distance between the two is less than 0.01 times the width of the entire image (i.e., including all blocks). Fourth, it is determined whether the sum of the width of the current block and the next block is less than a fixed threshold. In the preferred embodiment, the threshold is equal to 0.6 times the width of the entire image. If each of the foregoing tests is answered affirmatively, then in step S1105 the block is designated as "combined column". If not, processing proceeds to step S1106.

In step S1106, a three-part test is performed. In the first part, it is determined whether the previous block's left edge is left of the current block's left edge. The second part determines whether the previous block is horizontally separated from the current block, that is, whether there is a non-zero horizontal distance between the right edge of the previous block and the left edge of the current block. The third part determines whether the current block is at least a second

child of a horizontal grouping. This last part determines whether the current block is part of a horizontal grouping and is listed at least second in the hierarchical chart behind an element with which it is horizontally grouped, for example, (or is the first block in a grouping that is at least the second child of the horizontal grouping). If the foregoing tests are each answered affirmatively, then in step S1107 block is designated as multiple column. Otherwise, processing proceeds to step S1109.

In step S1109, a three-part test is applied. First, it is determined whether the previous block's top edge is vertically higher than the current block's top edge, that is, whether the y-coordinate of the top edge of the previous block is greater than the y-coordinate of the top edge of the current block. Second, it is determined whether the current block and the next block are vertically separated, that is, whether there is a non-zero vertical distance between the bottom of the previous block and the top of the current block. Third, it is determined whether the current block is included, somewhere up the tree structure, in a horizontal grouping, or whether the current block is vertically grouped with the previous block. If all parts of the foregoing test are answered affirmatively, then in step S1110 the block is designated as "joint column". Otherwise, in step S111 the block is designated as "single column".

The application of the processing steps shown in Figure 11 to the present example will now be discussed. Specifically, BLK1 is the first block, and therefore in steps S1101 and S1102 is designated as single column.

BLK2 is not the first block, nor is it the first block under its highest-level horizontal grouping, so the test specified in step S1101 fails, and the test specified is step S1104 is applied. In this case, the first part fails because BLK1 is not a non-text image. The test set forth at step S1106 is applied next. Here, the first part of the test also fails because the left edge of the previous block is not to the left of the current block's left edge. Accordingly, the test described in step S1109 is applied next. Here, BLK1's top edge is higher than BLK2's edge, vertical separation exists between the bottom edge of BLK1 and the top edge of BLK2, and BLK2 is vertically grouped with BLK1. Accordingly, the test in S1109 is satisfied and BLK2 is designated as joint column.

The analysis of BLK3 is identical to that of BLK2 and accordingly BLK3 will be designated as joint column also.

BLK4 is not the first block, nor is it the first block under its highest-level horizontal grouping, and BLK3 is not a non-text image. Accordingly, the tests in steps S1101 and S1104 fail, and the test in step S1106 is applied. Here, BLK3's left edge is left of BLK4's left edge, horizontal separation exists between the right edge of BLK3 and the left edge of BLK4, and by reference to the layout data set forth above, it can be seen that BLK4 is the second child of a horizontal grouping. Specifically, BLK4 is horizontally grouped with the vertical group consisting of blocks 1, 2 and 3. Moreover, BLK4 is shown in the layout data as being below that vertical grouping. Accordingly, the test in step S1006 is satisfied, and BLK4 is designated as multiple column.

BLK5 is not the first block, nor is it the first block under its highest-level horizontal grouping, and BLK4 is not a non-text image. Therefore, processing proceeds to step S1106. The left edge of BLK4 is not left of the left edge of BLK5. Accordingly, this test fails. With respect to the test at step S1109, the top edge of BLK4 is higher than the top edge of BLK5 and vertical separation exists between the two blocks. However, BLK5 is not under a horizontal grouping and is not vertically grouped with BLK4. Therefore, this test fails as well. Accordingly, BLK5 is designated as single column.

The analysis of blocks 6 and 7 is the same as that of BLK2, and accordingly, each such block is designated as joint column.

BLK8 is not the first block, but it is the first block of its highest-level horizontal grouping. Therefore, BLK8 is designated as single column.

BLK9 is not the first block, nor is it the first block under its first horizontal grouping. However, in step S1104, the previous block is a non-text image, the previous block's left edge is left of the current block's left edge, the two blocks are horizontally close, and the sum of the widths of the two blocks is less than 0.60 times the width of the entire image. Accordingly, BLK9 is designated combined column.

Similarly, BLK10, BLK12, BLK13 and BLK15 are designated as joint column, since the test in step S1109 is satisfied. BLK11 and BLK14 are designated as multiple column, since the test in step S1106 is satisfied.

Thus, each of the blocks is designated as set forth below:

BLK1: (SINGLE_COL)
BLK2: (JOINT_COL)
BLK3: (JOINT_COL)
BLK4: (MULTIPLE_COL)
BLK5: (SINGLE_COL)
BLK6: (JOINT_COL)
BLK7: (JOINT_COL)
BLK8: (SINGLE_COL)
BLK9: (COMBINED_COL)
BLK10: (JOINT_COL)
BLK11: (MULTIPLE_COL)

BLK12: (JOINT_COL)
BLK13: (JOINT_COL)
BLK14: (MULTIPLE_COL)
BLK15: (JOINT_COL)

An illustration of these classifications is shown in Figure 12.

[Determining Column Span and Row Span]

Step S507 will now be discussed in detail. Blocks having column span and row span are located by finding a nested horizontal-vertical-horizontal (parent-child-grandchild) structure that does not include any great-grandchild having been designated combined column in the layout analysis generated in step S505. To facilitate the following discussion, each element in the layout analysis will be referenced in relation to the higher level horizontal grouping in the nested horizontal-vertical-horizontal (parent-child-grandchild) genetic hierarchical structure, using tree structure terminology.

Once a nested horizontal-vertical-horizontal hierarchical structure without a combined column great-grandchild is located, the determination of row span and column span proceeds as follows. The column span number is determined to be the number of children of the horizontal grandchild. The block to which the column span number is assigned is selected to be the first non-horizontal grandchild that has a horizontal sibling, or the first great-grandchild whose parent has a horizontal sibling. The row span number is determined to be the number of horizontal grandchildren plus one. The block to which the row span number is assigned is selected to be the first non-vertical child, or the first grandchild that does not have a horizontal sibling. If a grouping is selected to be assigned the column span number or the row span number, the assignment passes to the first block in that grouping.

The above process is best illustrated by way of example. The structural portion of the layout analysis for the document shown in Figure 1 is reproduced below, with groupings and blocks numbered (beginning with number 101) for ease of reference.

```
101  VRT
102    VRT
103      HOZ
104        VRT
105          BLK1
106          BLK2
107          BLK3
108        BLK4
109      VRT
110        BLK5
111        BLK6
112        BLK7
113    HOZ
114      VRT
115        HOZ
116          VRT
117            HOZ
118              BLK8
119              BLK9
120            BLK10
121          BLK11
122        VRT
123          BLK12
124          BLK13
125      VRT
126        BLK14
127        BLK15
```

Since horizontal-vertical-horizontal (parent-child-grandchild) structure numbers 115-117 has a great-grandchild with a combined column designation (BLK9), the only horizontal-vertical-horizontal structure without a combined column great-grandchild in the layout analysis consists of numbers 113-115. The horizontal grandchild is numbered 115, and its only children are 116 and 121. Therefore, the column span number is two. The grandchildren of horizontal grouping 113 are horizontal grouping 115 and vertical grouping 122. Because BLK12 (123) is the first great-grandchild

whose parent (122) has a horizontal sibling (115), BLK12 is assigned the column span number.

With respect to row span, there is only one horizontal grandchild, grouping 115. Therefore, the row span number is equal to two. Because BLK14 (126) is the first grandchild that does not have a horizontal sibling, BLK14 is assigned the row span number.

Consequently, BLK12 is assigned a column span number of 2 and BLK14 is assigned a row span number of 2.

[Block Reordering]

Step S509 will now be discussed with reference to Figure 11B. As shown in Figure 11B, two general situations might occur in which the block order obtained in step S505 will have to be rearranged to accommodate the way HTML processes table data (i.e., down row by row, in sequence). In each, a row span block is to the right of and horizontally adjacent to a column span block.

Referring to Figure 11B, the first case exists when blocks are arranged such as blocks 141-144, with the top edge of row span block 144 above the top edge of column span block 143, and step S505 having ordered the group of blocks sequentially as: 141, 142, 143 and 144. In this case, the order of blocks 143 and 144 are switched, resulting in a final order of: 141, 142, 144 and 143.

Similarly, the second case exists when blocks are arranged such as blocks 146-149, with the top edge of row span block 149 above the top edges of blocks 147 and 148, and step S505 having ordered the group of blocks sequentially as: 146, 147, 148 and 149. In this case, the order of block 149 is moved up in position to be placed before block 147, resulting in a final order of: 146, 149, 147 and 148.

The foregoing are merely illustrative of the two generic cases. It should be noted that the number of blocks might vary and the blocks shown might be representative of more than one block. The key elements of this reordering step are: (1) the existence of a row span block to the right of and horizontally adjacent to a column span block, and (2) whether the column span block is on the first or second row that the row span block spans. Moreover, although the foregoing describes the two cases where the row span is two rows, the concept can be easily extended to row spans of more than two.

Applying the foregoing to the present example concerning the document shown in Figure 1, a row break occurs between BLK10 and BLK12 and then another after BLK12. Accordingly, this presents an example of the first case described above, with BLK10 being the equivalent of block 141, BLK11 being the equivalent of block 142, BLK12 (the column span block) and BLK13 together being the equivalent of block 143, and BLK14 (the row span block) and BLK15 together being the equivalent of block 144. Accordingly, BLK14 and BLK15 are moved up in order before BLK12 and BLK13. The final order is thus: BLK1, BLK2, BLK3, BLK4, BLK5, BLK6, BLK7, BLK8, BLK9, BLK10, BLK11, BLK14, BLK15, BLK12, BLK13.

[HTML Formatting]

Step S510, shown in Figure 5, will now be discussed in detail with reference to Figures 13A through 13D. Generally according to Figure 13, first an HTML header is inserted. Then for each block, processing is performed in accordance with the block type determined in step S506, a different set of processing steps being assigned to each block type. Finally, after all blocks have been processed, end-of-file codes are inserted.

More particularly, in step S1301 the following HTML header codes are inserted: "⟨HTML⟩⟨HEAD⟩⟨TITLE⟩title⟨/TITLE⟩⟨/HEAD⟩⟨BODY⟩⟨BODY GCOLOR="#FFFFFF"⟩⟨CENTER⟩", where "title" is the title of the file.

In step S1302, the first block is selected.

In step S1304, it is determined whether the block type for the current block is single column. If it is, then single column processing is performed in step S1305 in accordance with steps shown in Figure 13B, and then processing proceeds to step S1313. If not, processing proceeds to step S1306.

In step S1306, it is determined whether the block type is multiple column. If it is, then multiple column processing is performed in step S1307 in accordance with the steps shown in Figure 13C, and then processing proceeds to step S1313. If not, processing proceeds to step S1309.

In step S1309, it is determined whether the block type is joint column. If it is, then joint column processing is performed in step S1310 in accordance with the steps shown in Figure 13D, and then processing proceeds to step S1313. If not, then the block type must be combined column. Because no additional codes are inserted for combined column, processing simply proceeds immediately to step S1313.

In step S1313 the block is inserted into the HTML file. Typically, either the contents of the block are inserted into the HTML file (for text blocks) or a reference to file containing the relevant data (for graphics and bitmap images) is inserted.

In step S1311, it is determined whether the current block is the last block. If it is not, the next block is selected in step S1312 and processing proceeds to step S1304. If the current block is the last block, then in step S1314 the follow-

ing end-of-file HTML codes are inserted: "⟨/TABLE⟩⟨/CENTER⟩⟨/BODY⟩⟨/HTML⟩".

Single column processing of step S1305 will now be discussed in more detail with reference to Figure 13B. Specifically, in step S1315 it is determined whether the file is currently in a table. If it is, then in step S1316 the end HTML table code ("⟨/TABLE⟩") is inserted, and processing proceeds to step S1317. Otherwise, processing proceeds to step S1317 without inserting that code.

In step S1317, the following table definition code is inserted into the file: "⟨TABLE BORDER=0 CELLSPACING=0 CELLPADDING=7 WIDTH=w⟩", where BORDER, CELLSPACING, CELLPADDING, and WIDTH are attributes of the ⟨TABLE⟩ tag, and where w is equal to the width of the entire image times 72 and divided by the number of dots per inch in the image.

In step S1319, it is determined whether row span for the block (which was determined in step S507) is greater than one. If it is, the following code is inserted into the HTML file: "⟨TR⟩⟨TD VALIGN=TOP ROWSPAN=n⟩", where n is the row span number determined in step S507.

Otherwise, it is determined in step S1321 whether column span is greater than one. If it is, then in step S1322 the following HTML code is inserted into the HTML file: "⟨TR⟩⟨TD VALIGN=TOP COLSPAN=m⟩", where m is the column span number determined in step S507. Otherwise, in step S1324 the following code is inserted into the HTML file: "⟨TR⟩⟨TD VALIGN=TOP⟩".

Multiple column processing of step S1307 will now be discussed in more detail with reference to Figure 13C. Specifically, in step S1325 it is determined whether row span is greater than one. If it is, then the following HTML code is inserted: "⟨TD VALIGN=TOP ROWSPAN=n⟩". Otherwise, in step S1327 it is determined whether column span for the current block is greater than one. If it is, then in step S1329 the following HTML code is inserted into the HTML file: "⟨TD VALIGN=TOP COLSPAN=m⟩". Otherwise, in step S1330 the following HTML code is inserted: "⟨TD VALIGN=TOP⟩".

Joint column processing of step S1310 will now be discussed in detail with reference to Figure 13D. Specifically, in step S1331 it is determined whether the row span of the current block is greater than one. If it is, then in step S1332 the following HTML code is inserted: "⟨TR⟩⟨TD VALIGN=TOP ROWSPAN=n⟩". Otherwise, it is determined in step S1334 whether the column span for the current block is greater than one. If it is, then in step S1335 the following HTML code is inserted: "⟨TR⟩⟨TD VALIGN=TOP COLSPAN=m⟩". Otherwise, in step S1336 it is determined whether the number of lines in the block (which was determined in step S504 is one. If it is, then in step S1337 the following HTML code is inserted: "⟨BR⟩". Otherwise, in step S1339 the following HTML code is inserted: "⟨P⟩".

Step S510 is now applied to the present example. First, the HTML header is inserted as shown in step S1301. Next, each of blocks 1 through 15 is consecutively processed to construct the body of the HTML file.

Specifically, BLK1 is single column with no row span or column span. Accordingly, the HTML code is specified in steps S1317 and S1324. Immediately after the insertion of that code, BLK1 (or a reference thereto) is inserted into the HTML file. BLK2 is joint column with no column span or row span and containing one line of text. Accordingly, the ⟨BR⟩ code is inserted and immediately followed by BLK2. BLK3 has the same classification as BLK2 and thus is preceded by the same HTML code. BLK4 is multiple column with no row span or column span. Accordingly, the HTML code is specified by step

S1330. Proceeding in this manner, combined column BLK9 is eventually reached. Accordingly, no HTML code precedes BLK9. Proceeding even further, BLK14, which is multiple column with a row span of 2, is reached. Accordingly, the HTML code is specified by step S1326. Similarly, BLK12 is joint column with a column span of 2. Accordingly, the HTML code is specified by step S1335.

When all blocks have been processed, the end of file codes specified in step S1314 are inserted, and the HTML file is complete. The resulting code is set forth below:

```
<HTML>
<HEAD>
<TITLE>Earth1</TITLE>
</HEAD>
<BODY BGCOLOR = "#FFFFFF">
<CENTER>
<TABLE BORDER = 0 CELLSPACING = 0 CELLPADDING = 7 WIDTH = 738>
<TR><TD VALIGN = TOP> BLK1
                      <BR> BLK2
                      <BR> BLK3
    <TD VALIGN = TOP> BLK4
</TABLE>
<TABLE BORDER = 0 CELLSPACING = 0 CELLPADDING = 7 WIDTH = 738>
<TR><TD VALIGN = TOP> BLK5
                      <BR> BLK6
                      <BR> BLK7
</TABLE>
<TABLE BORDER = 0 CELLSPACING = 0 CELLPADDING = 7 WIDTH = 738>
<TR><TD VALIGN = TOP> BLK8
                           BLK9
                       <P> BLK10
    <TD VALIGN = TOP> BLK11
    <TD VALIGN = TOP ROWSPAN = 2> BLK14
                       <P> BLK15
<TR><TD VALIGN = TOP COLSPAN = 2> BLK12
                       <P> BLK13
</TABLE>
</CENTER>
</BODY>
</HTML>
```

Figure 14 illustrates the appearance of a Web page generated in accordance with the above HTML code. As can be seen from Figure 14, title 201 is properly placed in the upper-left corner, and article type 202 is placed very close to the upper-right corner. Columns 204, 205 and 206 have been retained and appear very similar to the layout of the original document. Also, footer 209 is in the lower-right corner and picture 207 is in the lower-left corner underneath columns 204 and 205 and to the left of column 206.

Thus, the HTML file generated can be used to generate a page which closely follows the layout presented in the original document.

The invention has been described with respect to particular illustrative embodiments. It is to be understood that the invention is not limited to the above described embodiments and modifications thereto, and that various changes and modifications may be made by those of ordinary skill in the art without departing from the spirit and scope of the appended claims.

Claims

1. A method for generating an HTML file based on a bitmap image, comprising:

   obtaining two horizontally adjacent image blocks in separate vertical columns of the bitmap image; and
   generating an HTML file in which the blocks are tagged as data elements in a row of an HTML tagged table.

2. A method according to Claim 1, wherein an HTML code is assigned to each block in accordance with a position of the block relative to other elements within the bitmap image.

3. A method for generating an HTML file based on a bitmap image, comprising:

   a segmenting step of segmenting the bitmap image to obtain image blocks;
   a determining step of determining where in the bitmap image each of the blocks is located;
   an identifying step of identifying positional relationships between the blocks based on their relative locations; and

a generating step of generating an HTML file in which the blocks are tagged as being data elements in an HTML tagged table, the tags being determined in accordance with the identified positional relationships.

4. A method according to Claim 3, wherein the identifying step includes forming vertical and horizontal groupings of the blocks.

5. A method according to claim 3, wherein the identifying step includes forming a hierarchical structure.

6. A method according to Claim 5, wherein the step of forming the hierarchical structure includes forming horizontal and vertical groupings of elements, the elements consisting of previously formed groupings and ungrouped blocks.

7. A method according to Claim 6, wherein horizontal and vertical groupings are formed by testing two elements at a time, a new grouping being formed when a rectangle can be formed which encloses the two tested elements and no other elements.

8. A method according to Claim 7, wherein the step of forming vertical and horizontal groupings includes alternating between forming all possible vertical groupings and forming all possible horizontal groupings.

9. A method according to Claim 7, wherein two identified elements are tested for vertical grouping only if the two identified elements are not horizontally adjacent to each other.

10. A method according to Claim 7, wherein two identified elements are tested for horizontal grouping only if the two identified elements are not vertically adjacent to each other.

11. A method according to Claim 6, wherein the identifying step includes ordering the blocks based on a sequence in which the vertical and horizontal groupings are formed.

12. A method according to Claim 3, wherein the identifying step includes a step of ordering the blocks based on the identified positional relationships.

13. A method for generating an HTML file based on bitmap image data, comprising the steps of:

segmenting the bitmap image into blocks;
analyzing layout relationships between the blocks based on the relative locations of the blocks in the bitmap image;
a block-type determination step of determining block type of each block based on the layout relationships obtained in said analyzing step;
a span determination step of determining column span and row span information for each block that spans more than one column and/or row; and
generating an HTML file based on the block type obtained in said block-type determination step and based on the column span and row span information obtained in said span determination step.

14. A method according to Claim 13, wherein said analyzing step includes iterative steps of combining blocks into vertical and horizontal groupings.

15. A method according to Claim 13, wherein said block-type determination step includes the step of assigning to each block a type selected from the group consisting of: a single block, a combined block, a joint block, and a multiple block.

16. A method according to Claim 15, wherein said step of assigning is based on proximity of each block to adjacent blocks and also based on layout information obtained is said analyzing step.

17. A method according to Claim 13, wherein analyzing step groups blocks into vertical and horizontal groupings, and wherein said span determination step includes the step of determining column span and row span by counting the number of blocks within vertical groupings and the number of blocks within horizontal groupings.

18. A method of generating layout data from block data comprised by horizontally and vertically arranged blocks, the method comprising:

a vertical determination step of determining whether a combination of vertically-arranged blocks overlaps onto any other of the horizontally and vertically arranged blocks;

a vertical combination step of combining the combination of vertically-arranged blocks in a case where said vertical determination step determines that there is no overlap;

a horizontal determination step of determining whether a combination of horizontally-arranged blocks overlaps onto any other of the horizontally and vertically arranged blocks;

a horizontal combination step of combining the combination of horizontally-arranged blocks in a case where said horizontal determination step determines that no overlap exists; and

a generation step of generating layout data based on the combinations made in said vertical combination step and said horizontal combination step.

19. A method according to Claim 18, wherein said vertical determination step is performed for all permissible combinations of vertically-arranged blocks before any determinations are made in said horizontal determination step.

20. A method according to Claim 19, wherein said block data includes coordinate data for each block, and wherein permissibility of combination is determined based on a comparison of coordinate data for respective blocks.

21. A method according to Claim 19, further comprising the step of repeating said vertical determination step after all permissible combinations of horizontally-arranged blocks are tested in said horizontal determination step.

22. A method according to Claim 21, wherein said block data includes coordinate data for each block, and wherein permissibility of combination is determined based on a comparison of coordinate data for respective blocks.

23. An apparatus for generating an HTML file based on a bitmap image, comprising:

obtaining means which obtains two horizontally adjacent image blocks in separate vertical columns of the bitmap image;

a memory which stores process steps that are executable by a processor; and

a processor which executes the process steps stored in the memory to generate an HTML file in which the blocks obtained by the obtaining means are tagged as data elements in a row of an HTML tagged table.

24. An apparatus according to Claim 23, wherein the processor assigns an HTML code to each block in accordance with a position of the block relative to other elements within the bitmap image.

25. An apparatus for generating an HTML file based on a bitmap image, comprising:

obtaining means for obtaining the bitmap image;

a memory which stores process steps that are executable by a processor; and

a processor which executes the process steps stored in the memory (1) to segment the bitmap image to obtain image blocks, (2) to determine where in the bitmap image each of the blocks is located, (3) to identify positional relationships between the blocks based on their relative locations, and (4) to generate an HTML file in which the blocks are tagged as being data elements in an HTML tagged table, the tags being determined in accordance with the identified positional relationships.

26. An apparatus according to Claim 25, wherein when the processor identifies positional relationships between the blocks, the processor forms vertical and horizontal groupings of the blocks.

27. An apparatus according to claim 25, wherein when the processor identifies positional relationships between the blocks, the processor forms a hierarchical structure.

28. An apparatus according to Claim 27, wherein the processor forms the hierarchical structure by forming horizontal and vertical groupings of elements, the elements consisting of previously formed groupings and ungrouped blocks.

29. An apparatus according to claim 28, wherein the processor forms the horizontal and vertical groupings by testing two elements at a time, a new grouping being formed when a rectangle can be formed which encloses the two tested elements and no other elements.

30. An apparatus according to claim 29, wherein the processor forms the vertical and horizontal groupings by alternat-

ing between forming all possible vertical groupings and forming all possible horizontal groupings.

31. An apparatus according to Claim 28, wherein the processor tests two identified elements for vertical grouping only if the two identified elements are not horizontally adjacent to each other.

32. An apparatus according to Claim 28, wherein the processor tests two identified elements for horizontal grouping only if the two identified elements are not vertically adjacent to each other.

33. An apparatus according to Claim 27, wherein the processor identifies the positional relationships between the blocks by ordering the blocks based on a sequence in which the vertical and horizontal groupings are formed.

34. An apparatus according to Claim 25, wherein the processor identifies the positional relationships between the blocks by ordering the blocks based on the identified positional relationships.

35. An apparatus for generating an HTML file based on bitmap image data, comprising:

obtaining means for obtaining the bitmap image data;
a memory for storing the bitmap image data and process steps executable by a processor; and
a processor which executes the process steps stored in the memory (1) to segment the bitmap image into blocks, (2) to analyze layout relationships between the blocks based on the relative locations of the blocks in the bitmap image, (3) to determine a block type of each block based on the layout relationships obtained by the processor, (4) to determine column span and row span information for each block that spans more than one column and/or row, and (5) to generate an HTML file based on the block type determined by the processor and the column span and row span information determined by the processor.

36. An apparatus according to Claim 35, wherein the processor analyzes layout relationships between the blocks by combining the blocks into vertical and horizontal groupings.

37. An apparatus according to Claim 35, wherein the processor determines a block type of each block by assigning to each block a type selected from the group consisting of: a single block, a combined block, a joint block, and a multiple block.

38. An apparatus according to Claim 37, wherein the processor assigns a block type to each block based on proximity of each block to adjacent blocks and also based on layout information obtained by the processor.

39. An apparatus according to Claim 35, wherein the layout relationship analysis step executed by the processor includes grouping the blocks into vertical and horizontal groupings, and wherein the processor determines column span and row span information for each block by counting the number of blocks within vertical groupings and the number of blocks within horizontal groupings.

40. An apparatus for generating layout data from block data comprised by horizontally and vertically arranged blocks, the apparatus comprising:

an obtaining means for obtaining the block data;
a memory for storing process steps executable by a processor; and
a processor which executes the process steps stored in the processor (1) to determine whether a combination of vertically-arranged blocks overlaps onto any other of the horizontally and vertically arranged blocks, (2) to combine the combination of vertically-arranged blocks in a case where said vertical determination step determines that there is no overlap, (3) to determine whether a combination of horizontally-arranged blocks overlaps onto any other of the horizontally and vertically arranged blocks, (4) to combine the combination of horizontally-arranged blocks in a case where said horizontal determination step determines that no overlap exists, and (5) to generate layout data based on the combinations made in said vertical combination step and said horizontal combination step.

41. An apparatus according to Claim 40, wherein the processor determine whether a combination of vertically-arranged blocks overlaps onto any other of the horizontally and vertically arranged blocks for all permissible combinations of vertically-arranged blocks before any determinations are made by the processor regarding overlapping of horizontally-arranged blocks.

42. An apparatus according to Claim 41, wherein said block data includes coordinate data for each block, and wherein permissibility of combination is determined based on a comparison of coordinate data for respective blocks.

43. An apparatus according to Claim 41, wherein the processor determines whether a combination of vertically-arranged blocks overlaps onto any other of the horizontally and vertically arranged blocks after all permissible combinations of horizontally-arranged blocks are tested.

44. An apparatus according to Claim 43, wherein said block data includes coordinate data for each block, and wherein permissibility of combination is determined based on a comparison of coordinate data for respective blocks.

45. Computer-executable process steps stored on a computer-readable medium, the computer executable process steps to generate an HTML file based on a bitmap image, the process steps comprising:

an obtaining step to obtain two horizontally adjacent image blocks in separate vertical columns of the bitmap image; and
a generating step to generate an HTML file in which the blocks are tagged as data elements in a row of an HTML tagged table.

46. Computer-executable process steps according to Claim 45, wherein an HTML code is assigned to each block in accordance with a position of the block relative to other elements within the bitmap image.

47. Computer-executable process steps stored on a computer-readable medium, the computer executable process steps to generate an HTML file based on a bitmap image, the computer-executable process steps comprising:

a segmenting step to segment the bitmap image to obtain image blocks;
a determining step to determine where in the bitmap image each of the blocks is located;
an identifying step to identify positional relationships between the blocks based on their relative locations; and
a generating step to generate an HTML file in which the blocks are tagged as being data elements in an HTML tagged table, the tags being determined in accordance with the identified positional relationships.

48. Computer-executable process steps according to Claim 47, wherein the identifying step includes forming vertical and horizontal groupings of the blocks.

49. Computer-executable process steps according to claim 37, wherein the identifying step includes forming a hierarchical structure.

50. Computer-executable process steps according to Claim 49, wherein the step of forming the hierarchical structure includes forming horizontal and vertical groupings of elements, the elements consisting of previously formed groupings and ungrouped blocks.

51. Computer-executable process steps according to Claim 50, wherein horizontal and vertical groupings are formed by testing two elements at a time, a new grouping being formed when a rectangle can be formed which encloses the two tested elements and no other elements.

52. Computer-executable process steps according to Claim 51, wherein the step of forming vertical and horizontal groupings includes alternating between forming all possible vertical groupings and forming all possible horizontal groupings.

53. Computer-executable process steps according to Claim 51, wherein two identified elements are tested for vertical grouping only if the two identified elements are not horizontally adjacent to each other.

54. Computer-executable process steps according to Claim 51, wherein two identified elements are tested for horizontal grouping only if the two identified elements are not vertically adjacent to each other.

55. Computer-executable process steps according to Claim 50, wherein the identifying step includes ordering the blocks based on a sequence in which the vertical and horizontal groupings are formed.

56. Computer-executable process steps according to Claim 47, wherein the identifying step includes a step to order the

blocks based on the identified positional relationships.

57. Computer-executable process steps stored on a computer-readable medium, the computer executable process steps to generate an HTML file based on bitmap image data, the computer-executable process steps comprising:

a segmenting step to segment the bitmap image into blocks;
an analyzing step to analyze layout relationships between the blocks based on the relative locations of the blocks in the bitmap image;
a block-type determination step to determine a block type of each block based on the layout relationships obtained in said analyzing step;
a span determination step to determine column span and row span information for each block that spans more than one column and/or row; and
a generating step to generate an HTML file based on the block type obtained in said block-type determination step and based on the column span and row span information obtained in said span determination step.

58. Computer-executable process steps according to Claim 57, wherein said analyzing step includes iterative steps to combine blocks into vertical and horizontal groupings.

59. Computer-executable process steps according to Claim 57, wherein said block-type determination step includes a step to assign to each block a type selected from the group consisting of: a single block, a combined block, a joint block, and a multiple block.

60. Computer-executable process steps according to Claim 59, wherein said step of assigning is based on proximity of each block to adjacent blocks and also based on layout information obtained is said analyzing step.

61. Computer-executable process steps according to Claim 57, wherein analyzing step groups blocks into vertical and horizontal groupings, and wherein said span determination step includes a step to determine column span and row span by counting the number of blocks within vertical groupings and the number of blocks within horizontal groupings.

62. Computer-executable process steps stored on a computer-readable medium, the computer executable process steps to generate layout data from block data comprised by horizontally and vertically arranged blocks, the computer-executable process steps comprising:

a vertical determination step to determine whether a combination of vertically-arranged blocks overlaps onto any other of the horizontally and vertically arranged blocks;
a vertical combination step to combine the combination of vertically-arranged blocks in a case where said vertical determination step determines that there is no overlap;
a horizontal determination step to determine whether a combination of horizontally-arranged blocks overlaps onto any other of the horizontally and vertically arranged blocks;
a horizontal combination step to combine the combination of horizontally-arranged blocks in a case where said horizontal determination step determines that no overlap exists; and
a generation step to generate layout data based on the combinations made in said vertical combination step and said horizontal combination step.

63. Computer-executable process steps according to Claim 62, wherein said vertical determination step is performed for all permissible combinations of vertically-arranged blocks before any determinations are made in said horizontal determination step.

64. Computer-executable process steps according to Claim 63, wherein said block data includes coordinate data for each block, and wherein permissibility of combination is determined based on a comparison of coordinate data for respective blocks.

65. Computer-executable process steps according to Claim 63, further comprising a repeating step to repeat said vertical determination step after all permissible combinations of horizontally-arranged blocks are tested in said horizontal determination step.

66. Computer-executable process steps according to Claim 65, wherein said block data includes coordinate data for

each block, and wherein permissibility of combination is determined based on a comparison of coordinate data for respective blocks.

67. A method and apparatus having the features of any combination of the preceding claims.

The universe from earth, and earth from the universe

1

2

SPOTLIGHT ON TECHNOLOGY

Canon is putting its technolgical capabilities to work in satellite and telescope projects that will change the way we view our world. Hideo Yokota, general manager of the SO (space optics) project in Canon's optical products operations, tells the story.

4

Satellite projects to reveal happenings at home

Canon's first project building equipment for use on a satellite commenced in 1990. "The device we worked on," Yokota explained, "is part of an earth observation satellite program currently in the development stages." The SO (space optics) project team handled lens development for this device, which was completed in late 1995. The satellite itself will be launched in the late 1990s by America's National Aeronautics and Space Administration (NASA).

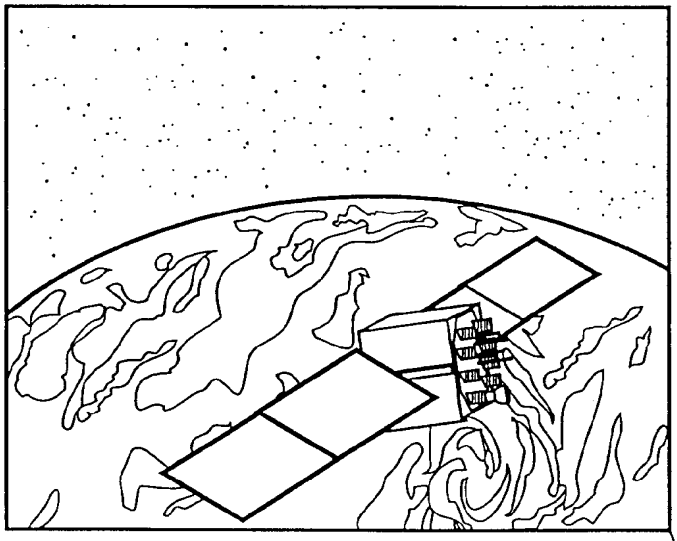
5

"The device using our lens monitors several bands in the thermal infrared region of the spectrum," Yokota continued, "making it useful for measuring thermal emission properties. This data will be used to study the warming of urban areas, the locations of mineral resources, the effects and extent of desertification, the movement patterns of marine life, and conditions in oceans and the atmosphere. The lens we developed improves geometric resolution, or the density of the land area covered, to 90 meters, enabling more detailed observation than previous systems."

6

The SO project team has also been involved in lens development for another satellite set to monitor conditions on planet earth. This observation satellite should be launched by the end of the century by the National Space Development Agency of Japan (NASDA).

"This time," Yokota said, "we are in charge of the lens system for a device similar to that developed in our first project. The difference is that this device will monitor spectral bands in the visable and near infrared regions, the short wavelength infrared region, and the middle and thermal infrared regions. The result will be a clearer picture of what is happening on land as well as in the sea and air.



▲ Shown here is an artist's rendition of optical communications between orbital satellites. (Illustrationsupplied by National Space Cevelopment Agency of Japan)

7

Eyes in the night

"For this lens, high performance and a large aperture were required, which is why we decided on an aspherical lens surface. Light coming into the mirror will be separated and sent, as appropriate, to one of several cameras. Through this project, it will be possible to observe vegetation and other environmental patterns."

Canon Chronicle May - June 1996

9

10

FIG. 1

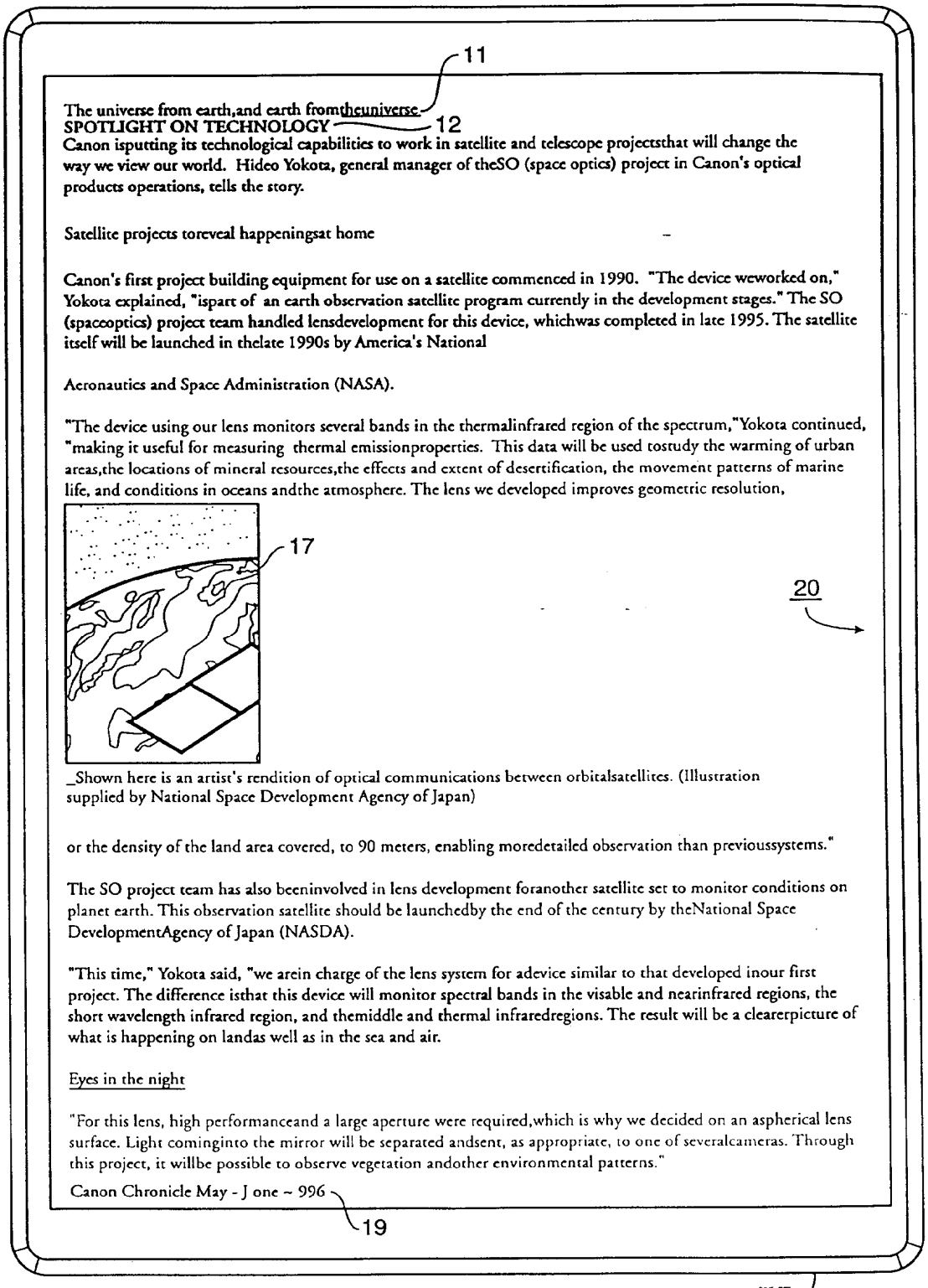


FIG. 2

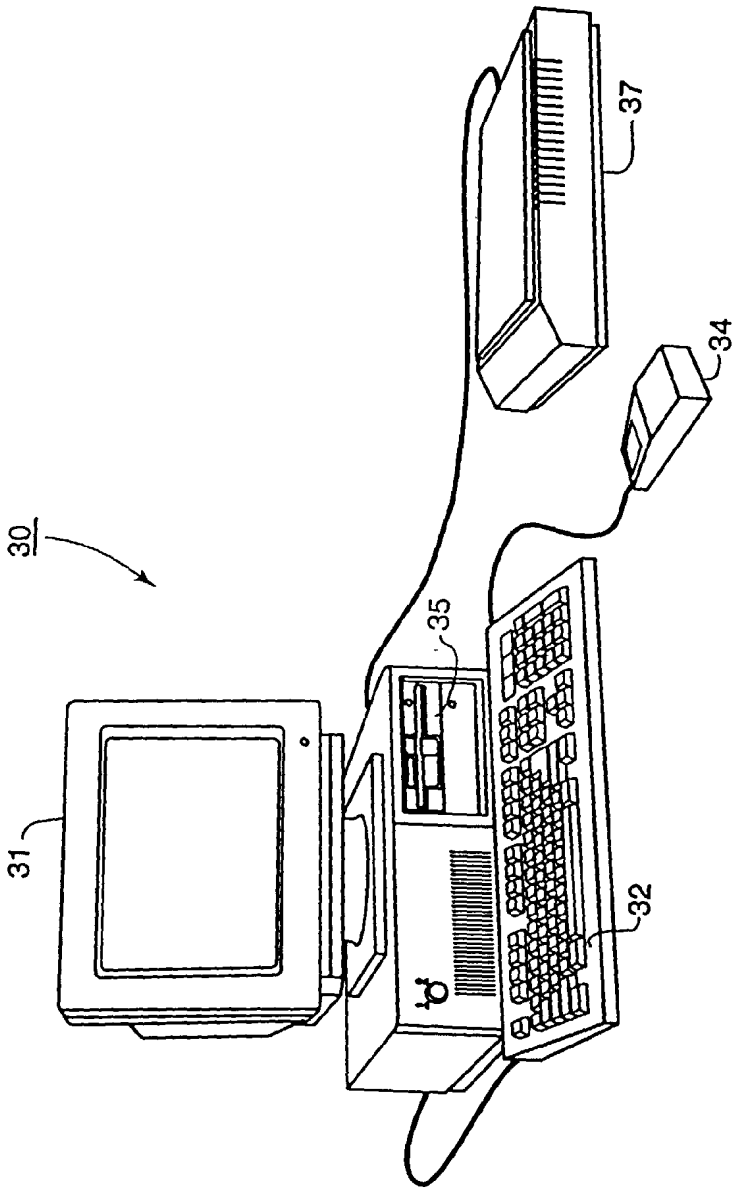


FIG. 3

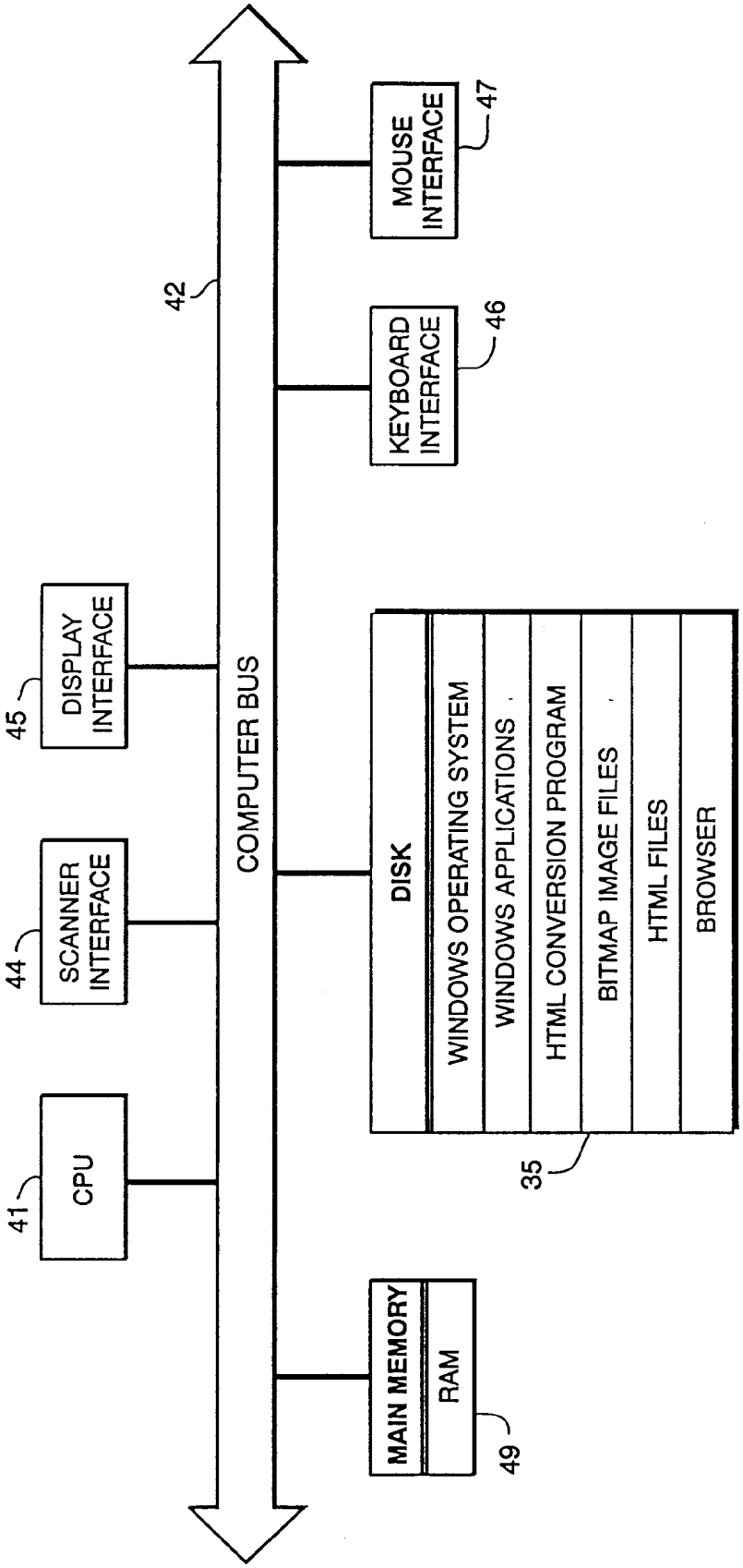


FIG. 4

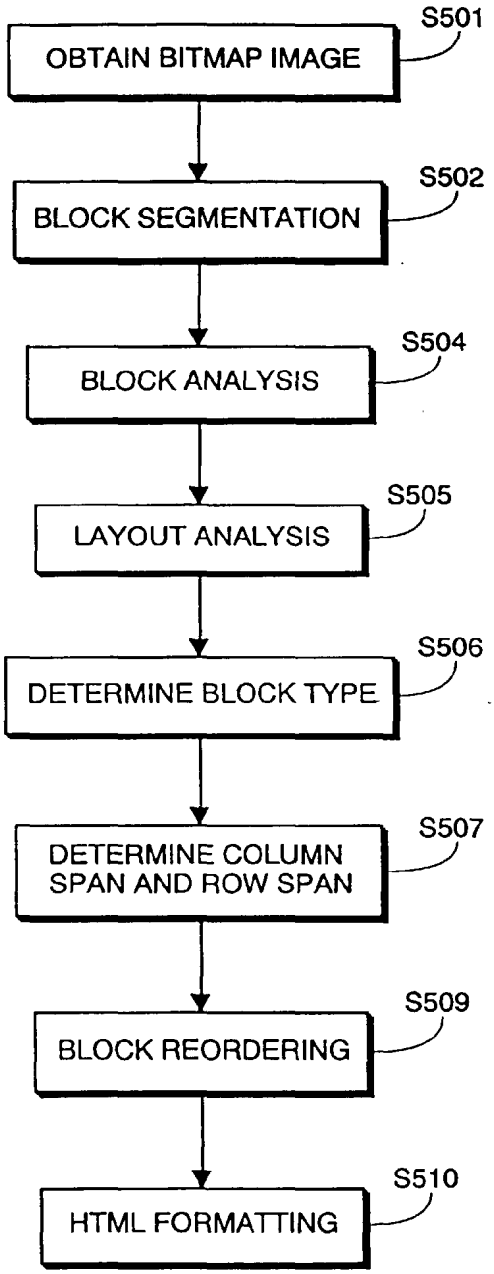


FIG. 5

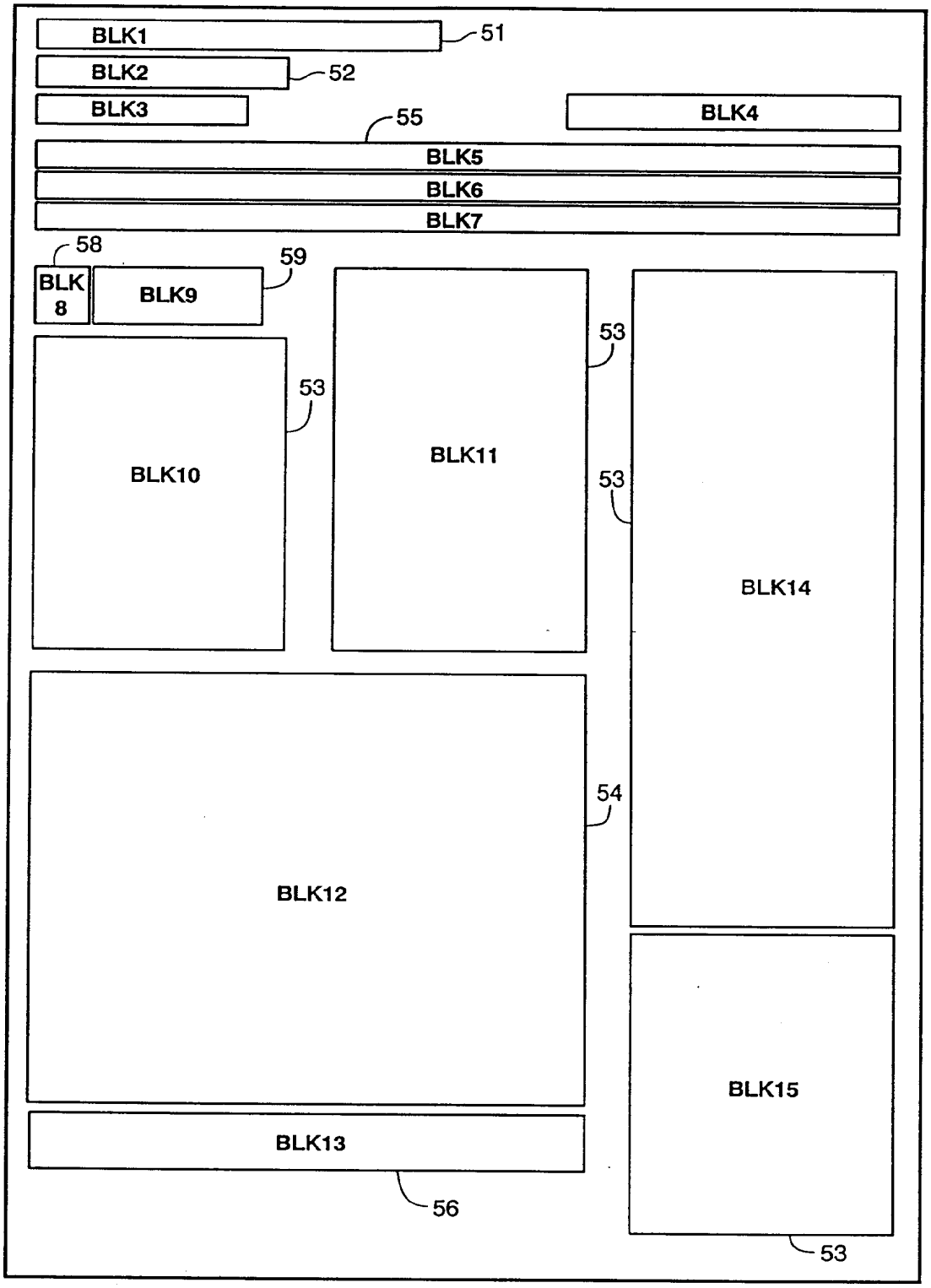


FIG. 6

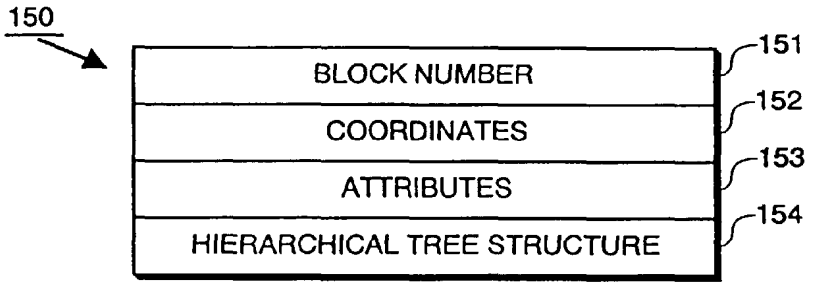


FIG. 7

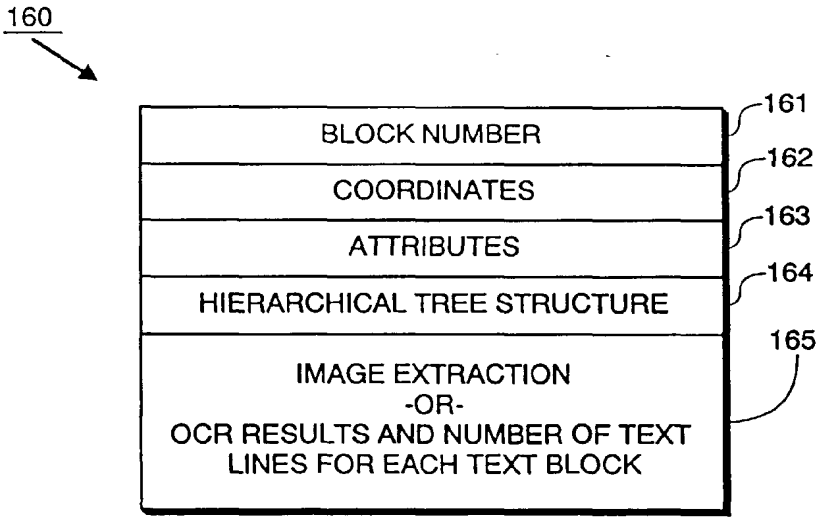


FIG. 8

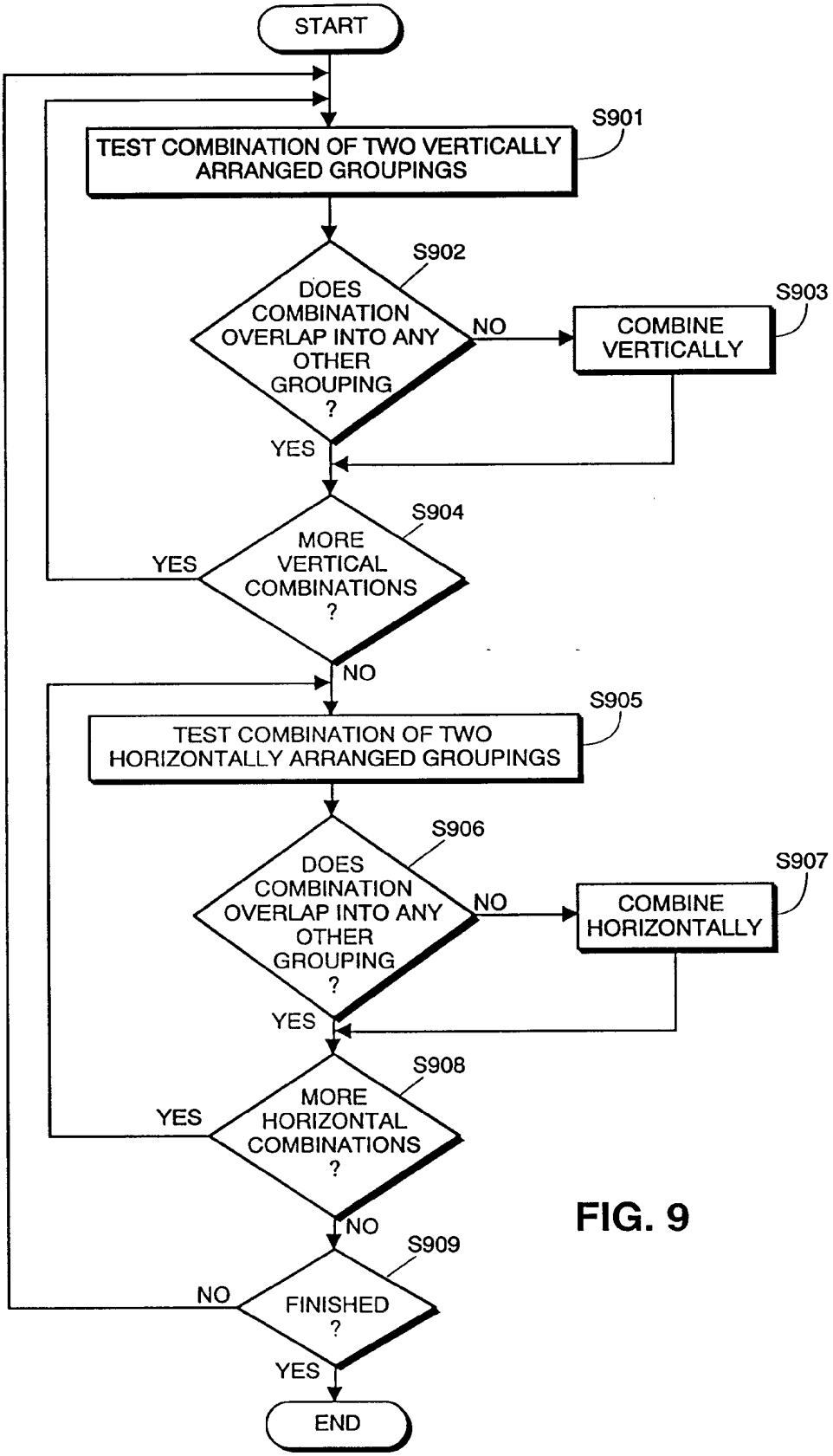


FIG. 9

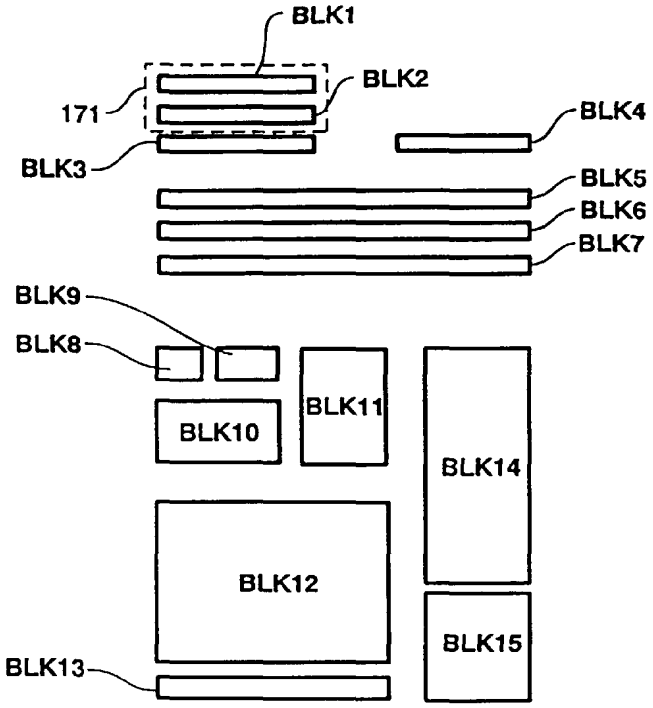


FIG. 10A-1

FIG. 10B-1

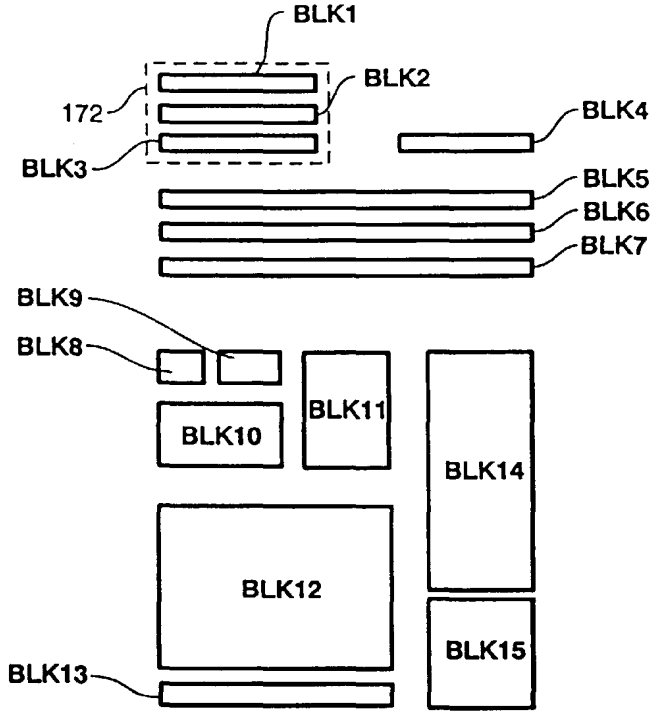


FIG. 10A-2

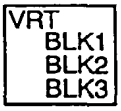


FIG. 10B-2

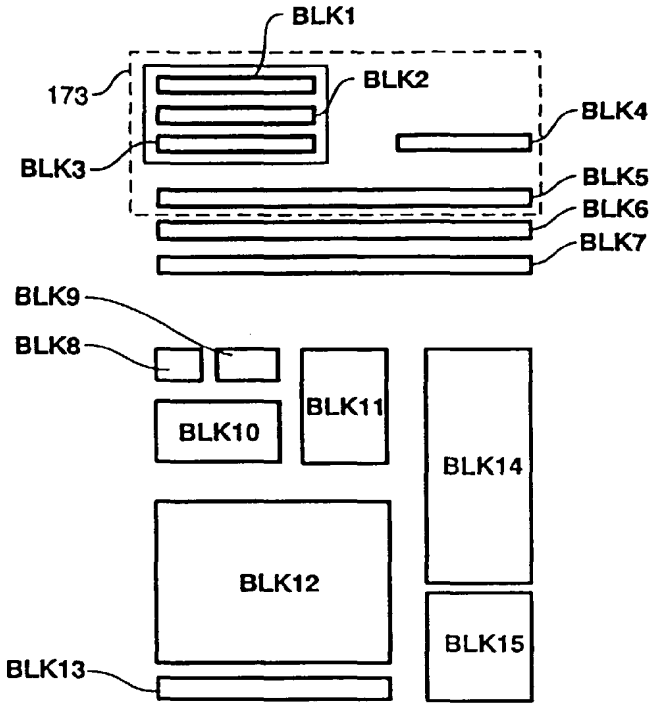


FIG. 10A-3

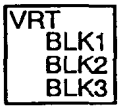


FIG. 10B-3

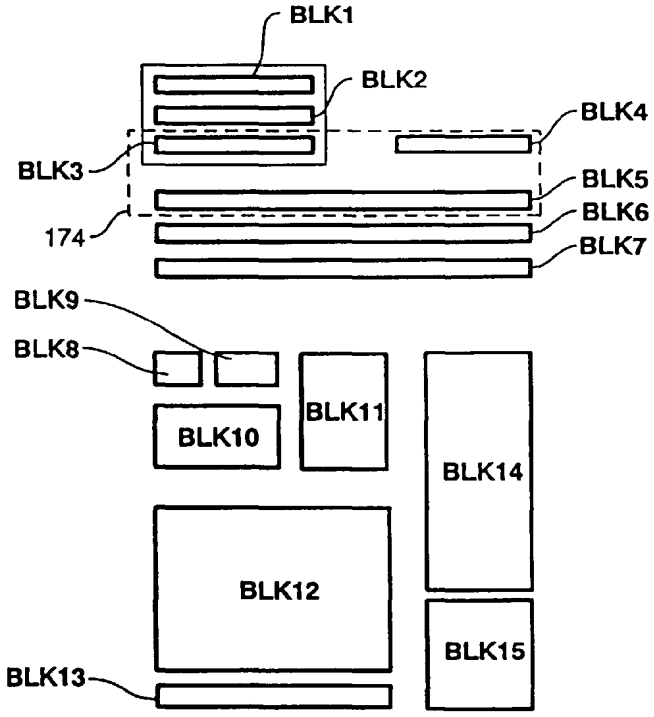


FIG. 10A-4

FIG. 10B-4

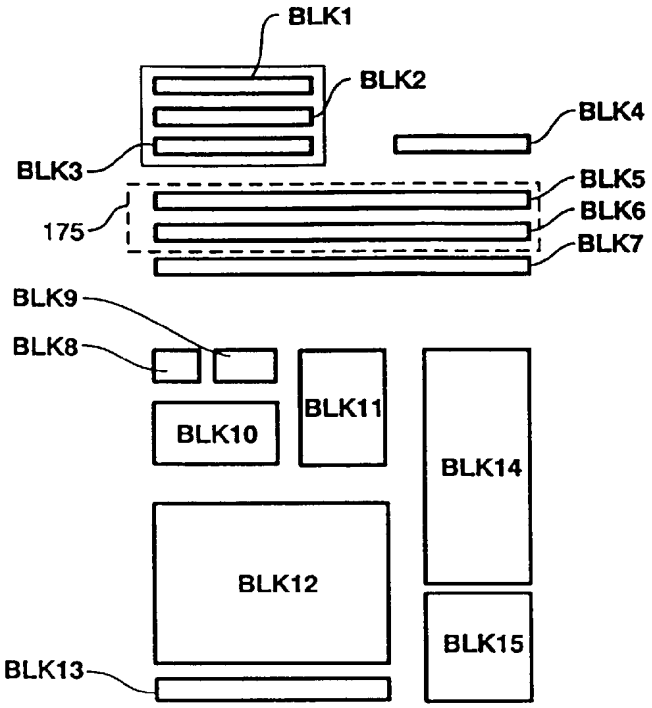


FIG. 10A-5

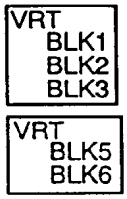


FIG. 10B-5

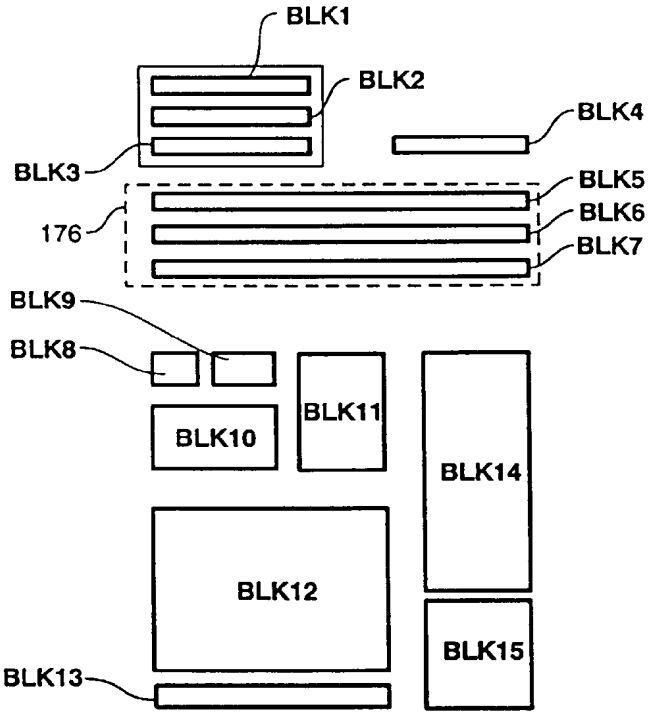


FIG. 10A-6

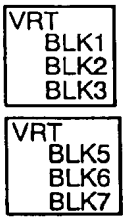


FIG. 10B-6

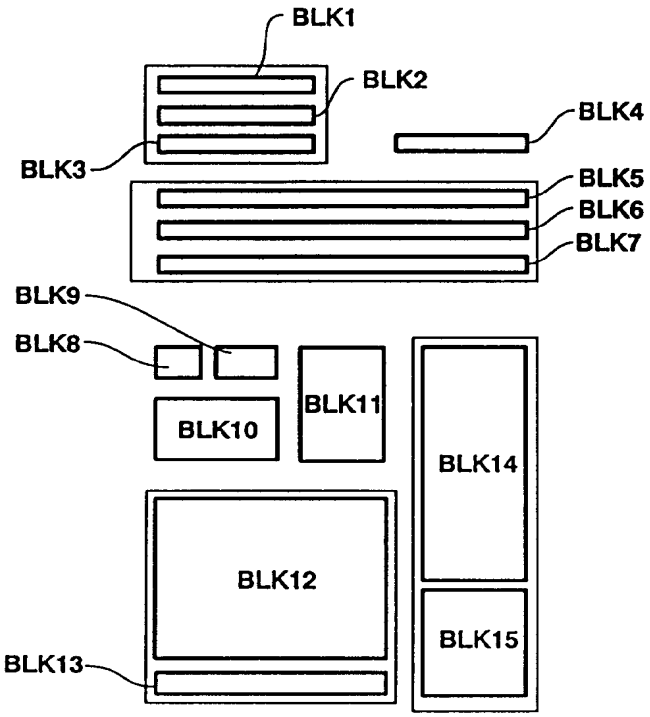


FIG. 10A-7

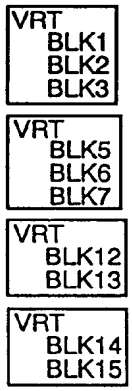


FIG. 10B-7

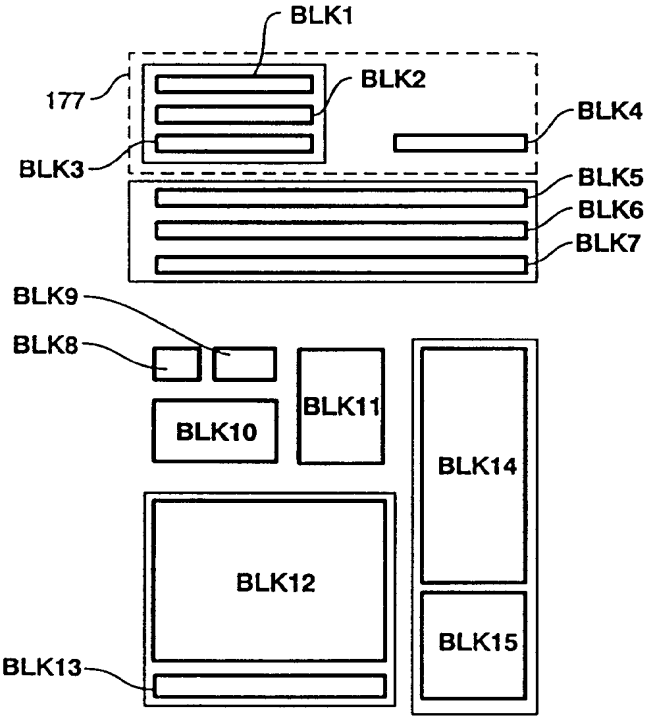


FIG. 10A-8

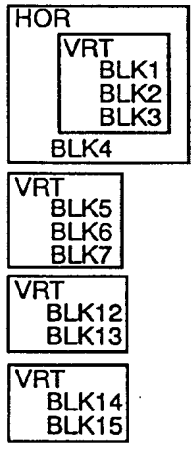


FIG. 10B-8

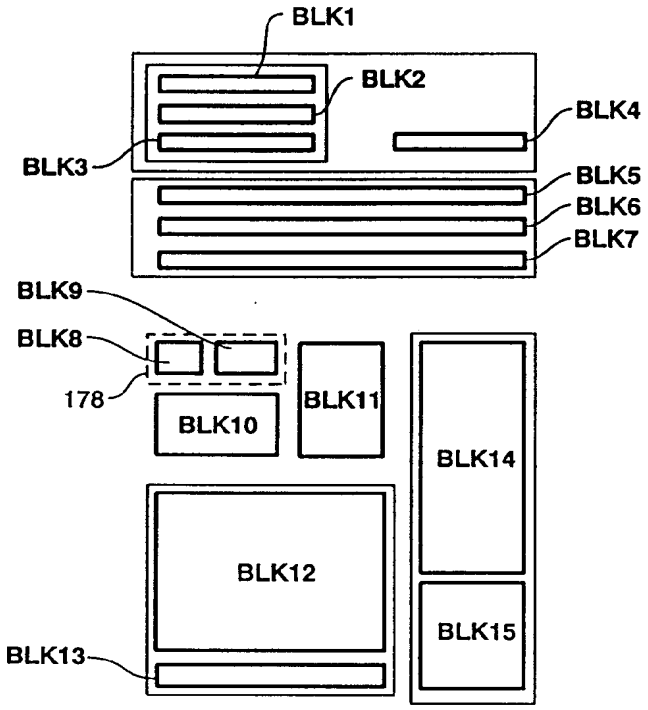


FIG. 10A-9

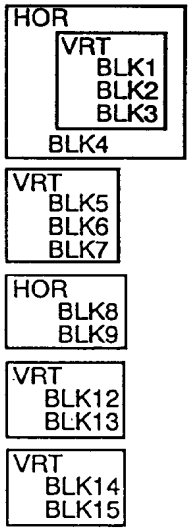


FIG. 10B-9

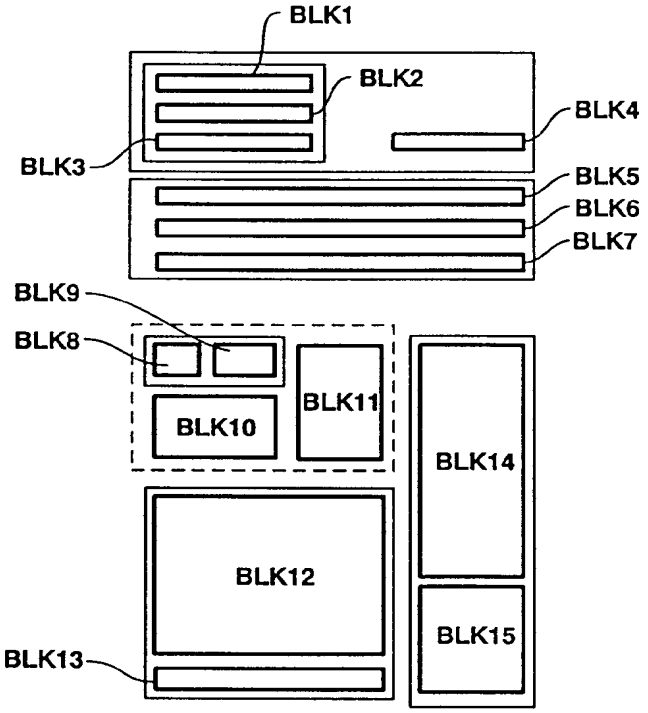


FIG. 10A-10

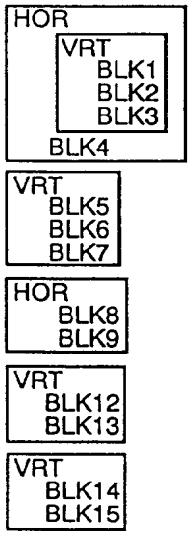


FIG. 10B-10

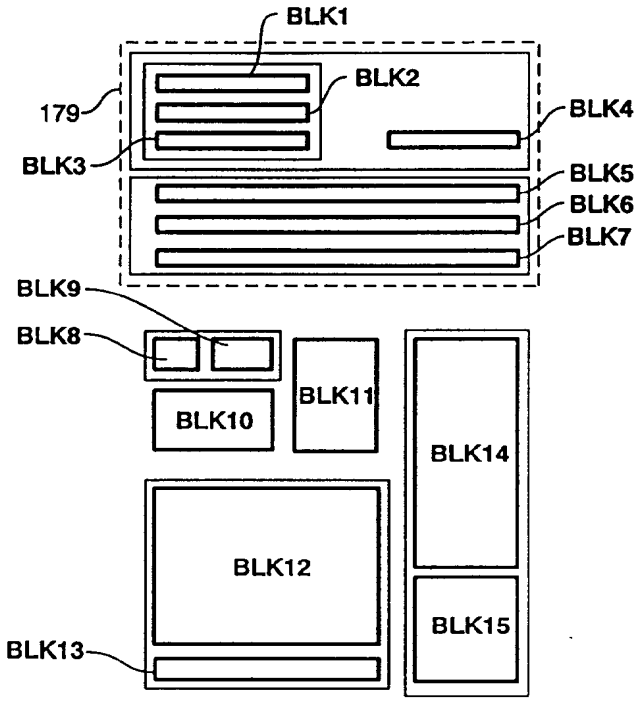


FIG. 10A-11

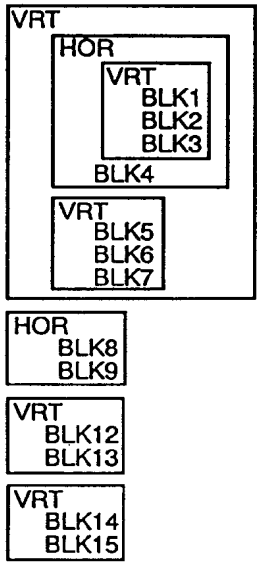


FIG. 10B-11

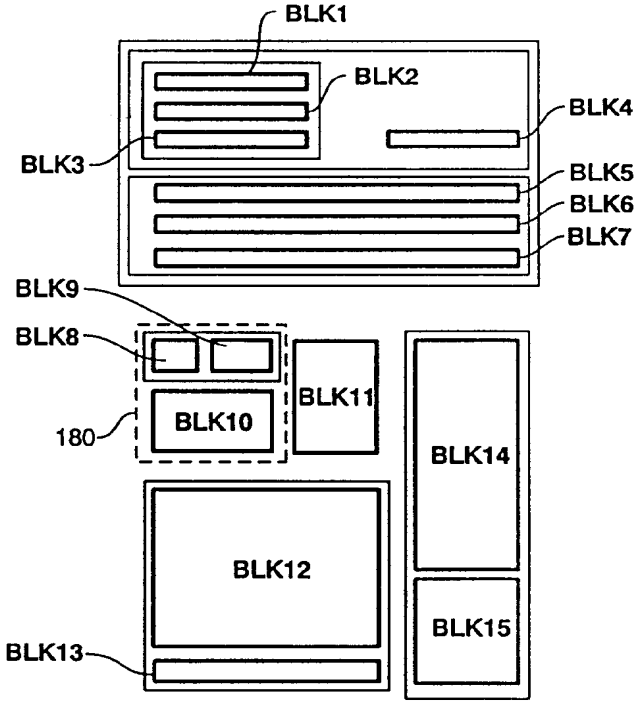


FIG. 10A-12

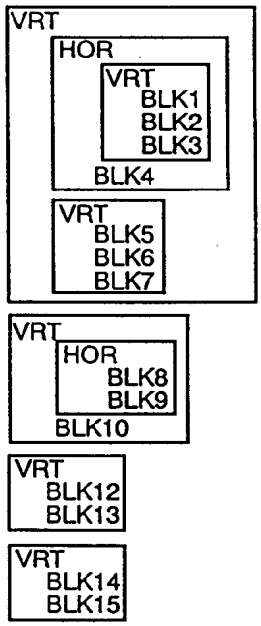


FIG. 10B-12

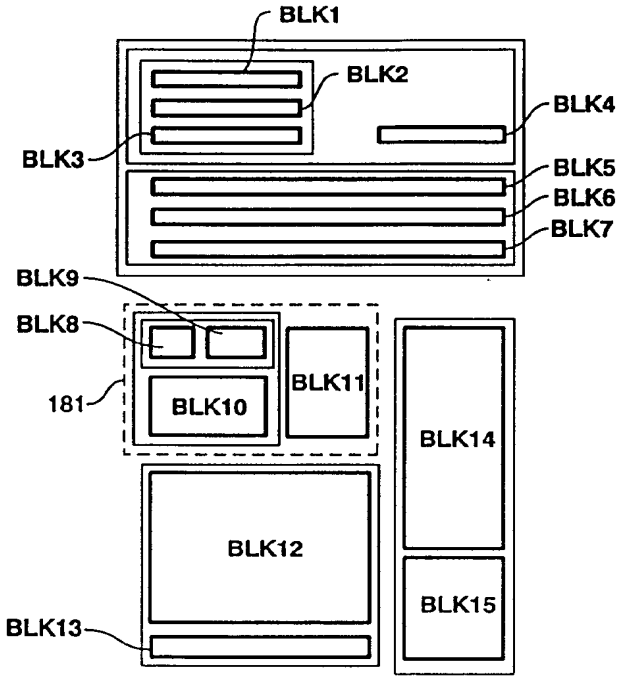


FIG. 10A-13

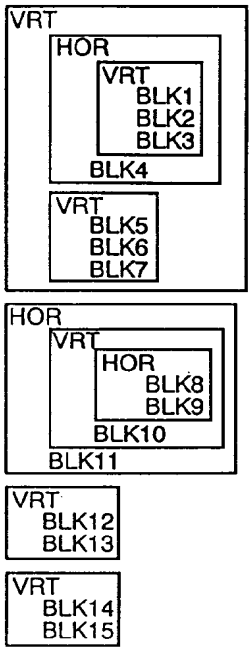


FIG. 10B-13

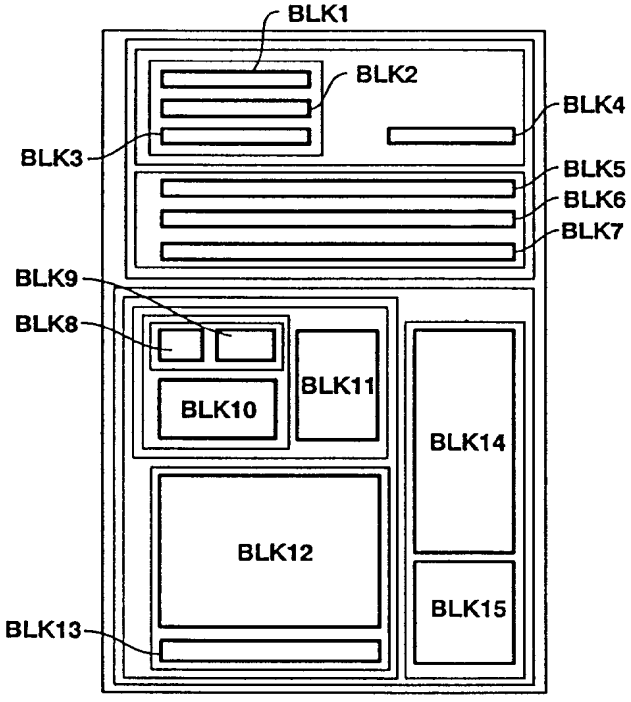


FIG. 10A-14

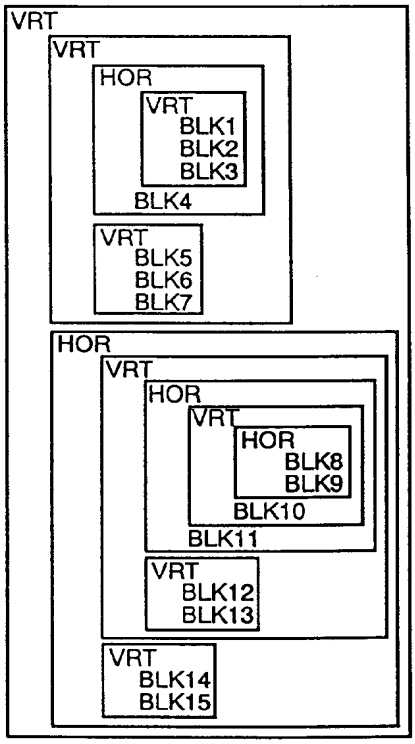


FIG. 10B-14

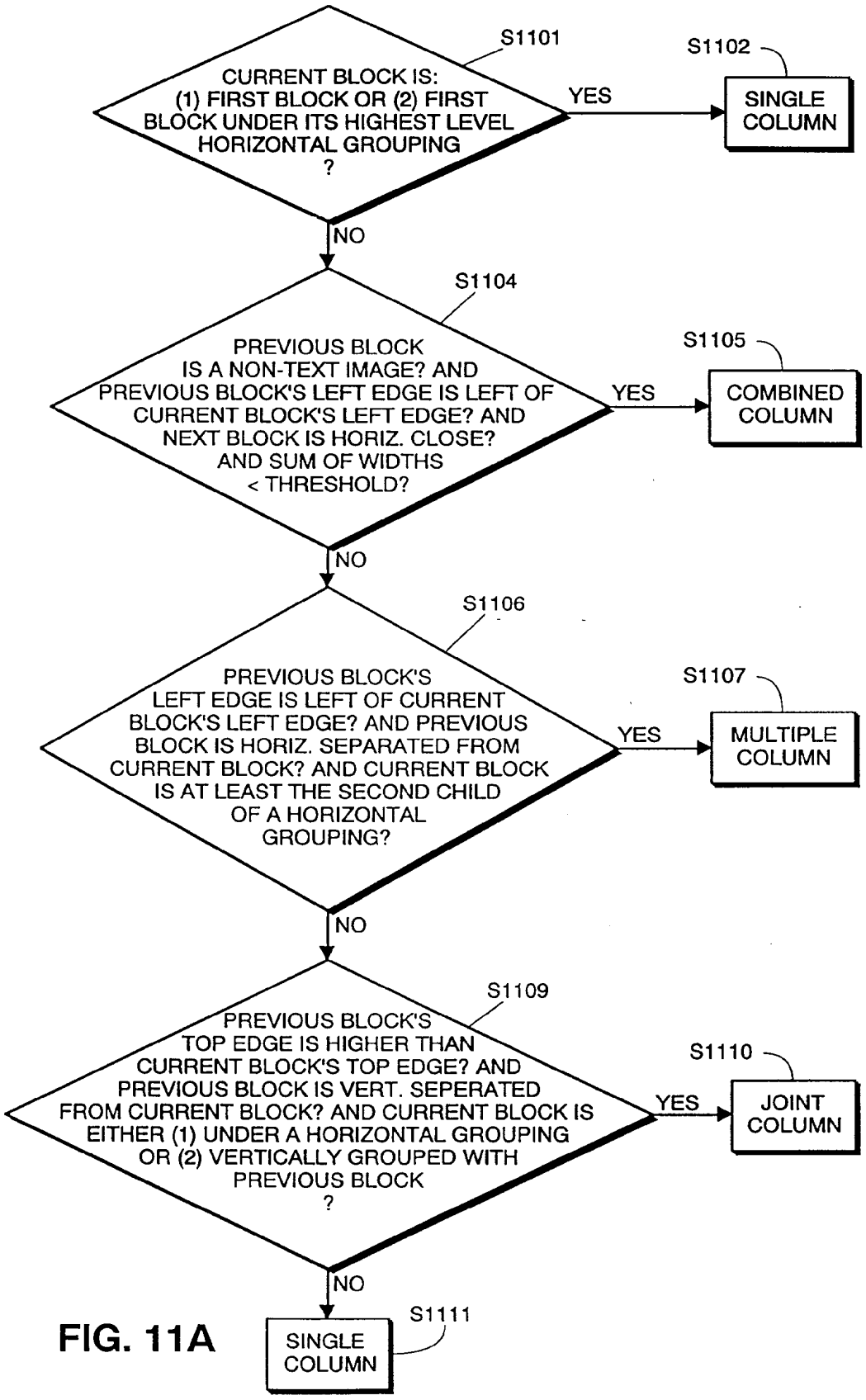


FIG. 11A

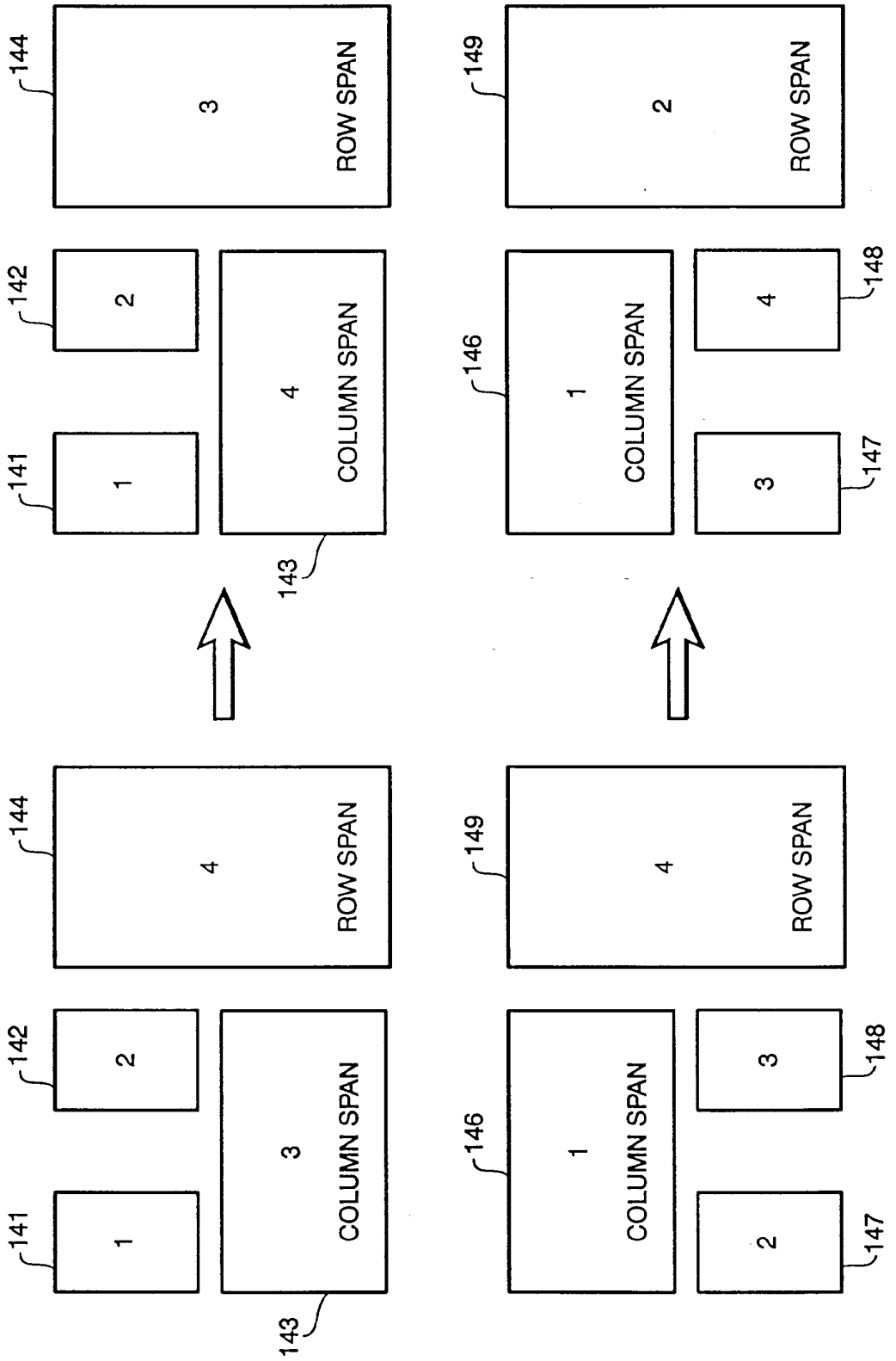


FIG. 11B

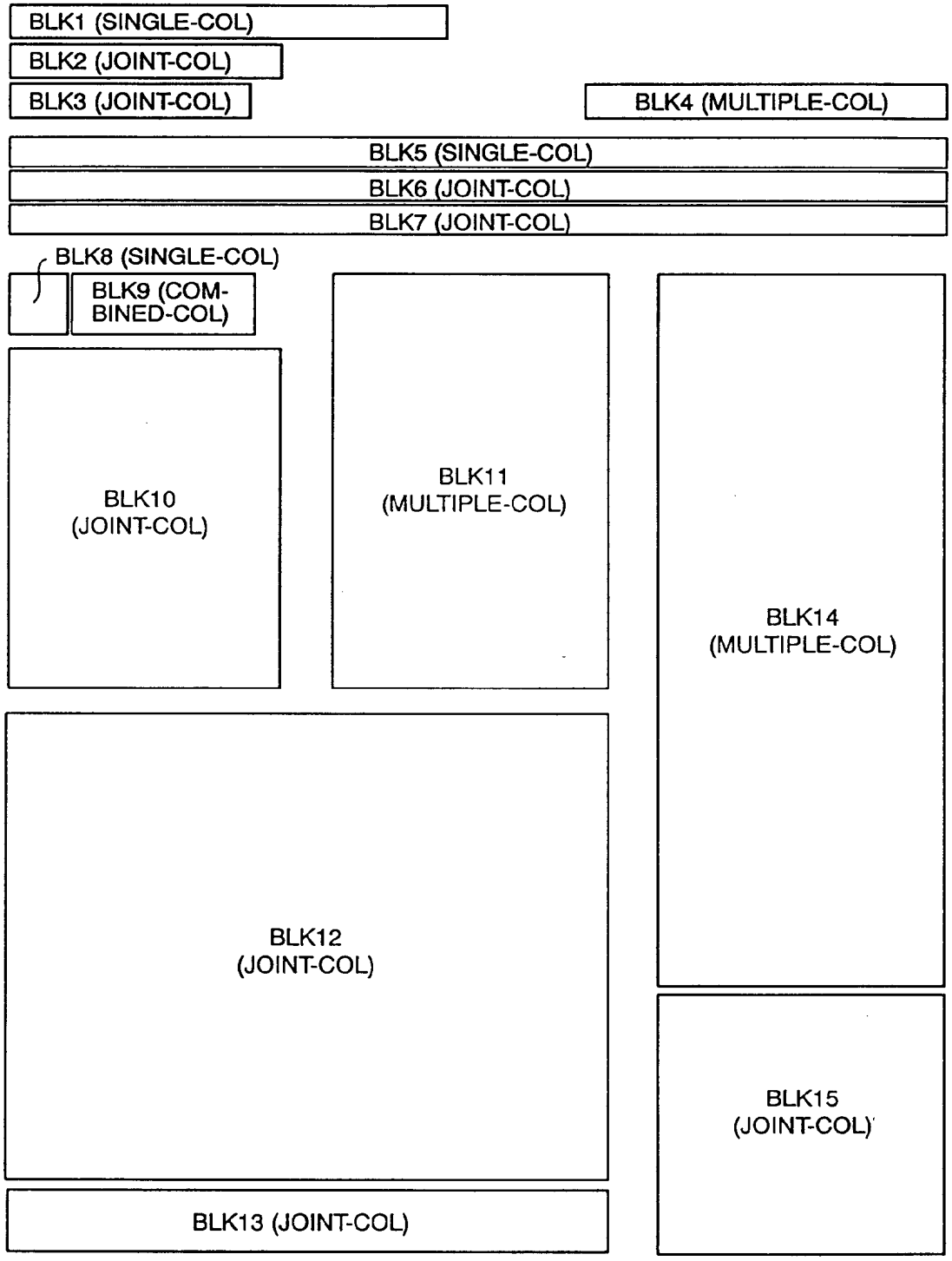


FIG. 12

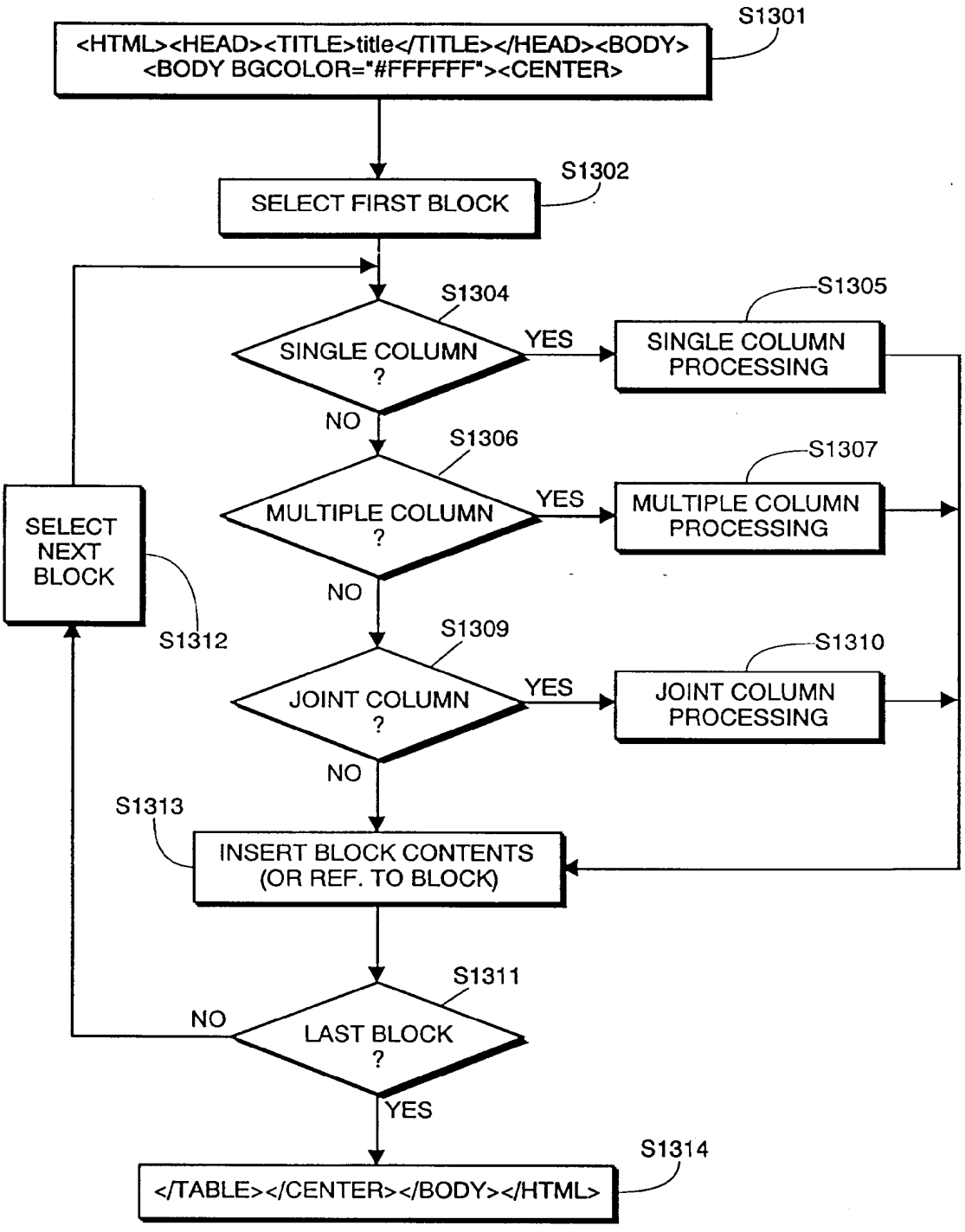


FIG. 13A

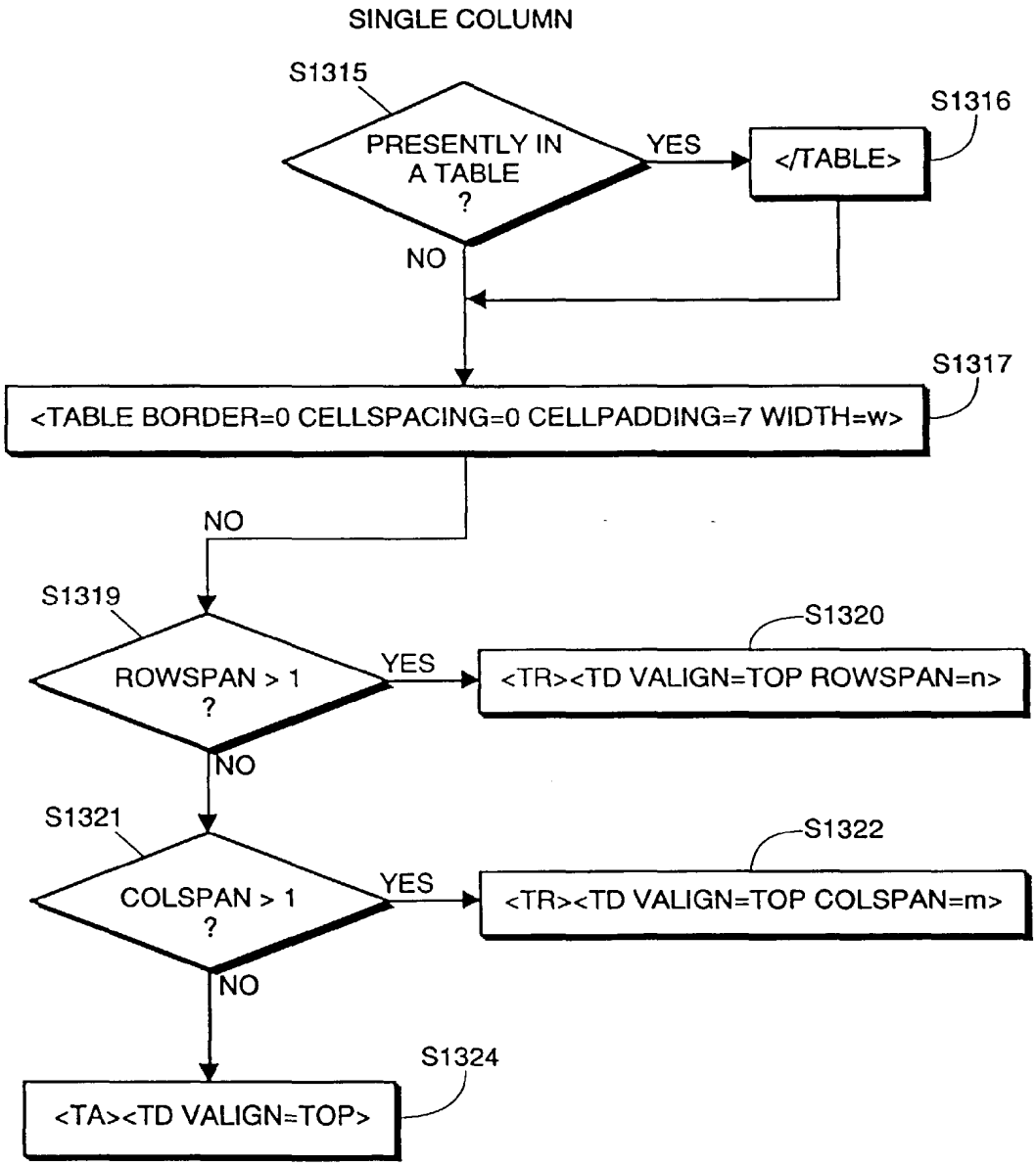


FIG. 13B

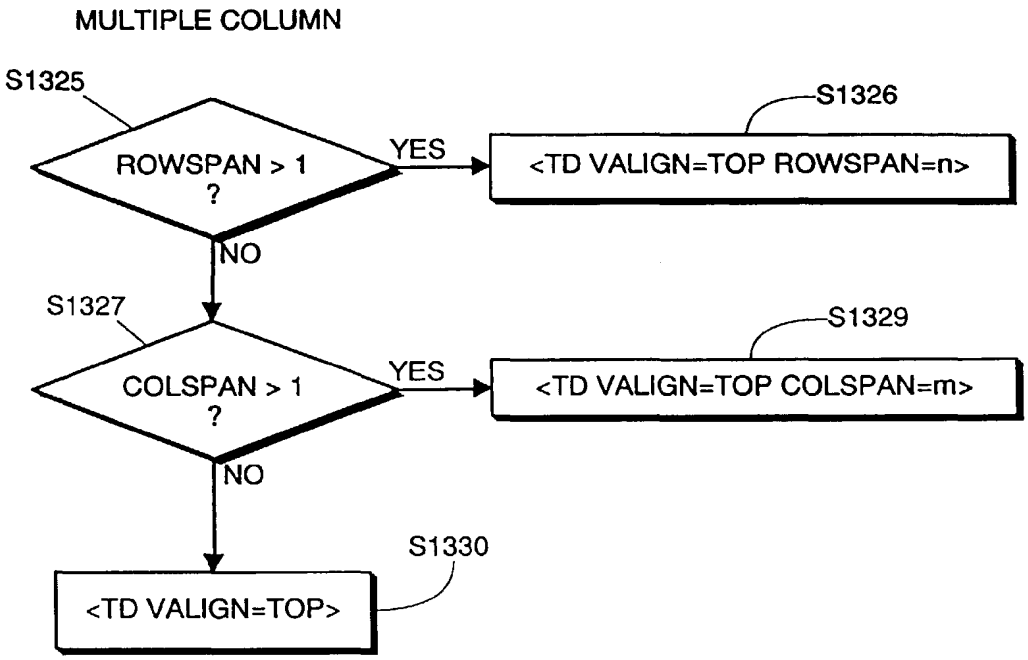


FIG. 13C

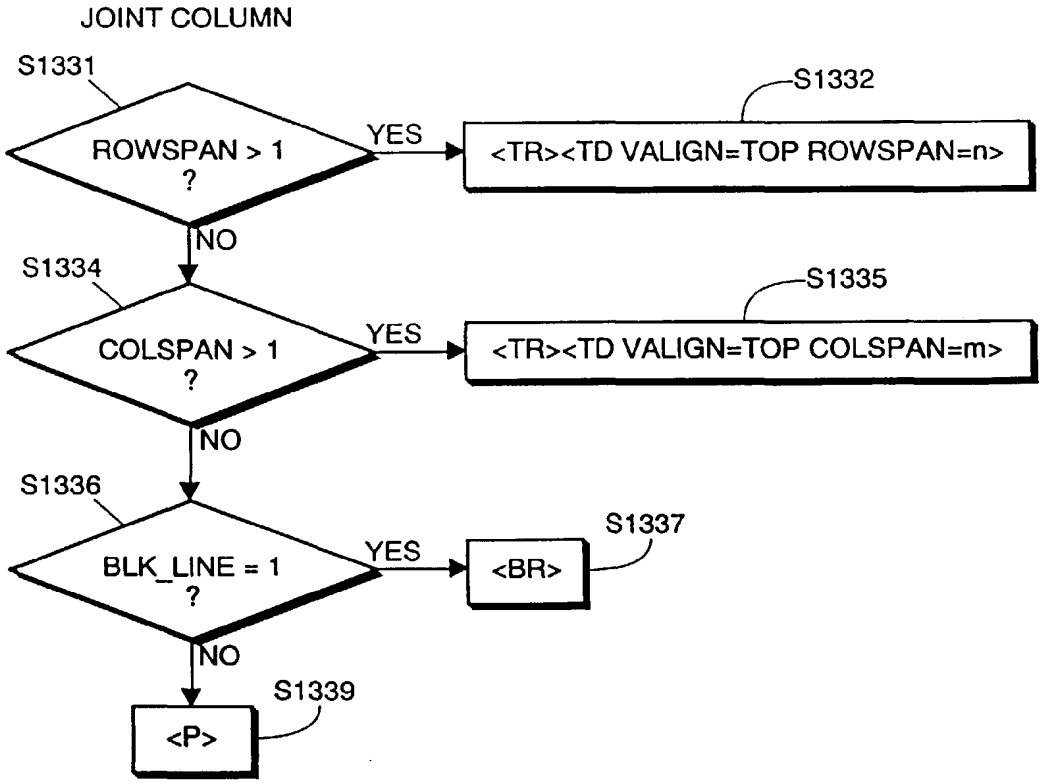


FIG. 13D

The universe from earth. and earth from the universe

201

SPOTLIGHT ON TECHNOLOGY

202

Canon is putting its technolgical capabilities to work in satellite and telescope projects that will change the way we view our world. Hideo Yokota, general manager of the SO (space optics) project in Canon's optical products operations, tells the story.

206

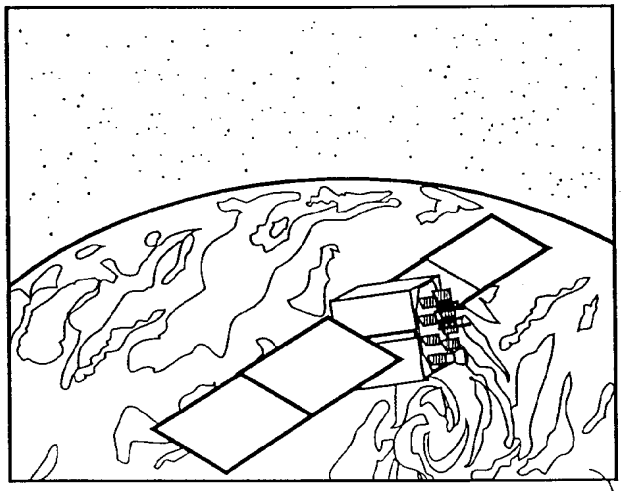
Satellite projects to reveal happenings at home

204

Canon's first project building equipment for use on a satellite commenced in 1990. "The device we worked on," Yokota explained, "is part of an earth observation satellite program currently in the development stages." The SO (space optics) project team handled lens development for this device, which was completed in late 1995. The satellite itself will be launched in the late 1990s by America's National Aeronautics and Space Administration (NASA).

"The device using our lens monitors several bands in the thermal infrared region of the spectrum," Yokota continued, "making it useful for measuring thermal emission properties. This data will be used to study the warming of urban areas, the locations of mineral resources, the effects and extent of desertification, the movement patterns of marine life, and conditions in oceans and the atmosphere. The lens we developed improves geometric resolution, or the density of the land area covered, to 90 meters, enabling more detailed observation than previous systems."

205



A Shown here is an artist's rendition of optical communications between orbital satellites. (Illustration supplied by National Space Cevelopment Agency of Japan)

207

The SO project team has also been involved in lens development for another satellite set to monitor conditions on planet earth. This observation satellite should be launched by the end of the century by the National Space Development Agency of Japan (NASDA).

"This time," Yokota said, "we are in charge of the lens system for a device similar to that developed in our first project. The difference is that this device will monitor spectral bands in the visable and near infrared regions, the short wavelength infrared region, and the middle and thermal infrared regions. The result will be a clearer picture of what is happening on land as well as in the sea and air.

Eyes in the night

"For this lens, high performance and a large aperture were required, which is why we decided on an aspherical lens surface. Light coming into the mirror will be separated and sent, as appropriate, to one of several cameras. Through this project, it will be possible to observe vegetation and other environmental patterns."

Canon Chronicle May -June 1996

209

FIG. 14

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 97 30 4451

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
Y	R. GANN: "Caere inproves its OCR interface" PC USER, no. 289, 21 August 1996 - 3 September 1996, GB, page 44 XP002055985 * the whole document *	1-67	G06K9/20 G06F17/21
Y	R. GANN: "Accurate OCR for complex pages" PC USER, no. 292, 2 - 15 October 1996, GB, page 50 XP002055986 * the whole document *	1-67	
P,Y	EP 0 758 775 A (CANON KABUSHIKI KAISHA) * the whole document *	1-67	
Y	EP 0 660 256 A (CANON KABUSHIKI KAISHA) * page 2, line 7 - page 18, line 6 *	1-67	
A	YOUNG SEAK PARK ET AL: "A Hierarchical Method for Block Segmentation and Classification of General Document Images" SYSTEMS AND COMPUTERS IN JAPAN, vol. 24, no. 9, 1993, NEW YORK, US, pages 84-96, XP000433079 * the whole document *	1-67	TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06K G06F
P,A	YUAN T. TANG ET AL: "AUTOMATIC DOCUMENT PROCESSING: A SURVEY" PATTERN RECOGNITION, vol. 29, no. 12, December 1996, GB, pages 1931-1952, XP000639709 * the whole document *	1-67	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	18 February 1998	Abram, R

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 876 034 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
04.11.1998 Bulletin 1998/45

(51) Int Cl.[6]: **H04L 29/06**, G06F 17/30

(21) Application number: **98300847.5**

(22) Date of filing: **05.02.1998**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **10.02.1997 US 797902**

(71) Applicant: **International Business Machines Corporation**
Armonk, N.Y. 10504 (US)

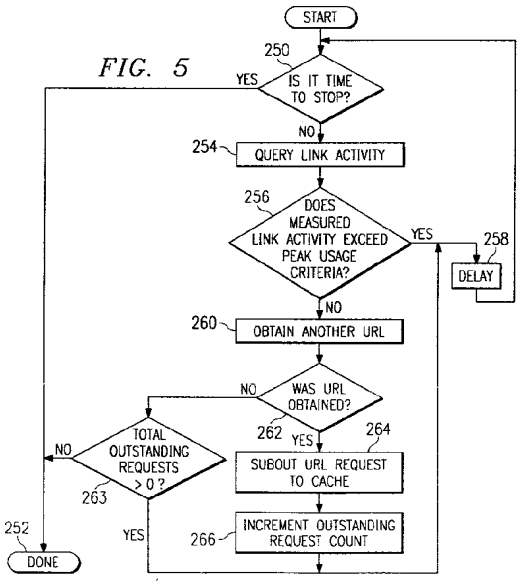
(72) Inventors:
- **Thompson, Joseph Raymond**
  Round Rock, Texas 78681 (US)
- **Berstis, Viktors**
  Austin, Texas 78746 (US)

(74) Representative: **Davies, Simon Robert**
IBM,
United Kingdom Limited,
Intellectual Property Law,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Method for content retrieval over a network

(57) A method is provided for retrieving Web content from a plurality of Web servers for delivery to a Web client connectable to the World Wide Web via a communication link 227. The Web client is preferably a data processing system connectable to a television 104 or other conventional monitor to provide low cost Internet access. The method begins by having the user define a set of one or more servers from which content is desired to be retrieved and stored in the cache. These servers are preferably identified by a list of favorite Web sites. A test is then made to determine whether a given download period has terminated 250. Typically, this download period occurs during an "off" period, such as in the middle of the night, to avoid traffic congestion at the Web server sites. If the given download period has not terminated, a determination is then made of an activity level for the communication link as content is being downloaded to the cache from the one or more servers 254. If the activity level for the communication link is less than a given threshold level, additional requests for content are issued to the cache 260 according to a so-called "fairness policy" that ensures that content from as many sites as possible is downloaded during the download period.

FIG. 5



EP 0 876 034 A2

Description

The present invention relates generally to information retrieval over the World Wide Web or such like, and more particularly to retrieving content for delivery to a client connectable to a network via a communication link, the client including a cache.

The World Wide Web of the Internet is the most successful distributed application in the history of computing. In the Web environment, client machines effect transactions to Web servers using the Hypertext Transfer Protocol (HTTP), which is a known application protocol providing users access to files (e.g., text, graphics, images, sound, video, etc.) using a standard page description language known as Hypertext Markup Language (HTML). HTML provides basic document formatting and allows the developer to specify "links" to other servers and files. In the Internet paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL) having a special syntax for defining a network connection. Use of an HTML-compatible browser (e.g., Netscape Navigator or Microsoft Internet Explorer) at a client machine involves specification of a link via the URL. In response, the client makes a request to the server identified in the link and receives in return a document formatted according to HTML.

There has been great interest in providing Internet access at minimal economic cost. While most computers now are pre-configured for Internet access, a significant percentage of households still do not have a personal computer. Thus, it has now been proposed to provide a data processing system that, much like a VCR, may be connected to a television set and used in lieu of a personal computer to provide Web access through a conventional remote control device associated with the system unit. Such a system enables the television to become, in effect, a "Web appliance". The viewer can rapidly switch between conventional television and Internet access using the remote control unit. All of the conventional Internet access tools and navigational functions are preferably built-in to the system and thus hidden to the user.

One such tool is so-called "off-line" browsing. As any casual user of the Internet can attest, interesting or attractive web sites are sometimes difficult to access due to large traffic demands. As a result, several companies have developed so-called "off-line" browser programs that are designed to deliver web pages from favorite Web servers to a user's hard drive for browsing at the user's convenience. Typically, such programs include some form of scheduling feature that enables the user to fetch identifiable pages at off-peak hours, saving time and connection charges. The user may then browse the pages at his or her convenience without a modem and even without an active connection to the Internet.

While off-line browser programs offer certain advantages, they do not have the capability to optimize utilization of the communication link between the client and the World Wide Web during the off-peak information retrieval process. This problem becomes more acute if there are constraints on the amount of time that off-peak information retrieval may be accomplished. In the future, it is anticipated that Web appliances of the type described above will be provided by computer network or other service providers, who will only allow their subscribers limited periods of time during which off-line browsing will be permitted. Thus, for example, a network operator may restrict subscribers to off-line browsing for just one hour per night. During this hour, a user may desire to obtain content from numerous Web sites. Thus, it would be desirable to provide some mechanism that could optimize retrieval of Web content during this limited period of time.

Accordingly, the present invention provides a method of retrieving content for delivery to a client connectable to a network via a communication link, the client including a cache, said method comprising the steps of:

(a) defining a set of servers from which content is desired to be retrieved and stored in the cache;
(b) determining an activity level for the communication link as content is being downloaded to the cache from the servers; and
(c) if the activity level for the communication link is less than a threshold level, issuing to the cache additional requests for content such that each of the servers of the set has an opportunity to deliver content to the client.

In the preferred embodiment the network is the World Wide Web, and said client and servers are a Web client and Web servers respectively. The additional requests for content may be issued to the set of servers as one request per server in an ordered sequence, or up to a predetermined number of requests per server in an ordered sequence, or based on any suitable predetermined parameters, for example the link depth of a document located on a particular server, or the number of bytes received in the cache from a particular server. The threshold level of the communication link may be a given number of outstanding requests for content, or related to an average link utilization rate for a given monitored interval, or any other suitable criterion. In the preferred embodiment the content received from a server is processed to remove duplicative links and non-local links.

Typically the content is downloaded to the cache during a predetermined time period, for example, approximately one hour. This period may be set by the network access provider, or is possibly selectable by the user. In such circumstances, it is determined whether a given download period has terminated, and if not, said step of determining an activity level for the communication link is performed. The additional requests for content are issued to the cache such that each of the serv-

ers at least has an opportunity to deliver content to the client during the given download period.

The invention further provides a computer program product for retrieving content for delivery to a client connectable to a network via a communication link, the client including a cache, the computer program product comprising: a computer-readable storage medium having a substrate; and a program encoded in the substrate of the computer-readable storage medium, wherein the program comprises means for performing the methods described herein.

The invention further provides a computer comprising a modem connected to a communication link; a processor; a memory including a cache; a browser program running on the processor for providing World Wide Web information retrieval and including means for generating a list of Web sites to be downloaded; a cache control program running on the processor and including means for initiating download requests to the communication link based on the list of Web sites; and a policy control program running on the processor and including means for issuing multiple HTTP GET requests to the cache control program to ensure that each of the Web sites on the list has an opportunity to contribute content during a given download session. Such a computer may function as a base unit in a data processing system having a remote control unit, in which the base unit is connectable to a monitor for providing Internet access under the control of the remote control unit.

Preferably the Web browser program includes menu means for generating the list of Web sites to be downloaded during an off-peak download session; the cache control program includes means for receiving downloads that are stored in the cache for off-line browsing; and the cache control program ensures that each of the Web sites on the list has an opportunity to contribute content during a given download session.

Viewed from another aspect, the invention provides a method of retrieving Web content for delivery to a Web client connectable to the World Wide Web via a communication link, the Web client including a cache, comprising the steps of:

(a) defining a set of Web servers from which content is desired to be retrieved and stored in the cache;
(b) determining an activity level for the communication link as content is being downloaded to the cache from the servers; and
(c) if the activity level for the communication link is less than a threshold level, issuing to the cache additional requests for content according to a policy that ensures that each of the servers of the set has an opportunity to deliver content to the Web client.

Viewed from a further aspect, the invention provides a computer program product for retrieving Web content for delivery to a Web client connectable to the World Wide Web via a communication link, the Web client including a cache, the computer program product comprising:

a computer-readable storage medium having a substrate; and
program data encoded in the substrate of the computer-readable storage medium, wherein the program data comprises: means for defining a set of servers from which content is desired to be retrieved and stored in the cache; means for monitoring an activity level for the communication link as content is being downloaded to the cache from the servers; and means responsive to the monitoring means for issuing to the cache additional requests for content according to a fairness policy that ensures that each of the servers has an opportunity to contribute content to the Web client during a download session.

Viewed from a further aspect, the invention provides a computer, comprising: a processor; a memory including a cache; a modem connected to a communication link; a Web browser program run by the processor for providing World Wide Web information retrieval and including menu means for generating a list of Web sites to be downloaded during an off-peak download session; a cache control program run by the processor for initiating download requests to the communication link based on the list of Web sites and for receiving downloads that are stored in the cache for off-line browsing; and a fairness policy control program run by the processor issuing multiple HTTP GET requests to the cache control program to ensure that each of the Web sites on the list has an opportunity to contribute content during a given download session.

Thus utilization of the communication link between a Web appliance and World Wide Web servers may be optimised by enhancing the off-peak caching of Web data when client access to the network is restricted, by ensuring that the link between a Web client and one or more Web servers is used to its maximum bandwidth during such a time-restricted off-peak browsing session. This represents an improvement in the functionality of off-line browsing programs to make more efficient use of limited communication resources, and helps to ensure maximum utilization of the Web client modem during off-peak caching of Web data from the World Wide Web of the Internet. The approach described herein allows equitable caching of content from a plurality of Web sites during an automatic download session so that a user obtains a significant percentage of the downloads that he or she desires, with each of a plurality of Web sites having an opportunity to deliver content to a client during an automatic download session.

The methods described herein are particularly suited to providing a Web appliance with an off-line browsing capability, and more particularly to allow efficient off-peak Web browsing for such a Web client appliance, for

example, a data processing system connected to a conventional television.

Thus as describe herein Web content may be retrieved from a plurality of Web servers for delivery to a Web client connectable to the World Wide Web via a communication link. The Web client is preferably a data processing system connectable to a television or other conventional monitor to provide low cost Internet access. Initially a user defines a set of one or more servers from which content is desired to be retrieved and stored in the cache. These servers are preferably identified by a "list" of favorite Web sites. A test is then made to determine whether a given download period has terminated. Typically, this download period occurs during an "off" period, such as in the middle of the night, to avoid traffic congestion at the Web server sites. If the given download period has not terminated, a determination is then made of an activity level for the communication link as content is being downloaded to the cache from the one or more servers. If the activity level for the communication link is less than a given threshold level, additional requests for content are issued to the cache according to a so-called "fairness policy" that ensures that content from as many sites as possible is downloaded during the download period.

Thus, for example, according to the fairness policy the additional requests for content are issued to the set of one or more servers as one request per server in an ordered sequence. Alternatively, the additional requests for content are issued to the set of one or more servers up to a predetermined number of requests per server in an ordered sequence. Or, the additional requests are issued to the set of one or more servers based on the number of bytes received in the cache from a particular server. The fairness policy ensures that no one server dominates the download session to the exclusion of the other servers from which the user desires to download content. When content (i.e. a Web document) is received from a particular server, it is stored in the cache for off-line browsing. Prior to storage, the routine preferably removes duplicative and/or non-local HTML links so that subsequent access requests to the same document are handled more expediently.

A data processing system is provided to facilitate low cost Internet access. The system comprises two major parts: a remote control unit, and a base unit connectable to a monitor for providing Internet access under the control of the remote control unit. The base unit is, in effect, a computer, and includes a modem connected to a communication link, a processor, a memory, and various embedded control programs. These programs include a browser program including means responsive to commands from the remote control unit for generating a list of Web sites, and a cache control program for initiating download requests to the communication link based on the list of Web sites. An optimization routine maintains maximum utilization of the modem during a download session by issuing multiple HTTP GET requests to the cache control program based on the fairness policy.

Various embodiments of the invention will now be described in detail by way of example only with reference to the following drawings:

FIGURE 1A is pictorial representation of a data processing system unit connected to a conventional television set to form a "Web appliance";

FIGURE 1B is a pictorial representation of a front panel of the data processing system unit of Figure 1A;

FIGURE 1C is a pictorial representation of a rear panel of the data processing system unit of Figure 1A;

FIGURE 1D is a pictorial representation of a remote control unit associated with the data processing system unit of Figure 1A;

FIGURE 2 is a block diagram of the major components of the data processing system unit;

FIGURE 3 is a representative "favorites" list created by a user as a result of browsing the World Wide Web;

FIGURE 4 is a representative server URL queue list for the favorites list of FIGURE 3;

FIGURE 5 is a flowchart of a preferred method of the present invention for optimizing communication link activity during an off-peak caching session;

FIGURE 6 is a flowchart of the process response routine that is executed for each Web document received in response to a URL request; FIGURE 7 is a block diagram of a representative Web server platform or Web site; and

FIGURE 8 is a flowchart of the methods that are carried out by a Web server in response to receipt of a request from an Internet client such as the Web appliance described herein.

With reference now to the figures, and in particular with reference to FIGURES 1A through 1D, various pictorial representations of a data processing system are depicted. FIGURE 1A is a pictorial representation of the data processing system as a whole. Data processing system 100 in the depicted example provides, with minimal economic costs for hardware to the user, access to the Internet. Data processing system 100 includes a data processing unit 102. Data processing unit 102 is preferably sized to fit in typical entertainment centers and provides all required functionality, which is conventionally found in personal computers, to enable a user to browse the Internet. Additionally, data processing unit 102 may provide other common functions such as serving as an answering machine or receiving facsimile transmissions.

Data processing unit 102 is connected to television 104 for display of graphical information. Television 104 may be any suitable television, although color televisions with an S-Video input will provide better presen-

tations of the graphical information. Data processing unit 102 may be connected to television 104 through a standard coaxial cable connection. A remote control unit 106 allows a user to interact with and control data processing unit 102. Remote control unit 106 emits infrared (IR) signals, preferably modulated at a different frequency from the normal television, stereo, and VCR infrared remote control frequencies in order to avoid interference. Remote control unit 106 provides the functionality of a pointing device (such as a mouse, glidepoint, trackball or the like) in conventional personal computers, including the ability to move a cursor on a display and select items.

FIGURE 1B is a pictorial representation of the front panel of data processing unit 102 in accordance with a preferred embodiment of the present invention. The front panel includes an infrared window 108 for receiving signals from remote control unit 106 and for transmitting infrared signals. Data processing unit 102 may transmit infrared signals to be reflected off objects or surfaces, allowing data processing unit 102 to automatically control television 104 and other infrared remote controlled devices. Volume control 110 permits adjustment of the sound level emanating from a speaker within data processing unit 102 or from television 104. A plurality of light-emitting diode (LED) indicators 112 provide an indication to the user of when data processing unit 102 is on, whether the user has messages, whether the modem/phone line is in use, or whether data processing unit 102 requires service.

FIGURE 1C is a pictorial representation of the rear panel of data processing unit 102 in accordance with a preferred embodiment of the present invention. A three wire (ground included) insulated power cord 114 passes through the rear panel. Standard telephone jacks 116 and 118 on the rear panel provide an input to a modem from the phone line and an output to a handset (not shown). The real panel also provides a standard computer keyboard connection 120, mouse port 122, computer monitor port 124, printer port 126, and an additional serial port 128. These connections may be employed to allow data processing unit 102 to operate in the manner of a conventional personal computer. Game port 130 on the rear panel provides a connection for a joystick or other gaming control device (glove, etc.). Infrared extension jack 132 allows a cabled infrared LED to be utilized to transmit infrared signals. Microphone jack 134 allows an external microphone to be connected to data processing unit 102.

Video connection 136, a standard coaxial cable connector, connects to the video-in terminal of television 104 or a video cassette recorder (not shown). Left and right audio jacks 138 connect to the corresponding audio-in connectors on television 104 or to a stereo (not shown). If the user has S-Video input, then S-Video connection 140 may be used to connect to television 104 to provide a better picture than the composite signal. If television 104 has no video inputs, an external channel 3/4 modulator (not shown) may be connected in-line with the antenna connection.

FIGURE 1D is a pictorial representation of remote control unit 106 in accordance with a preferred embodiment of the present invention. Similar to a standard telephone keypad, remote control unit 106 includes buttons 142 for Arabic numerals 0 through 9, the asterisk or "star" symbol (*), and the pound sign (#). Remote control unit also includes "TV" button 144 for selectively viewing television broadcasts and "Web" button 146 for initiating browsing of the Internet. Pressing "Web" button 146 will cause data processing unit 102 to initiate modem dial-up of the user's Internet service provider and display the start-up screen for an Internet browser. The browser includes a "Favorites" or "Bookmarks" feature that enables the viewer to record the Uniform Resource Locator (URL) for those Web sites that the user desires to revisit.

A pointing device 147, which is preferably a trackpoint or button pointing device, is included on remote control unit 106 and allows a user to manipulate a cursor on the display of television 104. "Go" and "Back" buttons 148 and 150, respectively, allow a user to select an option or return to a previous selection. "Help" button 151 causes context-sensitive help to be displayed or otherwise provided. "Menu" button 152 causes a context-sensitive menu of options to be displayed, and "Update" button 153 will update the options displayed based on the user's input, while home button 154 allows the user to return to a default display of options. One of the options is the Favorites or Bookmarks list. A representative list is shown in FIGURE 3 as a pull-down menu 155 on the television screen. "PgUp" and "PgDn" buttons 156 and 158 allow the user to change the context of the display in display-sized blocks rather than by scrolling. The message button 160 allows the user to retrieve messages.

In addition to, or in lieu of, remote control unit 106, an infrared keyboard (not shown) with an integral pointing device may be used to control data processing unit 102. The integral pointing device is preferably a trackpoint or button type of pointing device. A wired keyboard (also not shown) may also be used through keyboard connection 120, and a wired pointing device such as a mouse or trackball may be used through mouse port 122. When a user has one or more of the remote control unit 106, infrared keyboard, wired keyboard and/or wired pointing device operable, the active device locks out all others until a prescribed period of inactivity has passed.

Referring now to FIGURE 2, a block diagram for the major components of data processing unit 102 in accordance with a preferred embodiment of the present invention is portrayed. As with conventional personal computers, data processing unit 102 includes a motherboard 202 containing a processor 204 and memory 206 connected to system bus 280. Processor 205 is preferably at least a 486 class processor operating at or

above 100 MHz. Memory 206 may include cache memory and/or video RAM. Processor 205, memory 206, and system bus 208 operate in the same manner as corresponding components in a conventional data processing system.

Video/TV converter 210, located on motherboard 202 and connected to system bus 208, generates computer video signals for computer monitors, a composite television signal, and an S-Video signal. The functionality of Video/TV converter 210 may be achieved through a Trident TVG9685 video chip in conjunction with an Analog Devices AD722 converter chip. Video/TV converter 210 may require loading of special operating system device drivers.

Keyboard/remote control interface unit 212 on motherboard 202 receives keyboard codes through controller 214, regardless of whether a wired keyboard/pointing device or an infrared keyboard/remote control is being employed. Infrared remote control unit 106 transmits signals which are ultimately sent to the serial port as control signals generated by conventional mouse or pointing device movements. Two buttons on remote control unit 106 are interpreted identically to the two buttons on a conventional mouse, while the remainder of the buttons transmit signals corresponding to keystrokes on an infrared keyboard. Thus, remote control unit 106 has a subset of the function provided by an infrared keyboard.

Connectors/indicators 216 on motherboard 202 provide some of the connections and indicators on data processing unit 102 described above. Other connections are associated with and found on other components. For example, telephone jacks 116 and 118 are located on modem 222. The power indicator within connectors/indicators 216 is controlled by controller 214.

External to motherboard 202 in the depicted example are power supply 218, hard drive 220, modem 222 and speaker 224. Power supply 218 is a conventional power supply except that it receives a control signal from controller 214 which effects shut down of all power to motherboard 202, hard drive 220 and modem 222. In some recovery situations, removing power and rebooting is the only guaranteed method of resetting all of these devices to a known state. Thus, power supply 218, in response to a signal from controller 214, is capable of powering down and restarting data processing unit 102.

Controller 214 is preferably one or more of the 805x family controllers. Controller 214 receives and processes input from infrared remote control 106, infrared keyboard, wired keyboard, or wired mouse. When one keyboard or pointing device is used, all others are locked out (ignored) until none has been active for a prescribed period. Then the first keyboard or pointing device to generate activity locks out all others. Controller 214 also directly controls all LED indicators except that indicating modem use. As part of the failure recovery system, controller 214 specifies the boot sector selection during any power off-on cycle.

Hard drive 220 contains operating system and applications software for data processing unit 102, which preferably includes IBM DOS 7.0, a product of International Business Machines Corporation in Armonk, New York; an operating system such as Windows 3.1 (or higher), a product of Microsoft Corporation in Redmond, Washington; and Netscape Navigator (Version 1.0 or higher), a product of Netscape Communications Corporation in Mountain View, California. Minor modifications of these software packages may be desirable to optimize performance of data processing unit 102. Also, it is highly desirable to update one or more of these "off-the-shelf" programs as well as the other software used by the present invention by downloading new versions of the code via the Internet. Web appliance includes appropriate control software to facilitate such downloading. Hard drive 220 also stores data, such as the list of favorite Internet sites or unviewed downloads from one or more Internet site(s). A cache controller program 225 run by the processor is used to administer and manage these downloads as will be described below.

Modem 222 may be any suitable modem used in conventional data processing systems, but is preferably a 33.6 kbps modem supporting the V.42bis, V.34, V.17 Fax, MNP 1-5, and AT command sets. To maintain the slim height of data processing system 102, modem 222 is preferably inserted into a slot mounted sideways on motherboard 202. Modem 222 is connected to a physical communication link 227, which, in turn, is connected or connectable to the World Wide Web of the Internet (not shown). As is well-known, the World Wide Web is the Internet's multimedia information retrieval system, based on the Hypertext Transfer Protocol (HTTP), which provides users access to files using Hypertext Markup Language (HTML). A link activity monitor 229 determines the extent to which the communication link 227 is being utilized at a given point in time. The link activity monitor may be a hardware-based controller or a software application run by the processor.

A Web server, sometimes referred to as a Web site, supports hypertext documents in directories and files accessible through the Uniform Resource Locator. Typically, all hypertext documents available at a particular Web site are considered part of the same domain (e.g., www.domainname.com). Pages that are local to the domain usually are identified by a relative link, which is a reference to a path and/or filename within the domain, e.g., www.domainname.com/path/html1. A representative Web server is illustrated in FIGURE 7 below.

Those skilled in the art will recognize that the components depicted in FIGURES 1A-1D and 2 and described above may be varied for specific applications or embodiments.

It is desired to enable a user of the data processing system to browse the Web "off-line". This function is provided by the cache controller 225. The cache controller may be a piece of dedicated hardware, or it may be an

application program run by the processor. In the preferred embodiment, cache controller 225 is implemented as a software program upgradable through Internet downloads.

As noted above, during one or more on-line browsing sessions, a viewer may compile a list of "favorite" or "bookmark" Web sites that he or she desires to revisit. All or any portion of this list may also be designated for access off-line so that the content of such sites may be downloaded and stored in a dedicated cache of the hard drive for later viewing, preferably off-line. The cache controller program thus includes a control engine 231 (preferably implemented in software run by the processor) for controlling the modem 222 to dial up and connect to the Internet site(s) automatically (e.g., each night while the appliance is unattended). As seen in FIGURE 4, each favorite Web site is associated with a server URL queue 235. A server URL queue 235 is a data structure that identifies the URL of the Web site as well as one or more HTML links spawned from (i.e. located within) the page. Preferably, the server URL queue 235 includes only relative links, although this does not have to be the case in all embodiments. Moreover, although the queue 235 is shown as a dedicated portion of the memory 206, this is not a requirement, as the queue may be a linked list or any other convenient data structure.

In a representative embodiment, the user will not have the ability to set the time period during which the engine will cache Internet site content; rather, this time period is predetermined by the network service provider. Generally, this time period will be restricted, e.g., one (1) hour per night. Therefore, the cache controller program 225 also includes an optimization routine to ensure that the modem 222 is used to its maximum capability during the restricted period of time that Internet sites are cachable to the hard drive 220. Moreover, the optimization routine includes a "load balancing" function to ensure that content identified by the server URL queues is equitably cached during the download period. As will be seen, this enables the viewer to obtain a significant percentage of the downloads that he or she desires.

This optimization routine is now described with reference to the flowchart of FIGURE 5. The primary processing of the routine begins at step 250 with a test to determine whether it is time to stop the process, i.e. whether the predetermined download period has expired. As discussed above, in an exemplary embodiment, this download period is one (1) hour, although it should be appreciated that other time period(s) may be used as well. This period may also be selectively adjusted if desired, but typically not by the user. If the outcome of the test at step 250 is positive, the primary processing routine is complete at step 252. If more time is available, then the routine continues at step 254 to query the activity on the communication link 227 to which the modem 222 is connected. Step 254 determines how much "bandwidth" is being used since a last iteration or cycle (of the routine) by receiving information from the link activity monitor 229. The routine then continues at step 256 to test whether the measured link activity meets some peak usage criteria.

As discussed above, goal of the present system is to maximize download throughput to the cache during the download period. The peak usage criteria generally is dependent on conditions on the communication link, the modem type, or such other criteria as may be predetermined or defined. Thus, for example, peak usage criteria may be defined by an average link utilization for a monitored interval exceeded by some preset limit between 0-100%. Or, the peak usage criteria may be based on some predefined limit on the number of outstanding HTTP GET requests that are issued from the cache manager to the network. A given HTTP GET request is used to request download of the content from a given Web site. Thus, for example, the peak usage criteria may be defined to include: not less than N total outstanding HTTP GET requests, not more than M total outstanding GET requests (M>N), and so on. It will be appreciated that any other convenient "peak usage" criteria may also be used in the comparison at step 256.

If the link activity meets the peak usage criteria, then the modem 222 is being used to its maximum capacity. As a result, the outcome of the test at step 256 is positive and the routine loops back to step 258, which is indicated as a delay. This box reflects that no more content requests are submitted. The routine then returns to step 250, as previously described.

If, however, the link activity does not meet the peak usage criteria, then, in effect, the modem is not being used to its maximum capacity. This is a negative outcome of the test at step 256. As a result, the routine continues at step 260 to obtain another URL from a server URL queue. The particular way in which this is accomplished will be described below, but it will be appreciated that this includes a balancing function to ensure that content identified by the server URL queue(s) 235 is cached equitably during the session. At step 262, a test is made to determine whether a URL was obtained from a queue. If not, the routine continues by testing at step 263 whether the total number of outstanding requests is greater than 0. If the result of the test at step 263 is positive, then the routine returns to the path of the delay 258 and returns. If the result of the test at step 263 is negative, meaning that no more outstanding requests exist, the routine is done and terminates. This outcome would occur, for example, if there were no more unserviced URLs on any server URL queue. If a URL was obtained at step 260, the outcome of the test at step 262 is positive, and the routine continues at step 264 to submit the URL request to the cache controller. Although not described in detail here, it should be appreciated that the cache controller then processes the request in a known manner to initiate the download process. The routine then continues at 266 to increment a

count of the number of outstanding requests. This number may be a total for the overall sessions, or a per server URL queue count, or both. After step 266, the routine returns again through the delay loop and recycles until the outcome of the test at step 250 indicates that the download session is complete.

It should be appreciated that the flowchart shown in FIGURE 5 is merely representative of the processing flow. The precise sequence flow illustrated is not meant to be taken by way of limitation. Thus, for example, the step of obtaining a URL (shown as step 260) could be carried out before querying the link activity (at step 254), and so on.

According to a feature of the preferred embodiment, it is desired to "load balance" the content downloads from the "favorite" Web sites so that the user obtains as broad a range of content as is possible during the restricted download period. As a result, Web servers that are slow or busy (even during the off-peak hour) do not impact adversely the downloading process. This is achieved by implementing a so-called "fairness policy" during the download process. In the preferred embodiment, this means that each server on the favorites list has an opportunity to deliver content to the client during the caching session. In particular, some policy designed to achieve equitable caching of the content identified by the one or more server URL queues 235 is implemented for this purpose. The specific policy may be as simple as a "round robin" policy under which a particular URL is taken off each server URL queue (in first in, first out order) irrespective of the number of relative links within a particular queue. Thus, for example, during a first iteration of step 260 discussed above, the first URL is taken from the first server URL queue. At the next iteration, the first URL is taken from the second server URL queue, and so on. Alternatively, the policy may start with a particular server URL queue but then limit the number of outstanding requests per server before requests for another server are used. In this example, the fairness policy would dictate that "never more than x outstanding requests per server" would be used. Thus, the first x URLs (which would include the home page and x-1 relative links) would be taken from the first server URL queue before using the URLs from a second server URL queue, and so on. Another fairness policy could be based on the total number of bytes received from a particular server, irrespective of the number of URL requests generated from a particular server URL queue. In this manner, the policy could restrict the total number of bytes cached from any particular server. Another approach to fairness would be limit the link depth of documents retrieved from a particular server. A link depth limitation would ensure that a page with a large number of nested or embedded links would not unnecessarily bias the download process to the exclusion of other servers. Still another approach would be to identify certain types of sites on the favorites list and serve such sites last because they would otherwise conserve too much of the download cycle. Thus, for example, a directory site (with thousands of links) might be identified and placed on the backend of the service list to ensure that the other sites get their chance to contribute to the download.

Of course, one of ordinary skill will appreciate that the above examples are not mutually exclusive and are not limiting, so that any fairness policy that has a goal of ensuring that each server on the favorites list has at least an opportunity to contribute to the download may be employed (irrespective of whether content is actually received by the client during the session). Thus, the above examples are merely representative of the types of criteria that may be used to implement a fairness policy. One or more of these examples may be combined to provide even more fine tuning of the precise amount of content cached from each of the favorite sites during the download session. The result of this load balancing approach is that each server at least has a theoretical and practical opportunity to contribute content; the user thus obtains a significant percentage of the favorite sites requested, even though some of the particular content may not be downloaded during the actual session. This is typically not a problem, however, since the cache will usually retain a copy (albeit possibly outdated) of the particular content that does not make it down to the cache during a particular download session. Thus, the viewer will still have a "complete" version of the favorite site, and the load balancing process ensures that he or she will have a broader scope of the overall list.

Referring now to FIGURE 6, a flowchart is shown of the process response routine that is run for each GET request submitted to the network by the cache manager at step 264 of FIGURE 5. It begins at step 270 by decrementing the count of the number of outstanding requests, since the particular request has been delivered to the network for service. At step 272, a test is run to determine whether a timer has expired for a particular request without the document being received. If the response to the test at step 272 is negative, the document has been received and the routine continues at step 274 to parse the document links. Although not meant to be limiting, during this step duplicate URLs or links that have already been visited are stripped from the document and thus are not returned back to the server URL queue. In addition, non-local links (except possibly images, e.g., .gif or .jpeg files) are also stripped from the document. Removal of such links ensures that more efficient processing of the document may be achieved during the next download session involving the document. Although the above processing is desired, other types of intelligent processing may be applied to the returned document. Thus, for example, the step may include a command to follow a certain type of string, or to limit the link depth of the document, or to apply a byte-count restriction, and so on. The particular type of intelligent processing is not limited in any way, and any technique that may be useful in facilitating the subsequent download process may be employed. After step 274, the

routine is done.

If the outcome of the test at step 272 is positive, which indicates that the document has not yet been received in response to the original request, the routine continues at step 276 to test whether a retry count has run a predetermined number of times. If so, the routine terminates. If not, the routine continues at step 278 to re-queue the request to the head of its particular server URL queue and to increment the retry counter for this particular request. By placing the URL request at the head of the server URL queue, it has a better chance of being re-serviced quickly (depending, of course, on the particular fairness policy being implemented). After step 278, the process terminates.

Turning now to FIGURE 7, a block diagram is shown of a representative Web server platform which supports content to be downloaded to the Web client. This platform is representative of a Web site. The platform 312 comprises an IBM RS/6000 computer 318 (a reduced instruction set of workstation) running the AIX Operating System 320 (Version 4.1 or above) and a Web server program 322, such as Netscape Enterprise Server Version 2.0, that supports interface extensions. The platform 312 also includes a graphical user interface (GUI) 324 for management and administration. The various models of the RISC-based computers are described in many publications of the IBM Corporation, for example, RISC System/6000, 7013 and 7016 POWERstation and POWERserver Hardware Technical Reference, Order No. SA23-2644-00. The AIX operating system is described in AIX Operating System Technical Reference, published by IBM Corporation, First Edition (November 1985), and other publications. While the above platform is useful, any other suitable hardware/operating system/web server combinations may be used.

The Web Server accepts a client request and returns a response. The operation of the server 322 is governed by a number of server application functions (SAFs), each of which is configured to execute in a certain step of a sequence. This sequence, illustrated in FIGURE 8, begins with authorization translation (AuthTrans) 330, during which the server translates any authorization information sent by the client into a user and a group. If necessary, the AuthTrans step may decode a message to get the actual client request. At step 332, called name translation (NameTrans), the URL associated with the request may be kept intact or it can be translated into a system-dependent file name, a redirection URL or a mirror site URL. At step 334, called path checks (PathCheck), the server performs various tests on the resulting path to ensure that the given client may retrieve the document. At step 336, sometimes referred to as object types (ObjectType), MIME (Multipurpose Internet Mail Extension) type information (e.g., text/html, image/gif, etc.) for the given document is identified. At step 338, called Service (Service), the Web server routine selects an internal server function to send the result back to the client. This function can run the normal server service routine (to return a file), some other server function (such as a program to return a custom document) or a CGI program. Our prime concern here is the situation where the server function runs a CGI program resident on the Web server. At step 340, called Add Log (AddLog), information about the transaction is recorded. At step 342, called Error, the server responds to the client when it encounters an error. Further details of these operations may be found in the Netscape Web Server Programmer's Guide, Chapter 5, which is incorporated herein by reference.

Thus, the Web server includes a known set of server application functions (SAFs). These functions take the client's request and other configuration data of the server as input and return a response to the server as output. Referring back to FIGURE 7, the Web server also includes an Application Programming Interface (API) 326 that provides extensions to enable application developers to extend and/or customize the core functionality thereof (namely, the SAFs) through software programs commonly referred to as "plug-ins".

In the preferred embodiment, the optimization routine comprises a set of instructions (program code) in a code module resident in the memory of the processor of the data processing system. Alternatively, the routine may be program code resident in a random access memory of a computer. Until required by the computer, the set of instructions may be stored in another computer memory, for example, in a hard disk drive, or in a removable memory such as an optical disk (for eventual use in a CD ROM) or floppy disk (for eventual use in a floppy disk drive), or downloaded via the Internet or other computer network. When the code is electronically delivered, a computer program product is said to comprise the program data (electronically delivered) stored in the substrate of a computer-readable storage medium such as the hard drive, floppy disk or other conventional storage media. In addition, although the various methods described are conveniently implemented in a general purpose computer selectively activated or reconfigured by software, one of ordinary skill in the art would also recognize that such methods may be carried out in hardware, in firmware, or in more specialized apparatus constructed to perform the required method steps.

It will be appreciated that the client described herein may be any suitable computer or other device directly or indirectly connected or connectable in any known or later-developed manner to a computer network, such as the Internet, intranet, extranet, etc., and may represent the data processing system described above, taken alone or in combination with a television set or other device.

Further, although the invention has been described in terms of a preferred embodiment in a specific environment, those skilled in the art will recognize that many modifications, variations in hardware and operating system architectures, and so on, are possible. Thus, for example, the mechanism described herein may be imple-

mented conveniently as a plug-in to a conventional browser program operating in a personal computer. Moreover, the preferred processing flow and steps may be altered. Thus, for example, it is not required that the process operate during a time-bounded downloading or caching session. The optimization technique may be conveniently applied in any Internet client that desires to cache content from a plurality of servers according to a fairness policy wherein each of the servers has an opportunity to contribute to the download.

Claims

1. A method of retrieving content for delivery to a client (100) connectable to a network via a communication link (227), the client including a cache, said method comprising the steps of:

   (a) defining a set of servers from which content is desired to be retrieved and stored in the cache;
   (b) determining (254) an activity level for the communication link as content is being downloaded to the cache from the servers; and
   (c) if the activity level for the communication link is less than a threshold level, issuing (260) to the cache additional requests for content such that each of the servers of the set has an opportunity to deliver content to the client.

2. The method of Claim 1 wherein the additional requests for content are issued to the set of servers as one request per server in an ordered sequence.

3. The method of Claim 1 wherein the additional requests for content are issued to the set of servers up to a predetermined number of requests per server in an ordered sequence.

4. The method of any preceding Claim wherein the additional requests for content are issued to the set of servers based on predetermined parameters.

5. The method of Claim 4 wherein the predetermined parameters include a link depth of a document located on a particular server.

6. The method of Claim 4 wherein the predetermined parameters include the number of bytes received in the cache from a particular server.

7. The method of any preceding Claim further including the step of processing the content received from a server to remove duplicative links.

8. The method of any preceding Claim further including the step of processing the content received from a server to remove non-local links.

9. The method of any preceding Claim wherein the threshold level of the communication link is a given number of outstanding requests for content.

10. The method of any of Claims 1-8 wherein the threshold level of the communication link is related to an average link utilization rate for a given monitored interval.

11. The method of any preceding Claim wherein the content is downloaded to the cache during a predetermined time period.

12. The method of Claim 11, futher comprising the additional step of determining (250) whether a given download period has terminated, wherein said step of determining an activity level for the communication link is performed if the given download period has not terminated; and wherein the additional requests for content are issued to the cache such that each of the servers at least has an opportunity to deliver content to the client during the given download period.

13. The method as described in Claim 12 wherein the given download period is selectable.

14. The method as described in Claim 12 wherein the given download period is approximately one hour.

15. The method of any preceding Claim, wherein said network is the World Wide Web, and said client and servers are a Web client and Web servers respectively.

16. A computer program product for retrieving content for delivery to a client (100) connectable to a network via a communication link (227), the client including a cache, the computer program product comprising: a computer-readable storage medium having a substrate; and a program encoded in the substrate of the computer-readable storage medium, wherein the program comprises means for performing the method of any preceding claim.

17. A computer comprising a modem (222) connected to a communication link (227); a processor (204); a memory (206) including a cache; a browser program running on the processor for providing World Wide Web information retrieval and including means for generating a list of Web sites to be downloaded; a cache control program (225) running on the processor and including means for initiating download requests to the communication link based on the list of Web sites; and a policy control program running on the processor and including

means for issuing multiple HTTP GET requests to the cache control program to ensure that each of the Web sites on the list has an opportunity to contribute content during a given download session.

18. The computer of claim 17, wherein the Web browser program includes menu means for generating the list of Web sites to be downloaded during an off-peak download session; the cache control program includes means for receiving downloads that are stored in the cache for off-line browsing; and the cache control program ensures that each of the Web sites on the list has an opportunity to contribute content during a given download session.

19. A data processing system (100) comprising a remote control unit (106) and a base unit (102) connectable to a monitor (104) for providing Internet access under the control of the remote control unit, the base unit comprising the computer of claim 17 or 18.

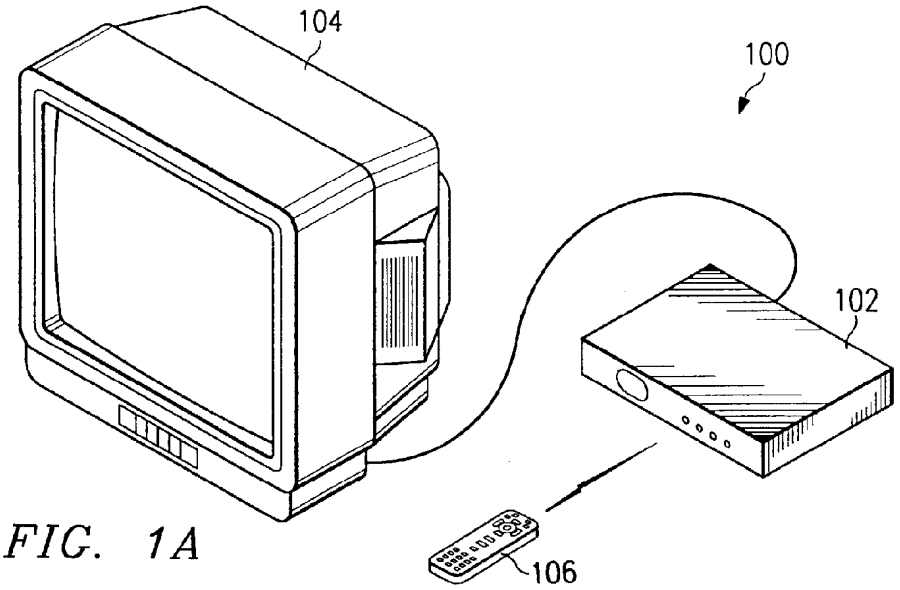


FIG. 1A

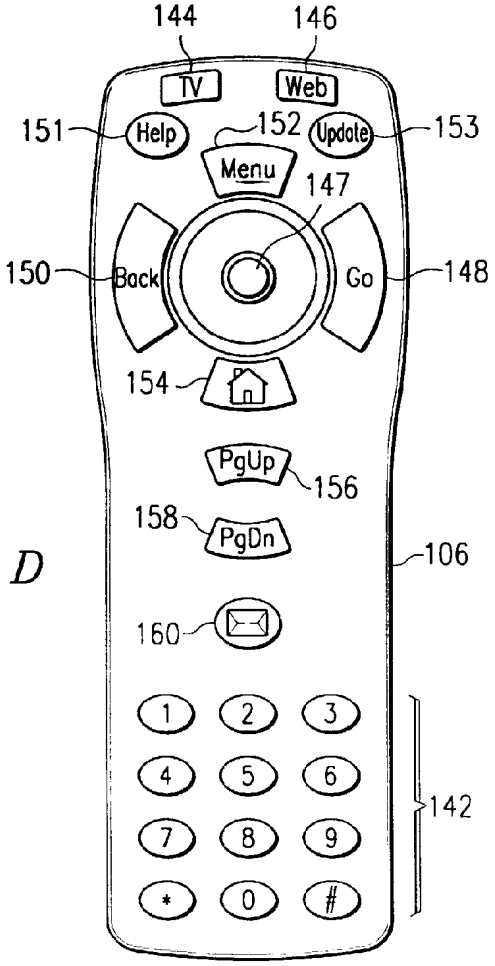


FIG. 1D

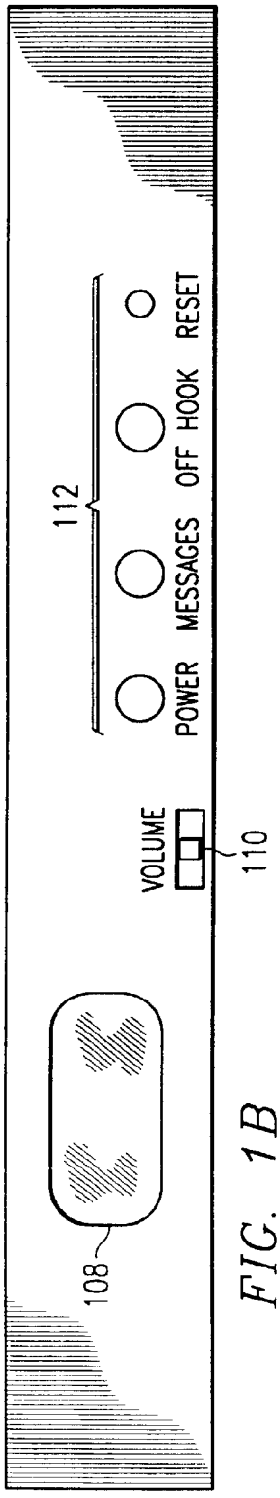


FIG. 1B

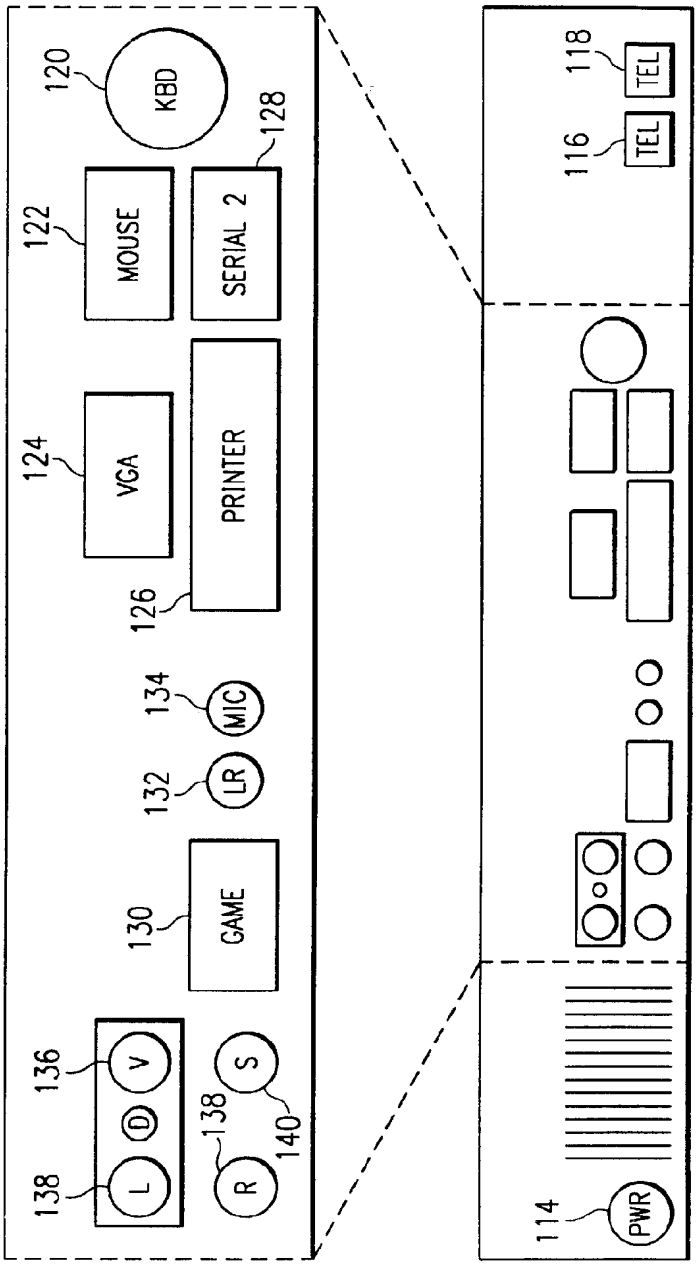


FIG. 1C

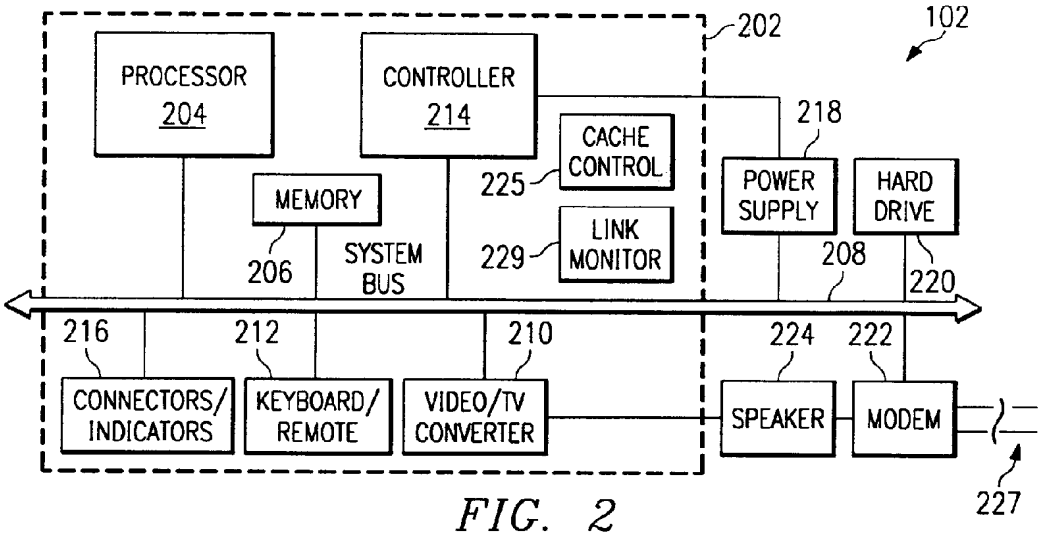


FIG. 2

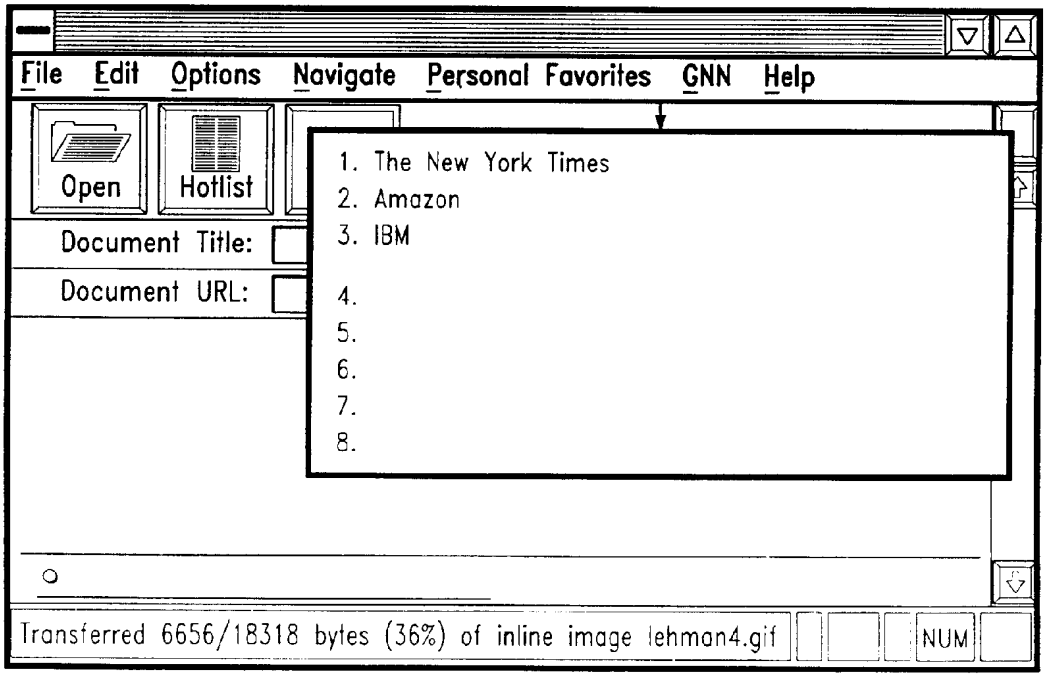


FIG. 3

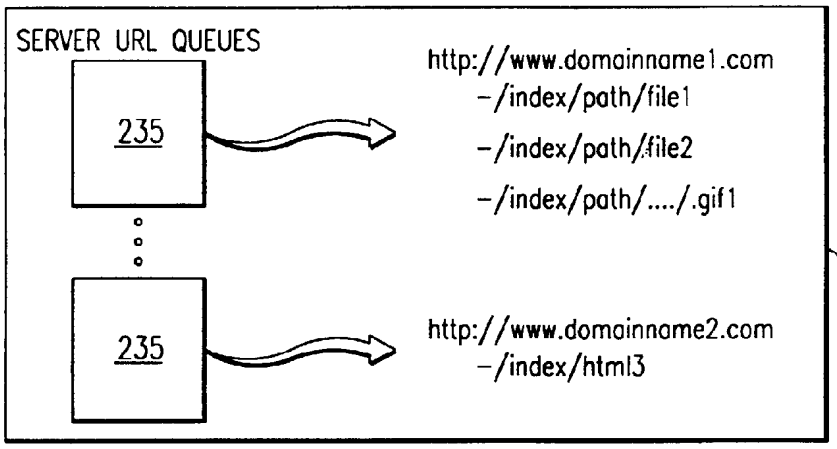
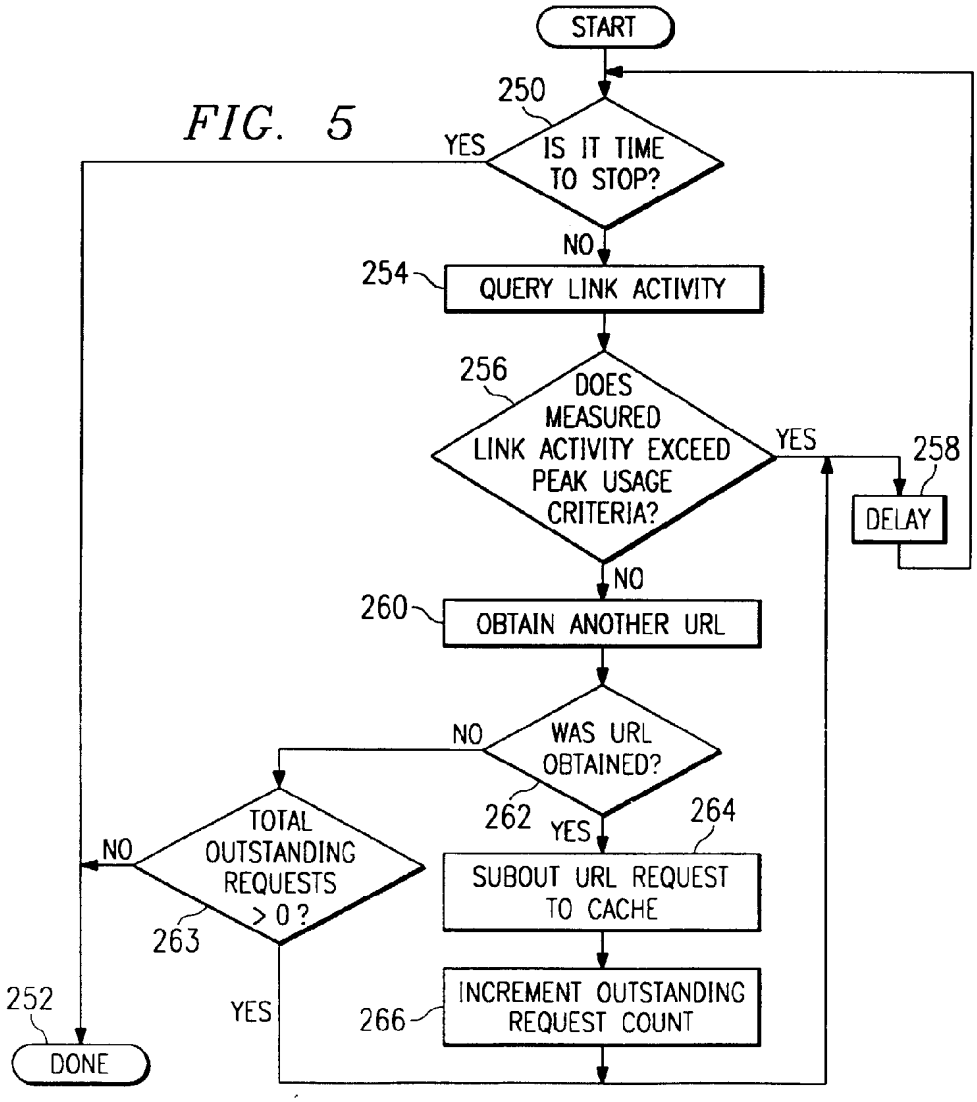


FIG. 4

FIG. 5



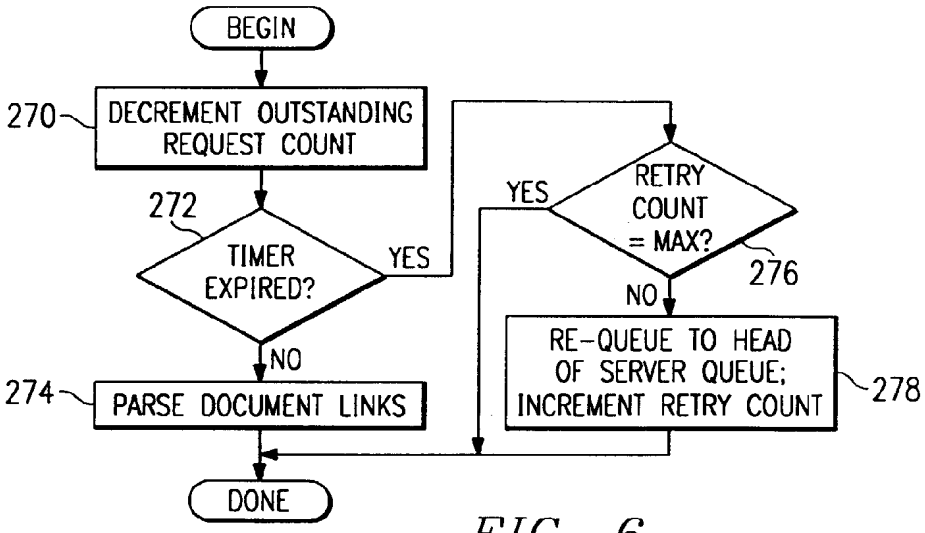


FIG. 6

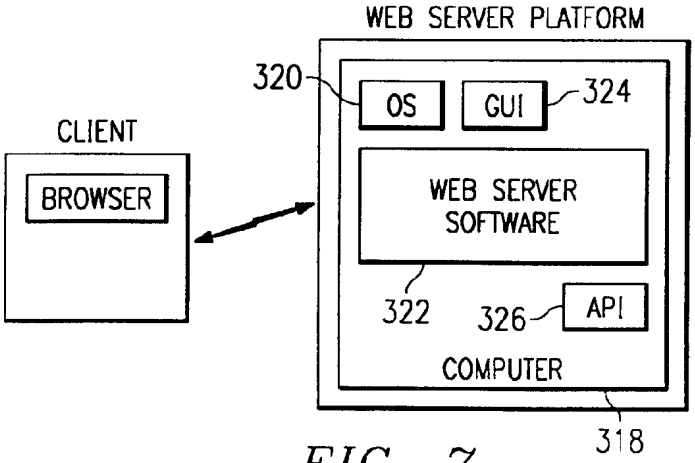


FIG. 7

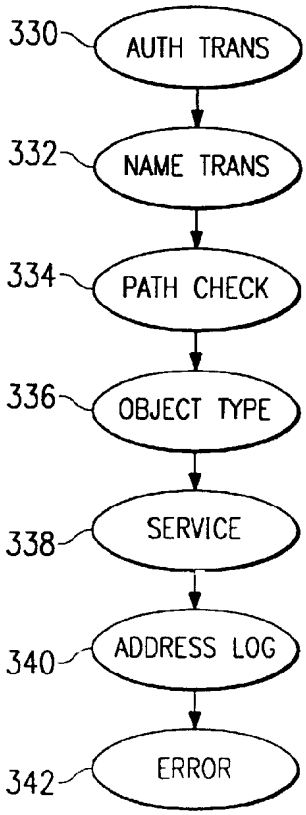


FIG. 8

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 883 068 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
09.12.1998 Bulletin 1998/50

(51) Int Cl.[6]: **G06F 17/30**

(21) Application number: **98201847.5**

(22) Date of filing: **28.05.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **28.05.1997 US 47998 P**
21.08.1997 US 915662

(71) Applicant: **Home Information Services, Inc.**
New York, NY 10019 (US)

(72) Inventor: **Ranger, Dennis**
New York, NY 10019 (US)

(74) Representative: **Quintelier, Claude et al**
Gevers & Vander Haeghen,
Patent Attorneys,
Rue de Livourne 7
1060 Brussels (BE)

(54) Entity retrieving system and method for retrieving entities

(57) The present invention relates to an entity retrieving system and a method for retrieving entities. The system and method according to the invention comprises mutation means provided for cooperating with a memory, a processor and an interface and for establishing, for each particular entity to be retrieved upon request of a user, if said particular entity pertains to a dependent classes of the class identified by the user, and retrieving, upon establishing that said particular entity pertains to one of said dependent classes of said selected particular class, said additional properties of said dependent class.

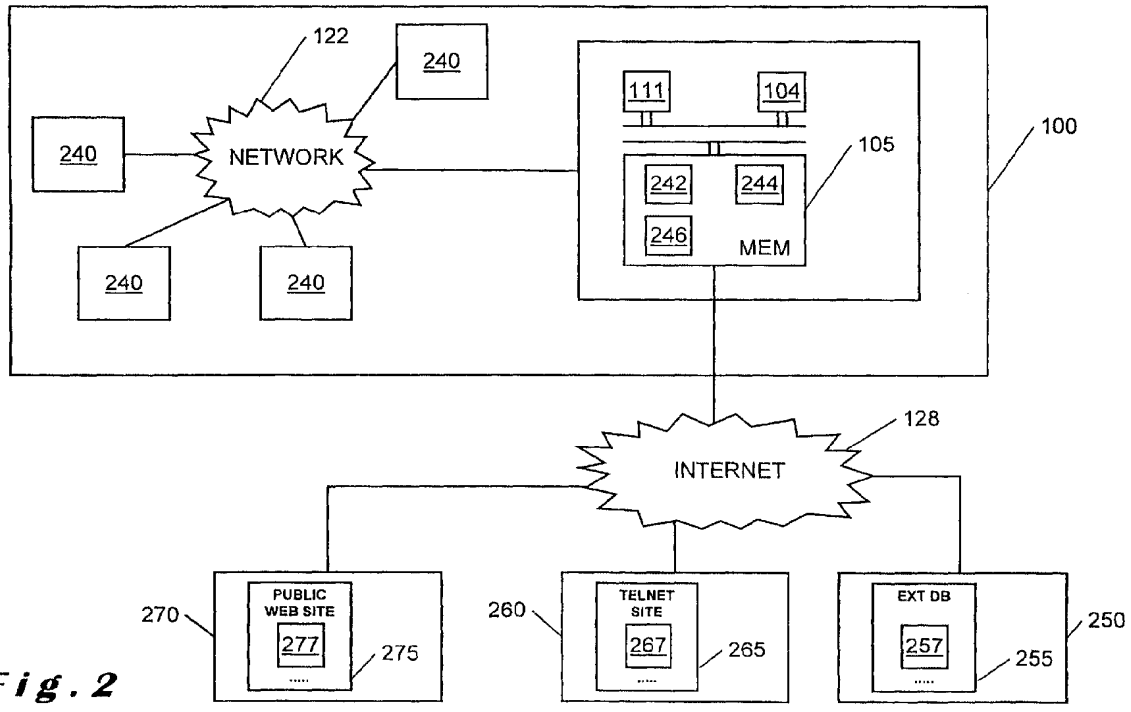


Fig. 2

EP 0 883 068 A2

(Cont. next page)

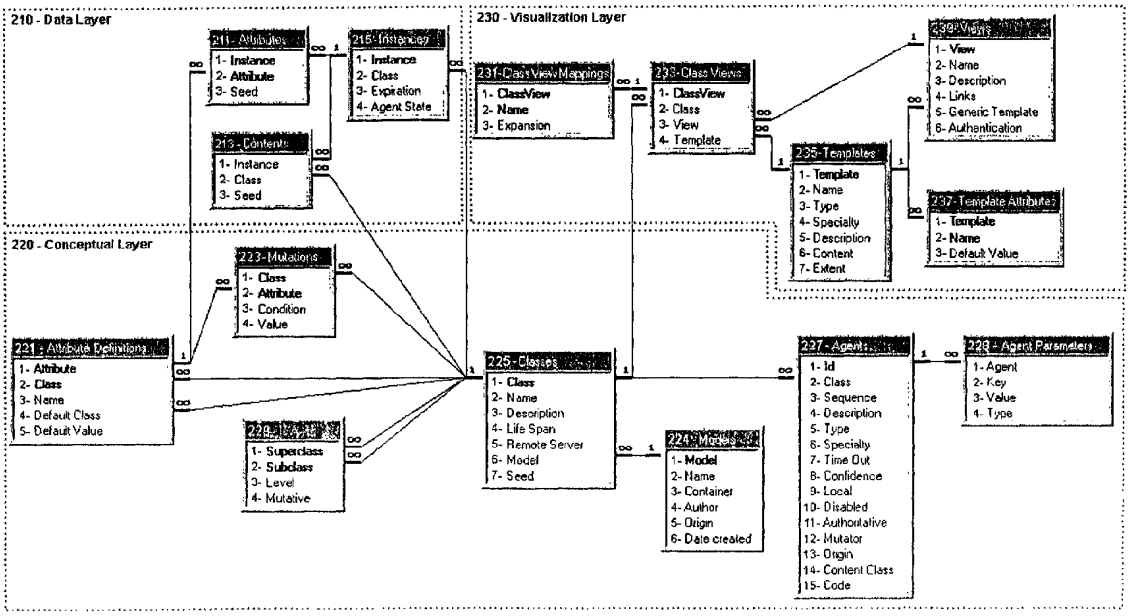


Fig. 3

Description

The present invention relates to an entity retrieving system connectable to at least one data source, said system having a memory into which a plurality of classes are stored, each class defining the structure of said entities, said structure comprising a plurality of property definitions, each property definition identifying property values to be retrieved dedicated to said property definition, said property values being stored in said data sources, said system further comprising an interface and a processor connected to each other, wherein :

(a) said classes comprise at least one dependent class hierarchically linked to at least one other class, said dependent class comprising additional property definitions specifying additional property values, in addition to the property values of the classes from which it depends,
(b) said interface is provided for receiving a query and supplying said query to said processor, said query comprising an identifier for identifying a particular class and at least one of said property values,
(c) said processor being provided, under control of said query, for selecting, among said classes, said particular class dedicated to said identifier, and
(d) said interface is further provided : i) for accessing said data sources, under control of said processor; ii) for retrieving property values pertaining to at least one particular entity that comprise said at least one of said property values; and iii) for outputting the retrieved entities.

Such a system is known from US patent 5,560,005, which enables to transform entities stored in a plurality of remote, heterogeneous structured relational databases into a homogeneous data model. In this patent, several classes are defined, e.g. "SERVICE PROVIDER" forming an abstract class and "HOSPITAL", "PHYSICIAN" and "PRACTISE" forming derived base classes or dependent classes hierarchically linked to the "SERVICE PROVIDER" class. Classes dependent from the dependent classes may also occur. This forms an hierarchic modelled structure of the classes into which property values or data may be incorporated. A query supplied by a user to the known system is formed by a request comprising the class name and search arguments forming thus the identifier to be supplied to the system. The system searches in the remote databases the table entries matching the provided search arguments and returns the property values of the table entries to the requester.

A problem with the known system is that the system will return a number of data, from the found particular entities, i.e. only the properties to be incorporated in that particular class. This means that if the user selects a particular class and the entity to be returned belongs in fact to a dependent class hierarchically dependent from the particular class, the number of properties returned to the user will be limited to the properties to be incorporated in that particular class. If the user wants to check if a particular entity pertains to a dependent class, he has to supply an additional query to the system, wherein the particular dependent class is mentioned. The more dependent classes there are and/or the more entities are found, the more time consuming this checking operation will be.

This problem is solved in the system according to the invention, which system further comprises mutation means provided for cooperating with said memory, said processor and said interface and for establishing, for each particular entity to be retrieved, if said particular entity pertains to one of said dependent classes of said selected particular class, and retrieving, upon establishing that said particular entity pertains to one of said dependent classes of said selected particular class, said additional properties of said dependent class. In case a user requests now for entities belonging to a particular class, the mutation means of the system will check if the found entity also pertains to one of the dependent classes of the particular class and, if this is the case, return the additional properties to be incorporated to the dependent class. The answer to the request of the user will therefore be more complete than expected, without requiring the user to manually check if the entity belongs to a particular class.

Preferably, agents are stored in the memory of the system, each agent being dedicated to at least one of said classes, each agent being provided for cooperating with said interface for accessing at least one particular property value pertaining to said particular entity of said class, each agent comprising : (a) an address for addressing one of said data sources from which said particular property values are to be retrieved; (b) a series of instructions indicating which data from the addressed data source are to be retrieved by said agent; and (c) at least one agent parameter for assigning, for each property value to be retrieved, a portion of said data to one of said property definitions. In particular, a reliability parameter is assigned to said agent.

Such a reliability parameter enables to indicate how trustworthy the addressed data source is. In case for example a first agent retrieves a first entity and a second agent retrieves a second entity, wherein the first entity is identical to the second entity except for one or more property values that are not equal to each other, the system may then, based on the reliability parameter of the agents, decide that one data source has provided a wrong property value or that there is one entity with an uncertainty concerning the property values that do not correspond or that those two entities in fact refer to two different entities.

According to a preferred embodiment, said system further comprises means for displaying the retrieved entities to a user and means for generating, upon receipt of said retrieved entities, at least one list, each list com-

prising a plurality of property value ranges for subdividing said retrieved entities, and said interface is further provided for enabling said user to select one property value range within one of said lists.

Such lists are very convenient when a large number of entities have been retrieved by the system. With known systems, the user would have to scroll through the results, without having an idea how the entities are structured. In this case, a logical structure of the retrieved entities is presented to the user, wherein the user can select a range. The entities belonging to that range will be shown to the user. This system enables thus to the user to view the results in a more structured way.

The present invention further relates to a method for retrieving entities by using a system according to anyone of the preceding claims comprising the steps of : a) receiving said query and supplying said query to a processor; b) selecting among said classes, said particular class dedicated to said identifier, c) accessing said data sources, under control of said processor, d) retrieving property values pertaining to at least one particular entity that comprise said at least one of said property values and e) outputting the retrieved entities. According to the invention, said method further comprises the step of establishing, for each particular entity to be retrieved, if said particular entity pertains to one of said dependent classes of said selected particular class, and for retrieving, upon establishing that said particular entity pertains to one of said dependent classes of said selected particular class, said additional properties of said dependent class.

The invention will now be described with reference to the annexed drawings, wherein :

Figure 1 illustrates schematically a preferred embodiment of the entity retrieving system according to the present invention.

Figure 2 illustrates schematically from which data sources the entities may be retrieved by the entity retrieving system according to an alternative embodiment of the present invention.

Figure 3 illustrates a database structure according to a preferred embodiment which is used for implementing the system according to the present invention.

Figure 4 illustrates an example of a number of classes with their hierarchy.

Figure 5 illustrates an example of the attributes and contents of a class

Figure 6 illustrates an example of an entity of the class according to Figure 5.

Figure 7 illustrates an example of agents returning inconsistent and ambiguous property values.

Figure 8 illustrates a display supplied to the user when a relatively large number of entities are retrieved by the system according to the present invention.

Figures 9 to 11 illustrate examples of agents.

As illustrated in Figure 1, the entity retrieving system 100 according to the present invention comprises a bus 102 to which a memory 105, an interface 111 and a processor 104 are connected. In particular, the memory is formed by a main memory 106, such as a random access memory (RAM), a read only memory (ROM) 108 and a storage device 110, such as a magnetic disk or optical disk. The interface comprises a display 112 for showing the retrieved entities to a user, an input device 114 such as a keyboard enabling a user to input a query, a communication interface 118 and optionally a cursor control 116 such as a mouse.

The system according to the invention is connectable to data sources, which may be internal to the system, for example data sources stored in the memory 105, and/or external to the system. For external data sources, use is made of the communication interface 118 for connecting the external data source to the system. Data sources may be located in a host device 124, connected to the system through the intermediary of the local network 122 and a network link 120. Data sources may further be stored in a server 130 connected to the network 122 through the intermediary of a communication network such as the Internet 128 which is accessible via an internet service provider (ISP) 126 or the like.

Figure 2 shows an entity retrieving system 100 according to an alternative embodiment, wherein data sources are stored. The retrieved entities may be represented to a plurality of user using a personal computer 240 with a user interface, in particular a web browser. The personal computers are connected, through the intermediary of network 122 to a network computer 126, in particular a web server, enabling a connection to the internet 128. The network computer comprises a memory 105 into which several databases are stored, for example spreadsheets 242, internal web sites 244 or other databases 246. External data source formats may be formed by external databases 257, telnet sites 267 or public web sites 277, stored in respective memories 255, 265 and 275 in respective computer systems 250, 260, 270. It is also conceivable, according to another embodiment of the present invention, to have for example video or sound data sources. The entity retrieving system of the present invention may be applied for retrieving entities stored in data sources having different formats, such as will be described further.

Referring to Figure 3, there is shown a possible database structure used when implementing the entity retrieving system according to the present invention. It forms a structured generic model or a so called "meta model" of a specific modelled structure, an example of which will be described when referring to Figures 4 to 6. It should be clear that this relational database structure is only one possible way to implement the present invention. In Figure 3, there are shown a number of tables, each table having a number of fields, each field defining a certain function. Many to one relationships between the fields are indicated by interconnecting lines with an indication "∞" on the many side and "1" on the one side. For example, instances 215 may have many attributes 211. Accordingly, there is an indication "∞" on the many

side of the Instance field 211-1 and an indication "1" on the one side of the Instance field 215-1.

In Figure 3, distinction is made between a conceptual layer 220 forming an intermediary between a data layer 210 and a visualisation layer 230.

CONCEPTUAL LAYER

The conceptual layer 220 comprises fields describing how data is organised within a defined model. The main part of the conceptual layer 220 is formed by the classes 225. A class 225 pertains to a model 224. The model table enables to support multiple models and dependent models. A specific model might be composed of a plurality of dependent models. For example, a banking model might have an accounting and a lending dependent model. The model table 224 comprises fields with the following meaning :

224-1 Model is a unique identifier, in particular a number for identifying the model. For example, the banking model has number "1001", the accounting "1101" and the lending model "1102"

224-2 Name indicates the name of the model in a human readable manner. In the example given above, this field would for example be "banking", "accounting" and "lending".

224-3 Container : If the model is a dependent model, this field contains the unique identifier of the model from which it depends. For example the "accounting" model would have in this field the unique identifier of the banking model, i. e. "1001". It should be clear that a dependent model may be dependent of a dependent model. For example, the accounting model may comprise several dependent models, which would have in this field the number "1101".

224-4 Author indicates the name or a user ID of the person that created this model.

224-5 Origin : If this model has been imported from a data source, the address of the data source is indicated here. In case the data source has been accessed through the Internet, it would be an URL.

224-6 Date Created indicates the creation date of the model.

Each model 224 has at least one class 225. Each class 225 is provided for defining the structure of entities to be retrieved. The class table 225 comprises the fields :

225-1 Class is a unique identifier for identifying the class, in particular a number.

225-2 Name is another unique identifier for the class but in a format convenient for human use. In particular it is formed by a string of characters, e.g. "book" or "product".

225-3 Description is provided for enabling the operator maintaining the system to add annotation and comments for this class.

225-4 Life Span indicates how long, for example in seconds, the entities belonging to that class should be kept in the memory of the system. For example a class in which the price of the entities is retrieved should have a relatively short life span, whereas a class from which the data is not quickly outdated may have a longer life span.

225-5 Remote server : In case a class is defined in another model and/or in a remote data source, the address of this model / data source is mentioned in this field.

225-6 Model contains the model unique identifier 224-1 to which the class is dedicated.

225-7 Seed indicates an Attribute Definition unique identifier, which is the attribute field 221-1 as will be described further, indicating which value is unique for each instance in the class. For example, a book may have as unique value its ISBN or ID number.

Each class can have a plurality of dependent classes or can be a dependent class from a plurality of classes. For this purpose, an Is A table 229 is provided for defining the hierarchy. It comprises the fields :

229-1 Superclass comprising the unique identifier of a class,

229-2 Subclass comprising the unique identifier of a dependent class of the superclass identified in field

229-3 Level showing the number of intermediate classes between the superclass 229-1 and the possibly indirect subclass 229-2, wherein level 0 indicates that the superclass and subclass are equal, level 1 indicates that subclass is a direct dependent class of the superclass, level 2 indicates that there is one intermediate dependent class between the subclass 229-2 and the superclass 229-1, etc.; this multiple level architecture improves the performance of the system and

229-4 Mutative indicating whether or not one or more mutation patterns or agents are dedicated to the class.

Each class has a plurality of attribute definitions. The attribute definitions table 221 comprises property definitions, with the following fields :

221-1 Attribute is a unique identifier for an attribute definition, eg. a number

221-2 Class identifies the class 225-1 that contains

this attribute as part of its structure

221-3 Name is an identifier of the attribute in text format

221-4 Default Class is an identifier of another class if the property contains a reference to this other class. For example a supplier attribute in a product class could refer to a supplier class.

221-5 Default Value comprises a default value in case the property value for this attribute is not found.

One or a plurality of mutation patterns can be dedicated to each class. The mutations table 223 comprises mutation pattern portion with one condition, for example book's pages is greater than or equal to 50. A mutation pattern may be formed by a plurality of conditions each condition being defined in a mutation pattern portion, for example book's pages is greater than or equal to 50 and price is less than $10. This table comprises the fields :

223-1 Class is the identifier of the class 225-1 to which the mutation pattern element is dedicated

223-2 Attribute is the identifier of the attribute 221-1 on which a condition applies, for example the attribute pages

223-3 Condition is an operator for example "greater than or equal to", "equal to", "not equal to", ...

223-4 Value is the value to which the property value should be compared, for example "50"

One or a plurality of agents can be dedicated to each class. The agents table 227 comprises the following fields :

227-1 ID is a unique machine readable identifier for the agent

227-2 Class identifies the class to which the agent is dedicated

227-3 Sequence is a number defining sequential order of invocation of the agents for a class (optional) It defines the agent's priority. If two agents are ready to be run, the one with the greater priority has precedence. The lower the sequence, the greater the priority.

227-4 Description is an annotation for providing a human readable description of the agent

227-5 Type specifies whether the agent is an attribute agent or a content agent. An attribute agent is provided for retrieving attributes while a content agent is provided for retrieving contents. The difference between contents and attributes is explained further when referring to the attributes and contents tables 211 and 213.

227-6 Specialty specifies the nature of the data source the agent queries, e.g. ODBC, Web, Corba, Telnet

227-7 Time Out indicates how long an agent should wait when the data source is not responding

227-8 Confidence indicates how trustworthy the property values retrieved from the data source (see origin 227-13) is

227-9 Local indicates whether or not the agent is local; if an agent is local then the agent is only used for the class to which it is dedicated, not in its dependent classes

227-10 Disabled indicates whether the agent is not to be used. This field is used for debugging and diagnostic purposes

227-11 Authoritative : if this field is yes and if this agent receives an empty response to its request, then the entity does not exist

227-12 Mutator indicates whether this agent is a mutation agent

227-13 Origin indicates the data source identity, in particular the path name of the data source from which the property values are to be retrieved

227-14 Content Class identifies the class 225-1 of the references returned by the agent, if the agent is a content agent.

227-15 Code comprises the instructions, forming a parameterised query, to be executed when running the agent

When an agent applies the code 227-15, data is returned comprising the requested property values. Each property value has to be extracted as a portion of the returned data. For this purpose, agent parameters are dedicated to the agent. The agent parameters table 228 comprises the fields :

228-1 Agent identifies the agent 227-1 to which the agent parameter is dedicated.

228-2 Key is a field dependent of the specialty 227-6 of the agent;
for ODBC agents, key is an index (e.g. 1,2,3,...) assigned to each portion of data returned by a query;
for Web agents, key is the identifier of the property definition to which the portion of data will be assigned

228-3 Value is a field dependent of the specialty 227-6 of the agent;
for ODBC agents, value is the identifier of the property definition to which the portion of data identified by the key field will be assigned
for Web agents, value is the pattern used for identifying the portion of data to be extracted and assigned to the property definition indicated by the key field 228-2

228-4 Type is a field dependent of the specialty 227-6 of the agent;
for ODBC agents, this field is not used for Web agents, type is a code indicating whether to

perform pattern matching on the HTML or on the text without the HTML tags.

For the purpose of clarity, an example of agents with parameters is given in Figures 9 to 11, wherein the reference numbers indicate in which field from Figure 3 the values are stored. Figure 9 gives an example of an attribute agent with specialty "ODBC", figure 10 a content agent with specialty "ODBC" and figure 11 an attribute agent with specialty "WEB". The fields provided below specialty are shown in function of the selected type and specialty.

VISUALISATION LAYER

The visualisation layer 230 is provided for comprising data from which a predetermined presentation of an entity is selected and produced. A view is here defined as what a group of users is allowed to see; it is represented as a set of templates attached to classes. It should be noted that some classes can have no template for a given view, meaning that the user has no access to the data requested or that there is a view to be inherited from one classes from which the dependent class depends, or that a default view has been assigned.

A class view table 233 is provided for determining a single template given a view and a class, or a single view given a template and a class or a list of classes given a template and a view. This table comprises the fields :

233-1 Class View is a unique identifier, e.g. a serial number, for a class view
233-2 Class is the unique identifier of the Class 225-1 to which the class view is dedicated
233-3 View is the unique identifier of the View 239-1 to which the class view is dedicated
233-4 Template is the unique identifier of the Template 235-1 to which the class view is dedicated

To each class view, one ore more class view mappings can be dedicated. The class view mappings table 231 holds variable substitution data. When a template is processed, for example as HTML or VRML generation, "value holders" such as "%supplier" are substituted by their values. A value holder can refer by name either to a class defined attribute, a class view mapping variable or a template variable. An attribute has precedence over a class view mapping variable which has precedence over a template variable. In other words, the value of a value holder in a template will default to the value of a template variable only as a last resort. The class view mappings table comprises the fields :

231-1 Class view is a unique identifier of the Class View 233-1 to which the class view mapping is dedicated.
231-2 Name is the name of the variable, for example supplier.
231-3 Expansion is a value of a variable, in particular a template.

Class views are dedicated to views. The view table 239 comprises the definition of a view. A view represents what a group of users is allowed to see. A view is a set of templates assigned to classes. Each template is retrievable from the class view table, given the view and a class. The view table comprises the fields :

239-1 View is a unique identifier for the view.
239-2 Name is the name of the view, for example "Inventory managers"
239-3 Description is provided for holding annotations
239-4 Links is the text of a default link template for the view, which is used when no template of the type link (see 235-3) has been dedicated to a view.
239-5 Generic template is a number identifying a default template 237-1 used when more than one entity is found; although the user has requested for one entity, for example one book; this can occur when there is a "conflict of opinion" as will be explained further.
239-6 Authentication indicates the name of a user's group if the present view is restricted to particular users. A password could be requested for some particular views. This password protection of views is performed with techniques known as such.

Each class view is dedicated to a template. The templates table 235 comprises data related for producing a presentation of an instance of a class, for example HTML, XML or VRML presentations. This table comprises the fields :

235-1 Template is a unique identifier for a template
235-2 Name is a text indicating the name of the template
235-3 Type indicates the type of the template, for example an object template (in particular an item), a space template (in particular a page), a link template (for representing a value of an attribute, in particular a hyperlink), ...
235-4 Specialty determines the presentation medium, for example HTML, VRML, XML, ...
235-5 Description enables the manager of the system to add comments and annotations
235-6 Content gives the actual text of the template, with embedded value holders
235-7 Extent indicates spatial dimensions for three dimensional objects or spaces for VRML presentations

Each template may comprise a number of template attributes. The template attributes table 237 comprises template variables used in value substitution, as explained with reference to the class view mappings table 231. The template attributes table comprises the fields :

237-1 Template is the unique identifier of the template 235-1 to which the template attribute is dedicated
237-2 Name identifies the name of the variable
237-3 Default value comprises, if applicable a default value of that variable

DATA LAYER

In order to improve response time, instances or entities retrieved by the system according to the present invention are preferably cached. The content and attributes of instances are stored separately. This enables to cache for example the attributes of an instance and not its contents or vice versa. An instance is cached no longer than is permitted by its class's life span 235-4.

The instance table 215 is provided for holding the instances or entities cached by the system according to the present invention. This table comprises the fields :

215-1 Instance is a unique identifier of an instance
215-2 Class is the identifier of the class 225-1 to which the entity or instance pertains
215-3 Expiration indicates the moment when a cached instance expires. This moment is calculated on the moment the data is retrieved plus the life span indicated in the life span field 225-4.
215-4 Agent state is a list of agent identifiers 227-1 that were used for retrieving the cached contents or attributes of an instance.

The contents table 213 comprises the content, retrieved by means of a content agent, of cached instances. A content is a list of references to instances of a given class. The content table comprises the fields :

213-1 Instance is the identifier 215-1 of the cached instance to which the content pertains
213-2 Class is the identifier of the class 225-1 of the cached content
213-3 Seed is a value from a list of values that makes up the content of an instance, that, together with the content class, specifies an instance of that class. The content seeds form a portion of property values to be retrieved.

The attributes table 211 comprises the attributes, retrieved by means of a content agent, of cached instances. This table comprises the fields :

211-1 Instance is the identifier 215-1 of the cached instance to which the content pertains
213-2 Attribute is the identifier of the attribute definition 221-1 to which the cached attribute pertains to
213-3 Seed is the cached value or property value of the attribute.

Figure 4 illustrates an example of a store model having a plurality of classes : "PRODUCT", "BOOK", "AUDIO TAPE" and "BEST-SELLER". Classes "BOOK" and "AUDIO TAPE" are dependent classes from the "PRODUCT" class. A best-seller is a particular book. Therefore, the "BEST-SELLER" class depends from the class "BOOK". In case a class "VIDEO TAPE" has to be added to the store model, it can easily be added as a dependent class of the "PRODUCT" class.

Each class has a number of property definitions, in particular attribute and content definitions. Referring to Figure 5, the "PRODUCT" class has for example the attribute definitions 221 with name 221-3 "ID", "SUPPLIER" and "TYPE", wherein ID is a unique identifier for a particular product, the supplier attribute is a reference to a supplier from the "SUPPLIER" class, and type describes the type of the product. ID and Type identify property values for simple data, e.g. numbers or strings, whereas Supplier identifies a property value referring to another entity, e.g. a Supplier entity pertaining to the Supplier class. The default class 221-4 field for this property definition will therefore be "SUPPLIER". The "BOOK" class is a dependent class from the "PRODUCT" class and inherits therefore all the property definitions from the classes from which it depends. In addition, it comprises additional property definitions, such as illustrated in Figure 5, i.e. the attributes "Title", "Author" and the contents "EDITION", "REVIEW", "CHAPTER 1", "CHAPTER 2".

Attributes are "single-valued" in the sense that each attribute has only one value. The title of a book is a single piece of data. Content properties, on the other hand, refer to open-ended lists of or references to other entities. For example, Chapters and Reviews are content properties of a book; they list the book's chapters and reviews. Content properties can also be inherited. For example, the Sales content property of a Book could be inherited from Product.

Figure 6 illustrates an example of a book entity having the following property values : "93-21123" as ID, "Doubleday" as Supplier, "Book" as Type, "War & Peace" as Title and "Tolstoy, Leo" as Author. The content properties refer to other entities. This entity is found by the system according to the invention after the user has input a request, for example :

http://www.server.com/query.pl?Product=93-21123&View=Customer For such a query, it is assumed that a product can only be retrieved based on its ID number. In this case, the seed field 225-7 would indicate the unique identifier of the attribute definition for ID number. The search possibilities could be en-

hanced by permitting to indicate in a query any property value pertaining to a class. The request would have then for example the following format :

http://www.server.com/query.pl?Product.ID=93-21 123&View=Customer

The interface 111 receives this query and supplies it to the processor 104 of the system. The processor will select from the memory the "Product" class, since the user has keyed in Product as particular class. An agent dedicated to the Product class, is provided for retrieving the Supplier and Type property values based on the ID number. These property values are for example stored in an internal data source, for example a relational database 246 (see Figure 2). The agent comprises an address in field Origin 227-13 indicating the path name of the database 246 data source. In order to enable to retrieve data from different types of data sources, there are provided different types of agents. For a relational database such as Oracle ®, the agent is an ODBC agent type. The agent further comprises a series of instructions indicating which data from the addressed data source are to be retrieved by the agent, for example :

"Select Key, Type, Supplier FROM Products"

The agent further comprises in its agent parameters 228 for assigning, for each property value to be retrieved, a portion of the data to one of the property definitions. In this case, "Key" is assigned to "ID" property definition, "Type" to "Type" property definition and "Supplier" to "Supplier" property definition.

This agent co-operates with the interface 111 for accessing the data source, under control of the processor 104 and for retrieving the requested data. In the example mentioned hereinabove, the following data will be returned : "93-21123" forming the ID, "Doubleday" forming the Supplier and "Book" forming the type. This data is stored in the memory of the system and will be provided to the user such as in known entity retrieving systems.

According to the invention, the system further comprises mutation means. The system will in particular check if mutation patterns or mutation agents are dedicated to one of the dependent classes. This can in particular be performed by verifying, for each dependent class, if the mutative field 229-4 is yes. In this case, there is checked if there are mutation patterns or mutation agents dedicated to the classes "Book" and "Audio Tape". A mutation pattern dedicated to the classes book comprises for example the condition : "If the Product Type = "book" then mutate the Product into a Book". The system thus verifies if the found property value for the product type falls within the condition. For this purpose, the processor compares the property values stored in the memory with the condition of the mutation pattern. In this case the retrieved property for the product type is a "Book". Consequently, a mutation occurs, wherein the class of the entity becomes "Book" and wherein the system will retrieve the additional property values pertaining to the class "Book".

In particular, retrieving the additional property values is again performed by one or several agents dedicated to this class. For example, an agent will be provided for retrieving a book's author and title from a web site given the book's ID. Since the data will be retrieved from a web site, the agent type is in this case "Web". The address in the Origin field is in this case an URL of the web site where the data should be retrieved, the instructions form in this case the steps required for accessing a web page where the requested data is shown. This is performed by providing the ID number "93-21123" and assigning this number to a corresponding parameter on the web site, for example LCCN Number. The agent further comprises agent parameters, by means of which pattern matching is performed, wherein the property values are extracted from web content by applying regular expressions, a known technique i.a. from "Mastering Regular Expressions" by Jeffrey E. F. Friedl, January 1997. For example the title property value could be found by searching on the page the expression "TITLE :" and looking for a series of words after the expression and located between spaces. It will be clear that this technique is applicable to all web pages having a predetermined structured presentation of the data. The ID is for example mentioned after the expression "LCCN NUMBER :" and the author after the expression "Author/Other Name :". In the example given, the following additional property values are retrieved by the system : "War & Peace" as title and "Tolstoy, Leo" as author.

Another agent could for example be provided for finding books chapters in a Telnet site given the book's ID. A further agent could be provided for accessing a relational database where book reviews are gathered from multiple sources, given a book's title and author. It should also be clear that the agents dedicated to a class to which the dependent class is hierarchically linked are also used. In this case, agents dedicated to the class products have already been processed, since book is a particular product and the user input the class "product" in his query. Now, the system has retrieved additional data, i.e. additional property values such as the title of the book, the author, reviews and chapters. This additional data is supplied to the user, although he had requested data relating to a product. The answer to the request of the user is automatically completed by the system according to the present invention with additional data which was unexpected by the user.

When several agents are provided for retrieving, from different data sources, property values that should correspond, it may occur that some property values retrieve are not equal to each other. For example, a customer's telephone number may be recorded differently in two data sources, or there might be three different authors for the same book title. In the first case, it is probable that the same customer has two phone numbers (an inconsistency), in the second case, we may be dealing with three altogether different books (an ambi-

guity).

Inconsistencies and ambiguities are unavoidable when integrating multiple data sources that were not conceived together and that may not even be managed by the same organisation. It is an object to deal in an appropriate way with ambiguities and inconsistencies within data. The manner in which the system according to the invention deals with those problems will be explained by means of an example.

Assume that agents are looking for a Person named Bob Smith. An agent A looks for a person's address given the person's name. Two agents B and C look for a person' age given the person's name, each agent targeting a separate data source. This example is illustrated in Figure 7.

The agent A returns with not one but two "Bob Smith", one living in New York and the other in Newark. The question is: Are we looking at two Bob Smith or only one but with a conflicting address? The answer depends on how much we trust agent A to be accurate or, in other words, how much we trust its data source to return correct addresses. For this purpose, a reliability or confidence parameter 227-8 is assigned to the agent. Let's say we trust agent A for 100%. Then we have no choice but to see two Bob Smith and two entities are thus shown to the user. In case agent A has a confidence parameter of only 10%, then the system will show only one entity showing two possibilities for a property value, e.g. "New York OR Newark".

Assume now agent A has a 100% reliability parameter. Agent B and C for the Bob Smith in New York obtain his age. Both agree that it is 35. However agents B and C for the Bob Smith in Newark disagree about his age. Agent B indicates 24 and agent C 27. Are we looking at two Bob Smith living in Newark? In this case, Agents B and C have been declared as fallible. That they disagree is not sufficient grounds to see two separate Bob Smith living in Newark; so we simply record the fact that there is a "conflict of opinion" between data sources about the age the Bob Smith living in Newark. Because of ambiguities and inconsistencies, a request to the system according to the invention to find an entity may end up returning more than one entity, with some "conflicts of opinion" about some of them. When this occurs, the user is presented with a display using the generic template 239-5 for the requested view, e.g. a Web page, that gives a choice between these entities and highlights conflicts.

In case agents B and C have substantially a same reliability parameter, which is relatively low, for example 10%, then one entity will be presented to the user with an indication of two property values for the age : "24 OR 27", such as illustrated in Figure 7. In case agents B and C have substantially the same reliability parameter, which is relatively high, for example 90%, then the systems interprets that there are two distinct entities as being two separate entities which will be presented to the user, each with its own age. In case agent B is substantially more reliable than agent C, for example agent B is at least 25 % more reliable than agent C, then the system will rely on the property value retrieved by agent B, i.e. 24, and only the entity retrieved having this value will be presented to the user.

Consequently, by providing a reliability parameter to the agents, the system interprets inconsistencies and ambiguities in property values, filters out unreliable property values and/or presents it in an appropriate fashion to the user. It should be noted that this provision of a reliability parameter could also be applied in other systems, in particular in systems without mutation means, and in general to any retrieving system provided for retrieving data from one or a plurality of data sources.

Referring to Figure 4, there is shown a dependent class "Best-seller" dependent from the class "Book". Now that the product requested by the user has been determined by the system as being a book, the system will further verify if the found entity should not further be mutated to the class "Best-seller". A mutation agent, i. e. an agent the mutator field 227-12 of which indicating that the agent is a mutation agent, which agent is dedicated to the class "Best-seller" is for example provided for accessing an external database or web site comprising a best-seller list. Based for example on the book's title and the author, the agent will search, in the same manner as explained hereinabove, in the addressed data source for the requested data. If the agent finds the requested book in the database, this means that the entity is a best-seller and the found entity is mutated to the class "Best-seller" as explained hereinabove. On the other hand, if the requested entity is not present in the best-seller data source, for example the agent receives a message such as "Could not be located", this signifies that the book is not a best-seller and therefore a mutation will not be performed.

The example of the mutation pattern described hereinabove had a single value condition with an operator "equal to". In general, it should be noted that all types of conditions are conceivable, with all types of operators such as "larger than", "between ... and ...". A list of single values is also conceivable. A mutation pattern could also have as condition "If all the other mutation patterns from the same level fail, then mutate to this dependent class". Such a mutation pattern signifies that the class to which the dependent classes of that level are hierarchically linked, is an abstract class, i.e. a class for which comprise no entities or instances. In the example given, if the store only sells books and audio tapes, the product class could then be an abstract class, since every entity is either a book or an audio tape. The mutation pattern for audio tape would be "If product type is not a book then mutate to audio tape class".

A further aspect of the present invention is the visualisation of the retrieved entities, in particular which property values are presented to the user based on its request. In the given example, the user has input in its query : "View=Customer", wherein all the found proper-

ties, except the supplier would be shown to the user. Another view could for example be a view for the staff, which view would require a password and provide in addition to the property values mentioned (including the supplier property value) an indication of the stock of the product in the store. An additional agent or above mentioned agent provided for retrieving the supplier and the type would be provided for retrieving in the internal database of the store, the numbers in stock of the requested product.

Based on the requested view, corresponding to views name 239-2, and the requested class corresponding to class name 225-2, a dedicated class view 233 is determined by the system, having a dedicated template 233-4. The content field 235-6 of the template table 235 will supply the instructions for producing the requested view to a user.

In general, once an entity has been recovered from the data sources, it can be shown to the user who requested it. How the entity is shown and how much of it is shown depends on the template that's used to generate the presentation. Each template is dedicated to a class. A dependent class can inherit presentations from its classes from which it depends or can define its own templates.

There are different kinds of templates, including :

a page template to display an entity as a full page, for example, a Book page will show the book's title, author, price, availability etc.;
an object template to display a summary of an entity in the page of another related entity, for example, a Book page would also show a list of its Reviews, each one summarized as a few lines and displayed using their object template; and
a link template to display an attribute value, in particular a hyperlink to an entity, for example, the summary of a Review listed in a Book page would include a hyperlink to the Review page where the full review can be read.

A user display, in particular a web page, is constructed on the fly from the templates. Each template is in fact a parameterised web page or VRML scene or other presentation to a user. The blanks are filled by the values of the entity's attributes and content properties before display. When the value filling a "blank" is a reference to another entity, then a hyperlink to that entity is automatically inserted. Hyperlinks are thus always correct and current, reflecting the data in the data sources at the time of the request.

If no template for a view on a class is defined (or inherited) for a given group of users, then no user belonging to that group can see entities of that class. For example, there would be no views defined on the class Inventory for customers, meaning that customers could not peruse the Inventory. If an attribute of a class is not shown in a template for a given group, then no member of that group can see the value of that attribute. For example, the views for class Employee would include the Salary attribute only for the Manager group (meaning that only managers can see employee salaries).

Between the moment an user requests to view an entity and the moment that entity is viewed for example as a Web page or a VRML scene or other presentation, a lot can happen as the relevant agents access data sources, triggering further agents until eventually all activated agents are done. Instead of waiting for all agents to have completed before displaying a Web page, a more dynamic approach may preferably be applied, a technique which is known as "server push", showing a Web page as soon as some data is available about the requested entity and then refreshing the page automatically when new data is retrieved. In particular, when mutation is applied, the user will first see data relating to the requested class and upon refreshing, the user will also see the additional property values. This way, the user does not have to wait too long to get a feedback and may elect to follow a hyperlink before the entity is completely shown and while the incomplete page is waiting to be refreshed with additional property values.

In a preferred embodiment, there is first checked which view is requested by the user and determined which properties should be supplied to the user. Based on this determination, only the agents required for the requested view are triggered, in order to supply more quickly the requested data to the user.

A further aspect of the present invention relates to supplying the results of a query input by the user in a more structured manner. Queries supplied to the system may lead in a large number of entities to be retrieved and presented to the user. With known systems, the user would be confronted with long lists of results, in particular hyperlinks, to scroll through.

The system according to the present invention takes advantage of the fact that the retrieved entities are dedicated to a structured model and thus that it has some understanding of entities. It is thus possible to organise long lists of entities. All entities belong to a class with defined properties. Using that knowledge, the system takes a long list and splits into smaller lists. Each smaller list represents entities falling within some range for a property value. For example, for a big list of Employees, the system could break the list down according to employee ID. A first sub-list would contain references to employees with IDs less than 236 and the other smaller list would contain references to employees with IDs greater than 342. This example is illustrated in Figure 8.

Since there is more than one property definition that can be used to create smaller lists, the system offers alternate subdivisions of the oversized content. This is in particular performed for each of the property values that may be viewed by the user. As illustrated in Figure 8, smaller lists are created for the first name, last name, city and state.

If the smaller sub-list is still too large for comfort, the

system applies the same operation again on the sub-list, until the user reaches a list small enough to be laid out in full. This is performed automatically for example by assigning to the system a predetermined maximum number of entities that the sub-list may not be exceeded.

Sometimes the amount of data that would be returned by an agent is so large that the system can only accept some of it from the data source and must discard the rest. Imagine what could happen if an agent returned a million references to Customers. The manageable portion that is accepted by the system is displayed using the same technique. As the user accesses restricted subsets of the original list, a more specific query is sent, yielding a smaller number of references. Again that smaller number, for example 100,000, may still be too large to be all taken in by the system. Eventually though, when the user has navigated to a narrow enough subset, the highly constrained query will return a complete yet manageable set of answers which can all be accepted and displayed by the system according to the present invention.

Consequently, such a subdivision of the results or automatic indexing provides to the user a logical structure of the retrieved entities, wherein the user can select a range. It should be noted that this provision could also be applied in other systems, in particular in systems without mutation means, and in general to any displaying system for displaying large sets of data.

In summary, when a user supplies a query to the user, the following steps are performed, according to a preferred embodiment of the present invention:

1. The query is received by the interface and supplied to the processor
2. If required, the user is asked to enter a user ID and a password for authentication, in particular when the requested view is a password protected view
3. The query is processed by the system
4. A plurality of data sources, addressed by the agents, are accessed to retrieve data pertaining to the requested entities, which data is mapped into property values as defined in the agent parameters of the agent.
5. The found entities are presented to the user according using templates dedicated to the requested view. In particular, this step may be performed before the previous step has been finished, in such a manner that the page presented to the user is dynamically updated when more property values are retrieved. In case the number of results is too large, then the list of results is subdivided in smaller list indexed according to several property values and presented to the user. The user may select one of these smaller lists.
6. Meanwhile, there is checked if mutation of each found entity should occur, using mutation patterns and mutation agents. If mutation occurs, additional property values are retrieved and presented to the user, by dynamically updating the users screen.
7. A new query may be input by the user to the system, in particular by selecting a link in the presentation supplied to the user.

The system according to the present invention improves the manner in which data is extracted from data sources, integrated into a model and presented to the user, in particular when the data has to be retrieved from a plurality of data sources which may have different formats.

Claims

1. An entity retrieving system connectable to at least one data source, said system having a memory into which a plurality of classes are stored, each class defining the structure of said entities, said structure comprising a plurality of property definitions, each property definition identifying property values to be retrieved dedicated to said property definition, said property values being stored in said data sources, said system further comprising an interface and a processor connected to each other, wherein

   (a) said classes comprise at least one dependent class hierarchically linked to at least one other class, said dependent class comprising additional property definitions specifying additional property values, in addition to the property values of the classes from which it depends,
   (b) said interface is provided for receiving a query and supplying said query to said processor, said query comprising an identifier for identifying a particular class and at least one of said property values,
   (c) said processor being provided, under control of said query, for selecting, among said classes, said particular class dedicated to said identifier, and
   (d) said interface is further provided

      i) for accessing said data sources, under control of said processor,
      ii) for retrieving property values pertaining to at least one particular entity that comprise said at least one of said property values and
      iii) for outputting the retrieved entities,

   characterised in that
   said system further comprises mutation means provided for cooperating with said memory, said processor and said interface and for establishing, for each particular entity to be retrieved, if said partic-

ular entity pertains to one of said dependent classes of said selected particular class, and retrieving, upon establishing that said particular entity pertains to one of said dependent classes of said selected particular class, said additional properties of said dependent class.

2. A system according to claim 1, wherein

(a) said mutation means comprise a mutation pattern dedicated to said dependent class, said mutation pattern comprising at least one condition, each condition assigning at least one predetermined property value range to one of said property definitions of said class to which said dependent class is hierarchically linked, and
(b) said mutation means are further provided for establishing if said particular entity pertains to said dependent class by verifying if the property value dedicated to said property definition of said particular entity falls within said predetermined property value ranges.

3. A system according to claim 1 or 2, into which memory agents are stored, each agent being dedicated to at least one of said classes, each agent being provided for cooperating with said interface for accessing at least one particular property value pertaining to said particular entity of said class, each agent comprising :

(a) an address for addressing one of said data sources from which said particular property values are to be retrieved,
(b) a series of instructions indicating which data from the addressed data source are to be retrieved by said agent,
(c) at least one agent parameter for assigning, for each property value to be retrieved, a portion of said data to one of said property definitions.

4. A system according to claim 3, wherein said mutation means are further provided

(a) for establishing if said agent comprises a mutation indicator indicating that said agent is a mutation agent, and
(b) for establishing if said particular entity pertains to said dependent class by verifying if said agent has established that said at least one particular property value pertaining to said particular entity is present, upon establishing that said agent comprises said mutation indicator.

5. A system according to claim 3 or 4, wherein a reliability parameter is assigned to said agent.

6. A system according to anyone of the claims 3 to 5, wherein said agent comprises a speciality indicator for indicating the type of the addressed data source.

7. A system according to anyone of the claims 3 to 6, wherein said agent comprises an authoritative indicator for indicating that if the property values retrieved by said agent are empty, then an indication that the entity does not exist is shown to a user of the system.

8. A system according to anyone of the preceding claims, said system further comprising means for displaying the retrieved entities to a user, said system further comprises means for generating, upon receipt of said retrieved entities, at least one list, each list comprising a plurality of property value ranges for subdividing said retrieved entities, and said interface is further provided for enabling said user to select one property value range within one of said lists.

9. A system according to anyone of the preceding claims, into which memory a plurality of models are stored, wherein each of said classes are assigned to one of said models.

10. A method for retrieving entities by using a system according to anyone of the preceding claims comprising the steps of:

(a) receiving said query and supplying said query to a processor;
(b) selecting among said classes, said particular class dedicated to said identifier,
(c) accessing said data sources, under control of said processor,
(d) retrieving property values pertaining to at least one particular entity that comprise said at least one of said property values and
(e) outputting the retrieved entities,

characterised in that said method further comprises the step of

(f) establishing, for each particular entity to be retrieved, if said particular entity pertains to one of said dependent classes of said selected particular class, and
(g) retrieving, upon establishing that said particular entity pertains to one of said dependent classes of said selected particular class, said additional properties of said dependent class.

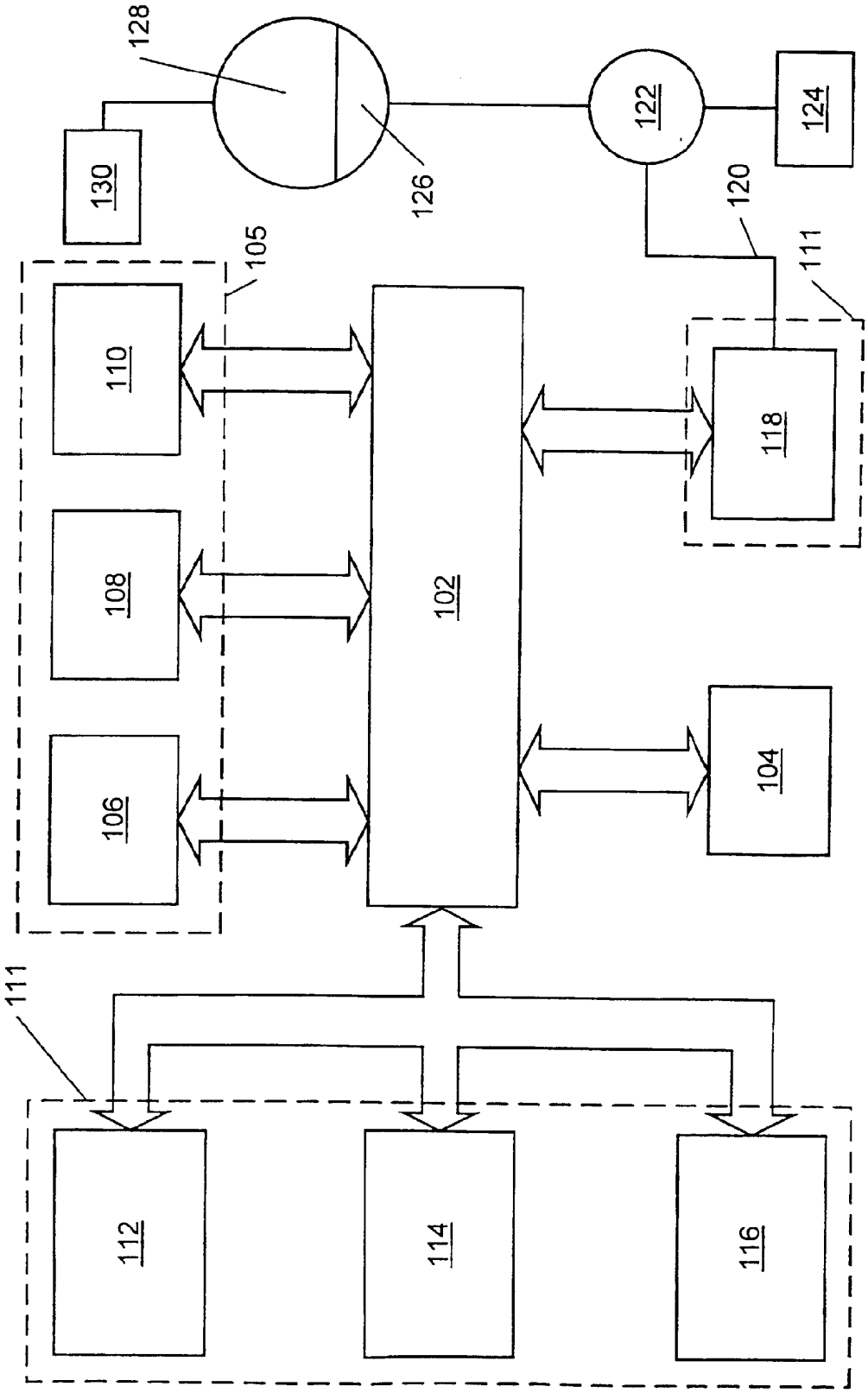


Fig. 1

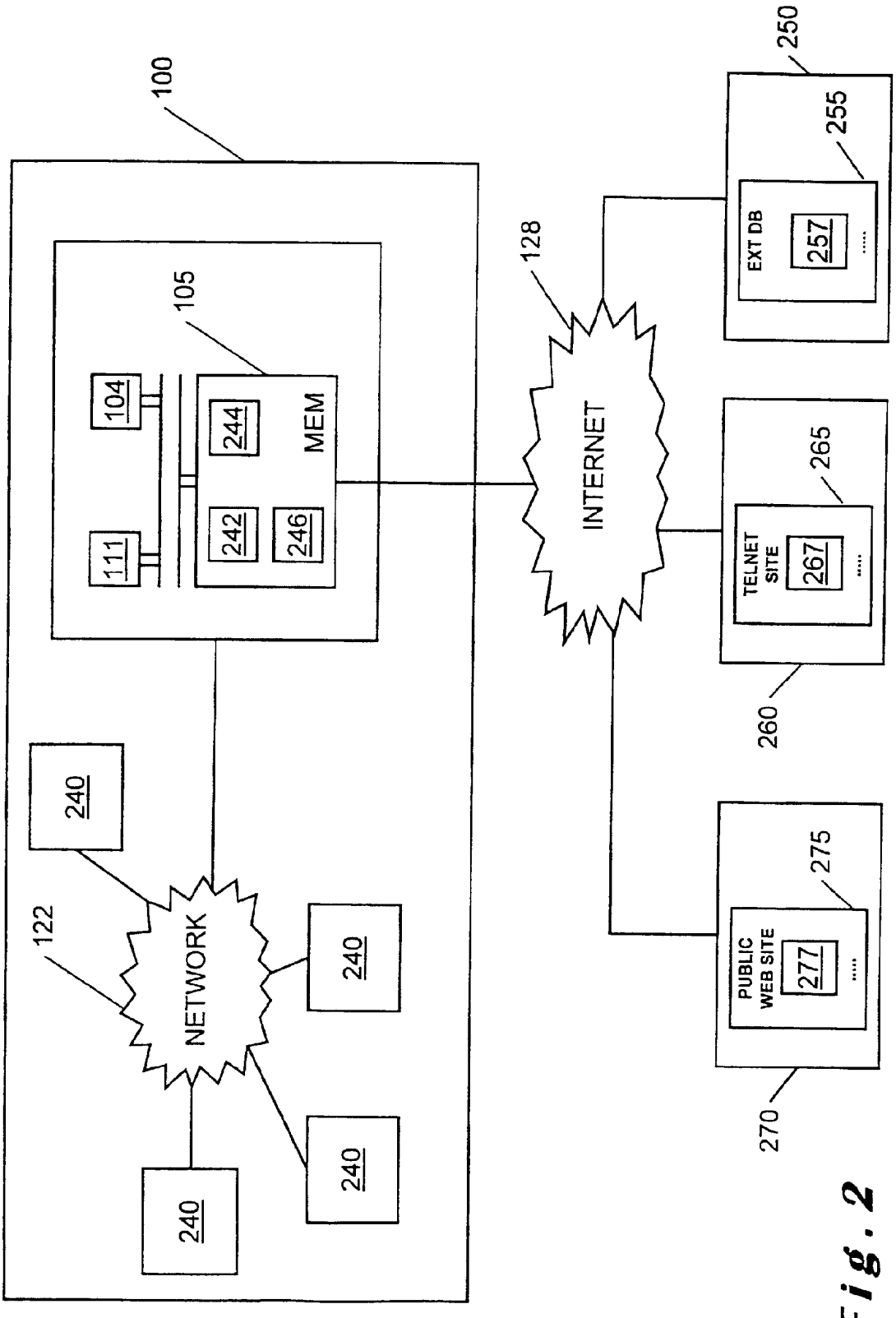


Fig. 2

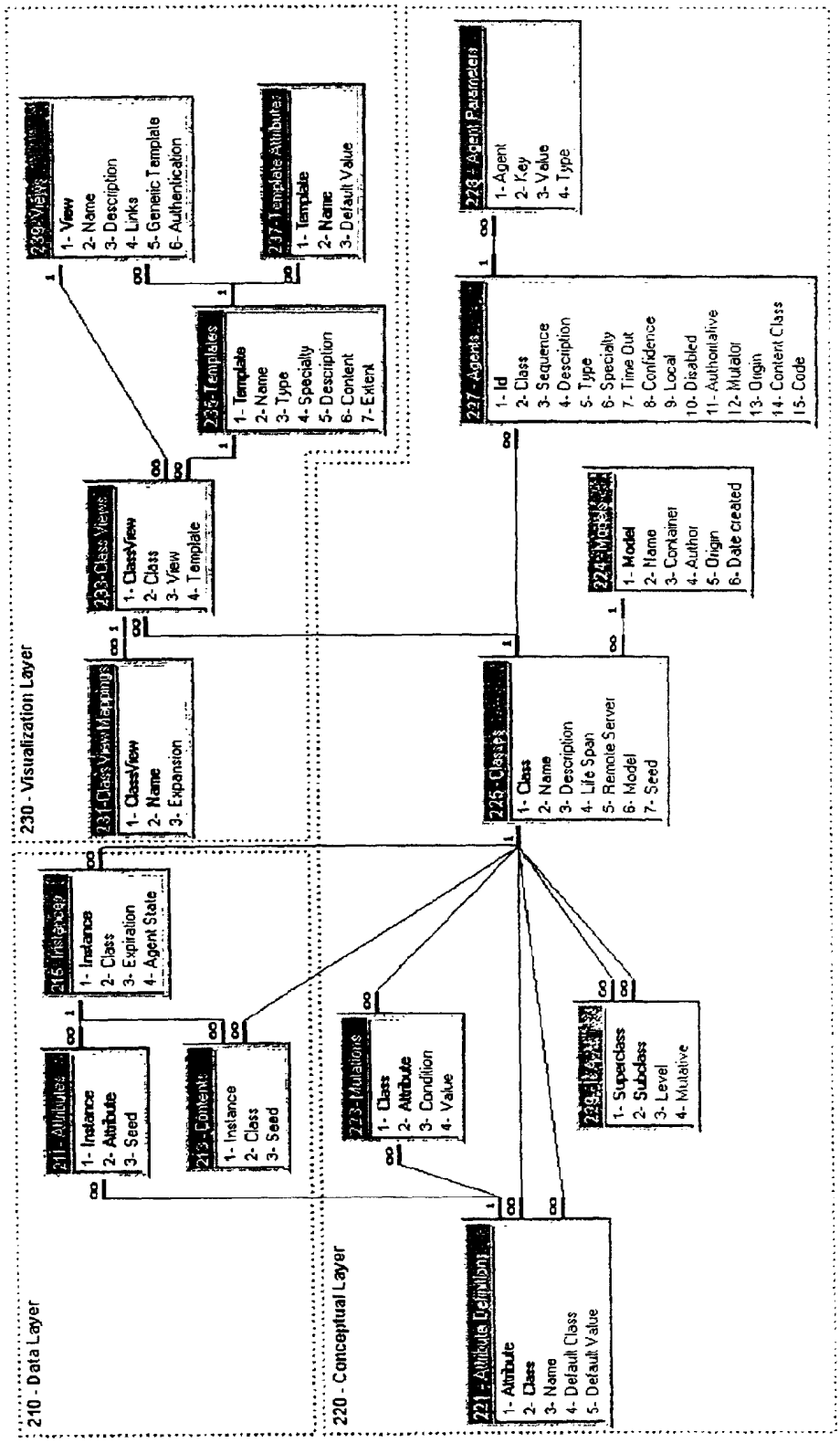


Fig. 3

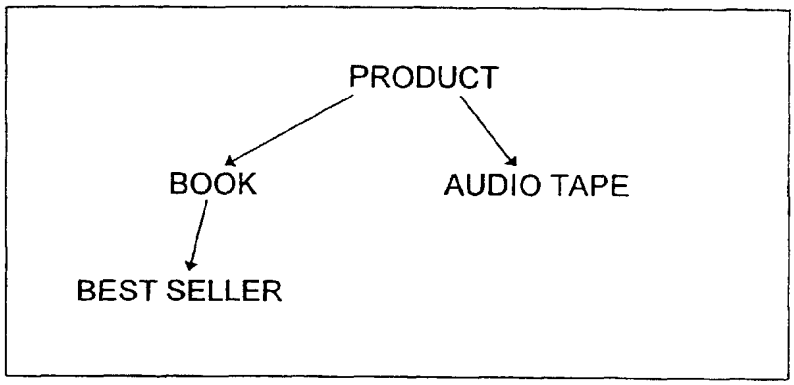


Fig. 4

BOOK	
ATTRIBUTES	DEFAULT CLASS
(product) ID	-
(product) Supplier	Supplier
(product) Type	-
Title	-
Author	-
CONTENTS	DEFAULT CLASS
Edition	Edition
Review	Review
Chapter 1	Chapter
Chapter 2	Chapter

Fig. 5

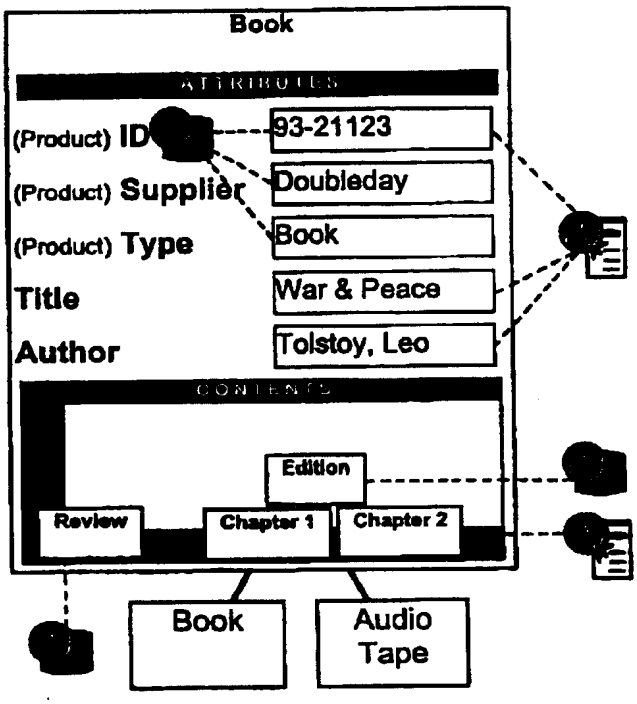


Fig. 6

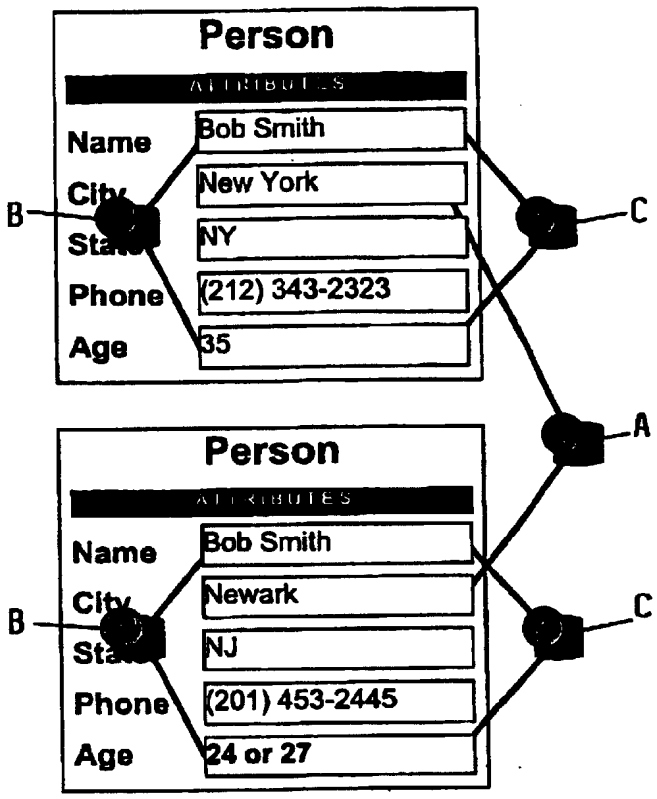


Fig. 7

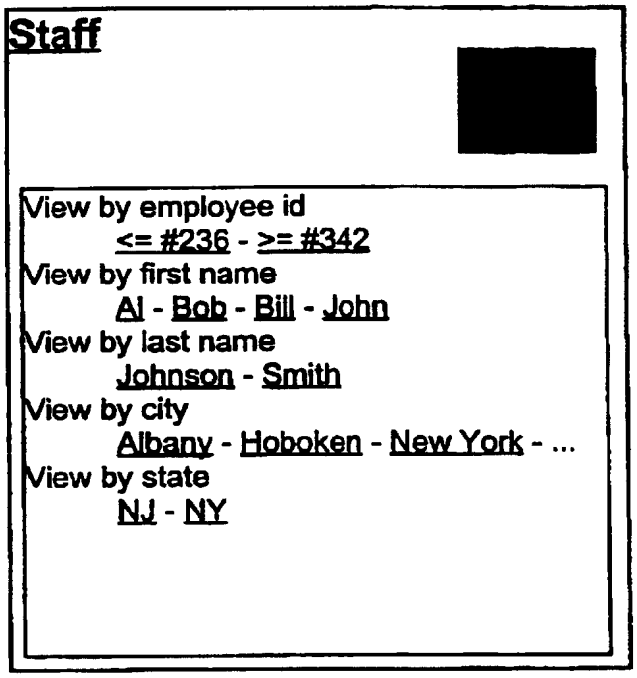


Fig. 8

PRODUCT (227-2) : agent # 20 (227-1)
Description (227-4) : Store database
Confidence (227-8) : 100 %
Time Out (227-7) : 3600 seconds
Sequence (227-3) : 10
Local (227-9) : No
Disabled (227-10) : No
Authoritative (227-11) : Yes
Mutator (227-12) : No
Specialty (227-6) : ODBC
Source (227-13) : Store
Code (227-15) : SELECT DISTINCTROW Key, Type, Supplier from Products
Columns : (228-2) (228-3)
1 = ID
2 = Type
3 = Supplier

Fig. 9

PRODUCT (227-2) : agent # 22 (227-1)
Description (227-4) : Store database
Confidence (227-8) : 100 %
Time Out (227-7) : 3600 seconds
Sequence (227-3) : 10
Local (227-9) : No
Disabled (227-10) : No
Authoritative (227-11) : Yes
Mutator (227-12) : No
Specialty (227-6) : ODBC
Source (227-13) : Library
Code (227-15) : SELECT Chap where Key = %ID from Chapterlist
Content class = chapter

Fig. 10

Book (227-2) : agent # 21 (227-1)

Description (227-4) : Library of congress

Confidence (227-8) : 100 %

Time Out (227-7) : 1 minute

Sequence (227-3) : 20

Local (227-9) : No

Disabled (227-10) : Yes

Authoritative (227-11) : Yes

Mutator (227-12) : No

Specialty (227-6) : Web

URL (227-13) : http://lcweb.loc.gov/cgi-bin/browse.pl

Parameters (227-15) :

tnaddress=locis.loc.gov
keystroke_file=tnlocis.txt
db=PREM
action=card
command=prem
slctoptne=
prefix=
screentitle=LCCN+Search+For:
searchtype=LCCN
startitem=1
userinput=@ID

Pattern :	(228-2)	(228-4)	(228-3)
	fail	Text	could not be located|Sorry
	Author	Text	\b(?:AUTHOR|OTHER NAME):\s*(.*?
	Description	Text	TITLE:\s*(1*?)\s*/
	...	...	...

Fig. 11

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 886 409 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
23.12.1998 Bulletin 1998/52

(51) Int. Cl.[6]: **H04L 29/06**, G06F 1/00

(21) Application number: **98107899.1**

(22) Date of filing: **30.04.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **01.05.1997 JP 113939/97**

(71) Applicant:
Digital Vision Laboratories Corporation
Minato-ku, Tokyo (JP)

(72) Inventor: **Muratani, Hirofumi**
Takatsu-ku, Kawasaki-shi (JP)

(74) Representative:
Lins, Edgar, Dipl.-Phys. Dr.jur.
Gramm, Lins & Partner GbR,
Theodor-Heuss-Strasse 1
38122 Braunschweig (DE)

(54) Information providing system

(57) An information providing system comprises an encryption unit for encrypting content data using a first key. The first key is included in message data which is associated with the content data and is separately transmitted to a user site. The message data is also encrypted using a second key within a security module. The second key is further encrypted using a third key within the security module. The third key is never read out to the outside of the security module.

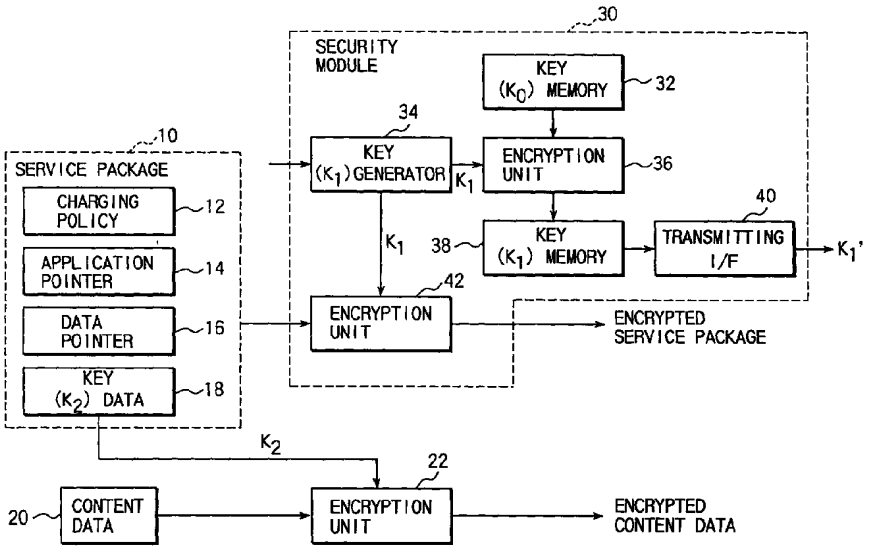


FIG. 2

EP 0 886 409 A2

Description

The present invention relates to an information providing system and more particularly to an information providing system which allows information to be provided readily while ensuring information protection.

The present application is based on Japanese Patent Application No. 9-113939, filed May 1, 1997, the content data of which is incorporated herein by reference.

With the advance of the Internet and the development of large-volume storage media such as DVDs and the like, various information providing services have been implemented regardless of online or offline. The information providers, which are companies that provide information as business, charge users for usage of information.

Forms of charging include charging on the basis of the amount of time that information is used, charging on the basis of the amount of information (the number of bytes) that is used, charging on the basis of a unit of information (for example, a movie), and so on. Under the present circumstances, only the forms of charging determined by the information providers are implemented. To be specific, in a closed system in which dedicated hardware is used, as in a cable television broadcasting service, a charging program is described in an application program installed in a data processing terminal (computer) on the user side or a server on the information provider side. In order to change the form of charging, therefore, it is required to rewrite the application program itself. It is thus not so easy to change the charging form. To provide a variety of forms of charging, it is desirable to add a new form or forms of charging and allow users to make a choice from the set forms of charging. However, this needs to modify the program considerably.

In addition, with the recent evolution of multimedia techniques, a case is also increasing in which one user subscribes to a number of information providers and receives a number of information providing services. In this case, an application program will be needed for each individual information providing service. The conventional charging facility, which is contained in an application program, cannot be commonly used in different application programs. For this reason, when an information service provider creates a new application program, it is also required to create a new charging program. However, the charging function is inherently independent of application programs and should be able to be used in common to different application programs. Preparing a charging program for each application program offers drawbacks that the program developing time is useless and each program increases in size and complexity.

The inventor of the present application proposed previously a system in which the usage of information (data processing function) and the charging process are separated from each other, the former being implemented by an application program, and the latter being implemented by a platform that differs from the application program (Japanese Patent Application No. 8-259,433). Here, the information provider separates information to be provided (hereinafter referred to as content data) or information, such as addresses, that identify content data and control information (referred to as a service description) required to utilize the content data or the information providing service. The service description includes information for identifying an application program utilizing content data to be provided, information for identifying a charging policy associated with the utilization of the content data, and information indicating a key needed to decrypt the content data in encrypted form. An example of the service description is such that a video playback application "A" is needed to utilize a video data "B", the charge for that video data "B" is 1000yen and the user must pay a fee to Mr. "C" in accordance with a settlement method "D". In this manner, users are allowed to utilize content data on the basis of the service description.

If the service description remained unprotected on the information transmitting path from an information provider to a user or at the user site, the service description might be altered. In such case, the information provider would be unable to collect a charge and have its digital rights infringed. The digital rights include a copy right or a counterpart right for service creation or service provision which should be belonged to the provider.

The provider's digital rights include the right of the service description as well as the copyright of the content data. For example, the information provider has rights to claim that "content data should be used in this manner", "content data should not be used in this manner", etc. For example, one who wrote a computer program can claim that the program may be run but no copying is allowed, or the program may be copied but no modifications are allowed and can define a charging policy such that the charge is 10yen per minute as the service description. The utilization that does not observe the service description constitutes an infringement of the digital rights. If the service description was not protected, then malicious users could rewrite the charging policy to thereby make the charges for information free. In such case, the information provider would suffer a great loss because the charging processor fails to work.

In order to protect the provider's digital rights, therefore, it is required to protect the service description as well as the content data. Like the content data, the service description is digital data and hence may be protected by encryption. That is, the content data and the service description are encrypted so that they cannot be interpreted at the time of utilization in the absence of a key, such as a token or ticket, issued by the information provider. The key is transmitted from the information

provider to the user via a protected secure path independently of the service description.

FIG. 1 is a block diagram of such a conventional system. A server 1 located on the provider side converts content data 3 into an encrypted form in an encryption unit 4 and then sends it to a terminal 2 located on the user side. An encryption key is generated by a key generator 6 and then transmitted by a key management unit 5 to the user device 2 over a secure path which is different from that for the data 3. On the user side, the key is stored in a key management unit 8 and the encrypted content data is stored in a decryption unit 7. Using the key in the key management unit 8, the content data is decrypted in the decryption unit 7, whereby content data 9 is made available.

However, even if the key is transmitted to the user site over a secure path, once the key is passed to the user or the user's application program, there arises the possibility that the service description after decryption may be altered at the user site. Thus, there is an essential drawback that the provider's digital rights may not be protected.

Even if the service description is passed to a user in an encrypted form that is not dependent on various content data transmitting forms such as broadcasting, on-demand, DVD, etc., a key is passed to the user on demand. It is not known when the user will make a service request. For this reason, the information provider is required to run the key issuing server all the time. This will cost the information provider and is not suitable for information providing service by individuals.

Accordingly, it is an object of the present invention to provide an information providing system which permits information to be provided readily while ensuring information protection.

According to the present invention, there is provided an information providing system comprising a provider device for providing information to users; a user device for utilizing information; and an information storage card adapted to be connected to the provider device and the user device and comprising means for storing a second key, in which the provider device comprises means for sending to the user device, a service package that describes information necessary for utilization of the provided information, the service package being encrypted in accordance with a first encryption system, and means for sending to the user device, a first key used in the first encryption system, the first key being encrypted using the second key which is stored in the information storage card; and the user device comprises means for decrypting the encrypted first key within the information storage card.

The service package after decryption is disabled from being retained within the user device or being output from the user device to outside.

The encrypted service package is decrypted within the information storage card and the decrypted service package is disabled from being output to outside of the information storage card.

The user device comprises service package decryption means for decrypting the encrypted service package and means for disabling the service package decryption means from decrypting the encrypted service package when it is not guaranteed that the decrypted service package should not be retained within the user device nor be output to the outside of the user device.

The service package comprises information for identifying information to be provided, information for identifying an application program that utilizes the information to be provided, and information indicating a charging policy relating to the utilization of the information to be provided, and the user device comprises an application program execution unit that operates in response to the decrypted application program identifying information, a charging unit that operates in response to the decrypted charging policy identifying information.

The application program execution unit is implemented by an application program, and the charging unit is implemented by a platform that is different from the application program.

The provider device comprises means for sending to the user device, a second charging policy identical to the charging policy contained in the encrypted service package without encryption.

The first key used in the first encryption system is generated in the information storage card.

The first key used in the first encryption system is generated by an authorized agent and is written into the information storage card.

The provider device comprises means for sending a ticket to the user device, the ticket associating information identifying the service package with information identifying a key used to encrypt that service package and the user device comprises means for identifying a key associated with the service package to be utilized on the basis of the ticket.

The information providing system further comprises a repeater unit for receiving the message data from the provider device and transmitting the received message data to the user device.

According to the present invention, there is provided another information providing system comprising a provider device for providing information to users; a user device for utilizing information; and a security module adapted to be connected to the provider device and the user device and comprising means for storing a second key in such a way that it cannot be read out to outside, in which the provider device comprises means for sending to the user device, a service package that describes information necessary for utilization of information, the service package being encrypted in accordance with a first encryption system, a first key used in the first encryption system being encrypted using the second key stored in the security module; and the user

device comprises means for decrypting the encrypted first key within the security module.

According to the present invention, there is provided an information providing device for providing information to users with an information storage card for storing a second key, the device comprising means for transmitting, a service package that describes information necessary for utilization of the provided information, the service package being encrypted in accordance with a first encryption system; and means for transmitting a first key used in the first encryption system, the first key being encrypted using the second key which is stored in the information storage card.

According to the present invention, there is provided still another information providing system for providing content data and message data in association with the content data, comprises means for calculating a first value of a unidirectional function or unidirectional hash function to which the whole message data or parts of the message data are input, a second value of a key generation function to which the first value is input, or a third value of a key generation function to which the first value and data included in the associated message or stored in the device are input, as a key.

The information providing system further comprises means for attaching the message data with a digital signature.

The message data contains charging information concerning a charge for usage of the content data.

The message data contains data described in a format including SGML, HTML, MHEG, or XML, and their extended or limited format.

According to the present invention, there is still another information utilization device for use with an information providing system in which content data and its associated message data are provided and the content data is encrypted, the device comprising means for calculating a first value of a unidirectional function or unidirectional hash function to which the whole message data or parts of the message data are input, a second value of a key generation function to which the first value is input, or a third value of a key generation function to which the first value and data included in the associated message or stored in the device are input, as a key.

The message data is attached with a digital signature.

The message data contains charging information concerning a charge for usage of the content data.

The message data contains data described in a format including SGML, HTML, MHEG, or XML, and their extended or limited format.

According to the present invention, there is provided still another information providing system comprising an information providing device which, in encrypting content data using an encryption key, uses a value of a unidirectional function or unidirectional hash function to which at least two parts of message data associated with the content data as the encryption key and transmitting the encrypted content data; a repeater unit for receiving the message data from the information providing device and transmitting the received message data; and an information utilization device which, in decrypting the encrypted content data transmitted from the information providing device using an decryption key, uses a value of a unidirectional function or unidirectional hash function to which the whole message data or parts of the message data associated with the content data and transmitted from the information providing device or the repeater unit as the decryption key.

The information providing device encrypts the message data, and the repeater unit decrypts the received encrypted message data, encrypts the message data again and transmits the encrypted message data.

The information providing device sends the message data with a provider's signature attached, and the repeater unit verifies the signature on the received message data and transmits the message data with a message data receiver's signature attached.

The repeater unit is in the form of the information utilization device.

According to the present invention, there is provided an encryption device for encrypting content data and its associated message data to be separately transmitted, the device comprising means for

calculating an value of a unidirectional function or unidirectional hash function to which the whole message data or parts of the message data are input; and

means for encrypting the content data using the value of the unidirectional function or unidirectional hash function as a key.

The invention can be more fully understood from the following detailed description when taken in conjunction with the accompanying drawings, in which;

FIG. 1 is a block diagram of a conventional information providing system;

FIG. 2 is a block diagram illustrating a system configuration on the provider side in accordance with a first embodiment of an information providing system of the present invention;

FIG. 3 is a block diagram illustrating a system configuration on the user side in accordance with the first embodiment of the present invention;

FIG. 4 is a detailed block diagram of the service instance of FIG. 3;

FIG. 5 is a block diagram of a security module in accordance with a second embodiment of the information providing system of the present invention;

FIG. 6 shows a security module for a user having a repeater function according to a second embodiment of the present invention;

FIG. 7 shows a security module for a repeater

according to the second embodiment of the present invention;

FIG. 8 shows a security module for an information provider having the repeater function according to the second embodiment of the present invention;

FIG. 9 shows a communication protocol between two security modules for two terminals;

FIG. 10 shows a system for an information provider according to a third embodiment of the present invention;

FIG. 11 shows a system for an agent or repeater according to the third embodiment of the present invention;

FIG. 12 shows a system for a user according to the third embodiment of the present invention;

FIG. 13 shows a modified system for the information provider according to the fourth embodiment of the present invention;

FIG. 14 shows a modified system for the agent according to the third embodiment of the present invention; and

FIG. 15 shows a modified system for the user according to the third embodiment of the present invention.

A preferred embodiment of an information providing system according to the present invention will now be described with reference to the accompanying drawings.

(First Embodiment)

Referring now to FIG. 2, there is illustrated an arrangement of a system on the information provider side in accordance with a first embodiment of the invention. In the present invention, as in the conventional system described previously, in order to allow the charging function to serve as a platform, a data processing unit, such as a server, on the information provider side creates a service package 10 that contains a pair of content data (name of the content data) the information provider provides and information (referred to as service description) required for control of the information or information representing the correspondence relationship between the content data and the service description. The service package 10 may include data described in accordance with a format such as SGML(Standard Generalized Markup Language), HTML(HyperText Markup Language), MHEG(Multimedia and Hypermedis Experts Group), XLM(eXtensible Markup Language), and their expanded or limited formats. The service package is such that MPEG data "D_1" (the name or address of content data) is encrypted using a key "K_2", processed by an application program "A_1", and subjected to a charging process "C_1". The user can actually make use of the content data on the basis of the service description in the service package 10. Thus, the service package 10 comprises a charging policy 12 indicating the form of charging, an application pointer 14 indicating an application program that utilizes content data, a data pointer 16 indicating the name or address of content data provided, and key data 18 indicating the key K_2 required to encrypt data. The charging policy 12 includes a usage fee, a usage condition, a paying method, or a payee.

Content data 20 provided is encrypted in an encryption unit 22 using the key K_2 and then transmitted to a user site over a transmitting interface (not shown). The encryption key K_2, which may be chosen freely by the provider, should preferably be one peculiar to the content data. Encrypted content data may be distributed online via the Internet or offline using DVD.

The encryption system may be either a common key system or a public key system. In the common key system, the key used to encrypt data and the key used to decrypt encrypted data are the same. In the public key system, on the other hand, the encryption key and the decryption key are different, and one of the keys is made public with the other kept secret. The provider encrypts data using the user's public key, and the user decrypts the encrypted data using his or her secret key. Thus, the public key encryption can be adopted only when each user is identified. According to the public key system, the content data is encrypted by using a public key K_2 so that a secret key corresponding to the public key must be transmitted to the user. Even with the colon key system, the key K_2 need not necessarily be contained in the service package 10 that is protected in accordance with the present invention. If there is a separate secure path, the key may be transmitted to the user over that path.

The service package 10 itself is encrypted for transmitting to the user. This transmitting may be made either online or offline. It should be noted however that the service package 10 is encrypted using a key K_1 which is different from the content encryption key K_2, and the key K_1 itself is also encrypted using still another key K_0 for transmitting to the user site. For encryption of the service package 10 and encryption of the encryption keys K_1 and K_2, in addition to the common key system the public key encryption system may also be used provided that each user is identified.

The key K_1 and the key K_0 for encrypting the key K_1 are kept absolutely unknown to the user. In the present invention, therefore, a security module 30 is used which is physically disabled from readout of data to outside. The encryption is performed inside the module 30. In order to increase terminal versatility, the module 30 should preferably be made of a semiconductor information storage card (a smart card, PCMCIA card, or the like) which provides the physical protection of data. However, if the terminal is implemented as a dedicated one, the module does not necessarily be removably mounted like a card, but can be fixedly mounted to part of the terminal.

The security module 30 comprises a key (K_0) mem-

ory 32, a key (K_1) generator 34, encryption units 36 and 42, an encrypted key (K_1') memory 38, and an encrypted key (K_1') transmitting interface 40. The key generator 34 generates an encryption key K_1 peculiar to the service package 10 in accordance with information that identifies the externally supplied service package 10. However, the key K_1 need not necessarily be generated within the module 30; it may be generated by a reliable organization and written into the security module 30.

The service package 10 is encrypted in the encryption unit 42 using the key K_1 and then transmitted to the user site via a transmitting interface not shown. The key K_1 used to encrypt the service package 10 is further encrypted in the encryption unit 36 using the key K_0 stored in the security module 30. The key (K_0) memory 32 consists of a nonvolatile memory. The key K_0 is stored at the time of creating the module 30 in such a way that it can never be accessed from outside at a later time. The key K_0 may be a key specific to the information provider irrespective of data/service package.

The encrypted key K_1' is stored in the memory 38. This is intended to omit work of encrypting the key K_1 each time the same service package 10 is encrypted. Thus, the key K_1' memory 38 can store keys K_1' obtained by encrypting encryption keys K_1, one for each of different service packages, using the key K_0. If, when a certain service package is specified, the corresponding key exists among the keys K_1' already stored, it is only required to read that key from the memory 38.

The key K_1' stored in the memory 38 is transmitted to the security module at the user site without being accessed from outside. When a semiconductor information storage card is used as the security module, the key K_1' is transmitted to the user's card in accordance with a card-to-card communications protocol. Thus, the key K_1' obtained by encrypting the service package encryption key K_1 is not output to the outside of the security module 30 nor does it become known to the user, preventing the service package 10 from being altered by a third party.

It is not required that the encrypted service package encrypted key K1' be transmitted simultaneously with the encrypted service package. As described previously, a key K_1 is peculiar to a service package. If, therefore, there exist multiple encrypted service packages and multiple encrypted service package encryption keys K_1' at the user site and the correspondence relationship between the service packages and the encryption keys is unknown, the encrypted service packages cannot be decrypted. It is therefore desirable to send from the provider to the user information that is used to establish a correspondence between information for identifying service packages and information for identifying keys K_1 used to encrypt those service packages. The information used to establish a correspondence is referred to as a ticket. By so doing, the user will be able to know from the ticket a key K_1 associated with a service package he or she wants to utilize.

FIG. 3 shows an arrangement of the terminal at the user site. At the user site as well, a security module 50 is used which preferably is made of a semiconductor information storage card. The security module 50 comprises a receiving interface 52, decryption units 54 and 60, a key (K_0) memory 56, a key (K_1) memory 58, and a service execution unit 62.

The user accepts the encrypted key K1' from the provider by secure module-to-module communications. The key K_1' will therefore not be transmitted to any user who does not subscript to the provider. The encrypted key K1' is supplied through the receiving interface 52 to the decryption unit 54. Like the provider's security module 30, the user's security module 50 has the memory 56 for storing the encryption key K_0. The memory 56, which is also a nonvolatile memory, is stored with the key K_0 at the time of creation of a card and is later made inaccessible from outside. Thus, the service package encrypted key K1' which was encrypted on the provider side using the key K_0 can be decrypted on the user side. Note that the public key encryption system may also be used here. The decrypted key K_1 is temporarily stored in the memory 58. This is also intended to omit work of decrypting the same encrypted key K_1' each time the same service package is decrypted. The memory 58 has an enough capacity to store keys K_1 corresponding to a plurality of service packages.

The encrypted service package is received by a receiving interface 64 and then temporarily stored in a service package memory 66. The encrypted content data are received by a receiving interface 68 and then stored in a content data memory 70. The encrypted service package is decrypted in the decryption unit 60 in the security module 50 using the key K_1 stored in the memory 58 and then entered into the service execution unit 62. The key K_2 contained in the decrypted service package is supplied from the security module 50 to a decryption unit 72. This decryption unit 72 decrypts the encrypted content data stored in the memory 70 using the key K_2. The decrypted content data is are delivered to a content reproducing unit (for example, a display unit) 74.

The user's terminal further comprises a user interface 76 and a service control unit 78. The service control unit 78 controls the key K_1 memory 58, the service package memory 66, and the service execution unit 62.

Referring now to FIG. 4, the service execution unit 62 comprises a charging module 82 which performs a charging process on the basis of the charging policy 12, an application program 84 which is run on the basis of the application pointer 14 and the data pointer 16, and a data transfer processing module 86 which allows content data 88 to be received in cooperation with the charging module 82 and the application program 84. The service execution unit 62 thus comprises hardware and software which are required to implement the information providing service on the basis of the service

description, and parameters that allow the hardware and software to work properly, i.e., the unit is a collection of facilities required to implement the information providing service.

The operation of the first embodiment will be described next. The first embodiment makes it a condition that the user uses the provider-distributed security module 50 having the key K_0 memory 56. An encrypted service package and encrypted content data are transmitted to the user site online via the internet or offline using a large-volume storage medium such as a DVD, i.e., in an arbitrary mode. When utilizing an information providing service, the user receives a key K_1' peculiar to the service package from the provider. The user plugs the security module 50 that has received the key K_1' into the terminal. In the module 50, the encrypted service package is decrypted by the decryption unit 60 and a service instance is produced from the service package. At the same time, the encrypted content data is decrypted by the decryption unit 72 using the key K_2 contained in the service package.

In the decrypted service package, the application program 14 and the data pointer 16 activate a predetermined application program 84. As the application program runs, the data transfer processing module 86 reads content data 88 from a server or storage medium to initiate the usage of the information providing service and the charging module 82 charges usage of the information providing service according to the charging policy 12.

As described above, according to the first embodiment, the content data 20 and the service package 10 are encrypted using separate encryption keys and then transmitted to the user. The encryption key K_1 used to encrypt the service package 10 is further encrypted using an additional encryption key K_0 and then transmitted to the user. This additional key K_0 used to encrypt the service package encryption key K_1 is held in the memory 32 in the security module 30 that cannot be accessed from outside and the encrypted key K_1' itself is directly transmitted to the user site on a module-to-module communications basis, in other words, in an externally inaccessible state. Therefore, the user and the application program cannot rewrite the service package, which prevents the charging policy from being altered for illegal utilization of services.

A modification of the first embodiment will be described. In FIGS. 2 and 3, the security module has only circuits that meet minimum requirements built in. If it has room, however, the encryption unit 22 (FIG. 2) and the decryption unit 72 (FIG. 3) may be built into the security modules 30 and 50, respectively.

In addition, the provider's security module 30 and the user's security module 50 may be arranged identically. An example therefor is illustrated in FIG. 5. A key (K_0) memory 100 is connected to an encryption/decryption unit 102 to which a key (K_1) memory 106 and a key (K_1') memory 108 are connected. A transmitting/receiving interface 110 is connected to the key (K_1') memory 108. Information identifying a service package is given to a key (K_1) generator 104, which generates a key K_1 used to encrypt that service package. The key K_1 is stored in the key memory 106 and supplied to a service package encryption/decryption unit 112 to which a transmitting/receiving interface 114 and a service execution unit 116 are connected.

In the module used by the provider, a key K_1 peculiar to a service package is generated or received from an external organization to encrypt the service package in the service package encryption/decryption unit 112. The resulting encrypted service package is transmitted over the transmitting/receiving interface 114 to a user site. At the same time, the key K_1 is encrypted by the encryption/decryption unit 102 using a key K_0 and the resulting encrypted key K_1' is transmitted over the transmitting/receiving interface 110.

In the module at the user site, the encrypted key K_1' received from the provider over the transmitting/receiving interface 110 is decrypted in the encryption/decryption unit 102 using a key K_0 and the decrypted key K_1 is then stored in the key (K_1) memory 106. The encrypted service package received over the transmitting/receiving interface 114 is decrypted in the service package encryption/decryption unit 112 for application to the service execution unit 116.

According to such an arrangement, the provider and the user are allowed to use the security modules of the same arrangement, providing an advantage of reduced cost. In this case as well, if there is room to accommodate more hardware in the security module, the data encryption/decryption unit may also be built into the module. In addition, if the user keeps a security module of the same arrangement as the provider's security module, there is no need for the provider to directly send the service package encrypted key K1' to the user. In such a case, the user will be allowed to send the key K_1' to other users. Further, the user can also send the key K_1' via security modules of a plurality of users, allowing the key to be communicated from individual to individual like word-of-mouth communication. This will eliminate the need of operating the key publishing server all the time and is therefore suitable for an information providing service by individuals. In this case, the users who merely repeats the key will not need all the hardware of FIG. 5, but requires only the key (K_1') memory 108 and the transmitting/receiving interface 110. However, when the public key system is used, the encrypted key K_1' must be decrypted once at each repeater terminal; therefore, in the arrangement of FIG. 5 it is only the service package encryption/decryption unit 112, the transmitting/receiving interface 114 and the service execution unit 116 that can be omitted.

(Second Embodiment)

Referring to FIGS. 6 to 8, there is illustrated an

arrangement of a second embodiment in which the service package and key can be repeated via a repeater. The key must be repeated using the security module. The service package is not necessary to be repeated using the security module since it is encrypted. The service package can be stored in a personal computer as a file and read out to be transmitted to the other repeater or user.

FIG. 6 shows a security module for the user having a repeater function. An input/output interface 202 receives the encrypted service package and the encrypted key K_1' from the information provider or repeater. The encrypted key K_1' is supplied to a key receiver 208 of a key management section 204. The key management section 204 comprises a key controller 206, key memory 210, and key transmitter 212 in addition to the key receiver 208. The encrypted key K_1' is written into the key memory 210 by the key receiver 208. The key k_1' read out from the key memory 210 is supplied to the key transmitter 212. The key transmitter 212 sends out the key k_1' via the I/O interface 202. Thus, the encrypted key K_1' is repeated by the user's security module.

The user's security module further comprises a service package decryption unit 214, a service execution unit 216, and a service control unit 218. The I/O interface 202 supplies the input service package (encrypted service package) to the package decryption unit 214 in which the encrypted service package is decrypted by using the key K_1' supplied from the key memory 210. The encrypted service package is supplied to the service execution unit 216 which causes the information providing service to be started. In the same manner as the first embodiment, the service execution unit 216 is controlled by a service control unit 218.

FIG. 7 shows a security module for the repeater. An input/output interface 222 receives the encrypted key K_1' from the information provider or repeater. The encrypted key K_1' is supplied to a key receiver 228 of a key management section 224. The key management section 224 comprises a key controller 226, key memory 230, and key transmitter 232 in addition to the key receiver 228. The encrypted key K_1' is written into the key memory 230 by the key receiver 228. The key k_1' read out from the key memory 230 is supplied to the key transmitter 232. The key transmitter 232 sends out the key k_1' via the I/O interface 222. Thus, the key K_1' is repeated by the repeater's security module.

FIG. 8 shows a security module for the information provider having a repeater function. An input/output interface 242 receives data necessary for synthesize the service package. The security module comprises a key management section 244 which is formed of a key generator 248, key controller 246, key memory 250, and key transmitter 252. The key generator 248 generates an encrypted key K_1' which is an encrypted form of the encryption key K_1 of the service package 10. The key K_1' is stored in the key memory 250. The key k_1' read out from the key memory 250 is supplied to the key transmitter 252. The key transmitter 252 sends out the key k_1' via the I/O interface 242.

The provider's security module further comprises a service package synthesis unit 254, a service package encryption unit 256, and a service package generation controller 258. The I/O interface 202 supplies the input data to the service package synthesis unit 254 in which the service package is synthesized based on the input data. The service package output from the service package synthesis unit 254 is encrypted by the service package encryption unit 256. The encrypted service package is externally output from the I/O interface 242.

FIG. 9 shows a key transmission protocol between two terminals each having a security module. When the user of a terminal "A" wishes to receive the key from a terminal "B", the terminal "A" sends a transmission request to the terminal "B". The terminal "B" communicates with its security module a transmission command and then sends a reception request to the terminal "A". The terminal "A" communicates with its security module a reception command and then sends a reception agreement to the terminal "B". When the terminal "B" sends a start command to its security module, the security modules of the terminal "B" and terminal "A" start verification process and then the session is setup between the security modules of the terminal "B" and terminal "A". The key is transmitted from the security module of the terminal "B" to the security module of the terminal "A". After the session is terminated, the security module of the terminal "B" reports the terminal "B" of the complete of transmission and the security module of the terminal "A" reports the terminal "A" of the complete of reception.

According to the second embodiment, there can be provided a security module in which the service package and the key can be repeated.

According to the first and second embodiments, as in the conventional system described previously, in order to allow the charging function to serve as a platform, a data processing unit, such as a server, on the information provider side creates a service package 10 that contains a pair of content data (name of the content data) the information provider provides and control information (referred to as service description) required to utilize the content data.

An information providing system can be provided which provides service package security protection at the user site, including the transmitting path from the information provider to the user.

(Third Embodiment)

In the third embodiment, the provider encrypts content data and the encrypted content data are transmitted to the user directly or via an agent for repeating data. The service description (hereinafter referred to as a message) associated with the content data is trans-

mitted to the user directly or via the agent. The examples of the service description is the same as that of the first embodiment. The number of agents through which the data is transmitted is not limited to one. The agent is not limited to a person who only repeats the data. Other users can be agents. The third embodiment is characterized in that the content data and/or message are transmitted to an end user via another user or agent. FIG. 10 is a schematic representation of a server on the provider side, FIG. 11 is a schematic representation of a system on the agent side, and FIG. 12 is a schematic representation of a terminal on the user side.

Content data CN, such as video, music, images, etc., which are information to be delivered, are encrypted by an encryption unit 312 and then published as encrypted content data CN' (= T(CN)) to the agent or user. A message M associated with the content data contains multiple (at least two) submessages M_1 and M_2. The encryption unit 312 uses, as an encryption key K, an output of a correlation unit 314 in which the submessages M_1 and M_2 are input to a unidirectional function or unidirectional hash function $f(M_1, M_2)$, a value of a predetermined key generation function to which the output of the correlation unit 314 is input, or a value of the predetermined key generation function to which the output of the correlation unit 314 and data included in the associated message or stored in the device are input, therefore, the content data encryption key K are correlated with the submessages M_1, M_2 associated with the content data.

Unlike the content data CN, the message M is appended with a digital signature of the provider "A" in a signature unit 316 and a signed message M_{sign} (= $S_A(E_B(M))$) is transmitted to the agent or user. Here, $E_B(M)$ is an encrypted message obtained by encrypting the message M using a public key of the agent "B" (or a common key that the provider "A" and the agent "B" share) in order to transmit the message M to the agent B. $S_A(E_B(M))$ represents the encrypted message $E_B(M)$ appended with the digital signature of the provider "A". Hereinafter, E, D, S and V represent operations of encryption, decryption, signature, and verification, respectively. The purpose of transmitting of the message M with the digital signature is to guarantee that the message is a true message transmitted from the provider.

As shown in FIG. 11, in the message agent's system, the message M_{sign} with the digital signature is verified by a signature verification unit 322 and the original message M (= $D_B(V_A(M_{sign}))$ is reproduced. Here, $V_A(M_{sign})$ is the encrypted message for which verification has been made that it was signed by the provider "A", and $D_B(V_A(M_{sign}))$ is a message in plaintext obtained by decrypting the encrypted message $V_A(M_{sign})$ using the secret key of the agent "B" (or the common key that the provider "A" and the agent "B" share).

Suppose here that the agent is malicious and alters part of submessages M_1 and M_2 to yield M_1' and M_2 by using a forgery unit 324. It is assumed that at least one of $M_1' \neq M_2$ and $M_1' \neq M_2$ is satisfied. The agent "B" attaches the own digital signature to the forged message M' by using a signature unit 326. The forged message with the signature M'_{sign} (= $S_B(E_C(M'_{sign}))$) is transmit to the user. Alternatively, the signed message M'_{sign} may be transmitted to the user via still another agent.

As shown in FIG. 11, the agent's system can be implemented by a conventional data processing apparatus, such as a personal computer. For a honest agent, the message is received and merely retransmit to the agent or user. Therefore, the forgery unit 324 is replaced with a mere buffer memory.

As shown in FIG. 12, in the user's system, the message M'_{sign} with the digital signature transmitted from the provider or agent (in this case, the message is forged by the agent) is verified in a signature verification unit 332. The original forged message M' (= $D_C(V_B(M'_{sign}))$ is obtained. Here, $V_B(M'_{sign})$ is the message M' forged and signed by the agent "B", and $D_C(V_B(M'_{sign})$ is a message in plaintext obtained by decrypting the encrypted message M', for which verification has been made that it was signed by the agent "B", using the secret key belonging to the user "C" (or the common key that the agent "B" and the user "C" share). The encrypted content data CN' is decrypted by a decryption unit 336. The decryption unit 336 uses, as a decryption key K', an output of a correlation unit 334 in which the submessages M_1' and M_2' contained in the message M' are input to a unidirectional function or unidirectional hash function $f(M_1', M_2')$, a value of a predetermined key generation function to which the output of the correlation unit 314 is input, or a value of the predetermined key generation function to which the output of the correlation unit 314 and data included in the associated message or stored in the device are input. If the message M is not forged by the agent, then the key K' supplied to the decryption unit 336 will be equal to the key K supplied to the encryption unit 312 on the provider side. In this case, however, the message is forged as described above, the key K' supplied to the decryption unit 336 is $f(M_1', M_2')$.

If the message M is not forged by the agent, then the key K' equals to the key K and the decrypted content data CN" (= R(CN')) matches the content data CN on the provider side.

However, when, as shown in FIG. 11, the agent forges the message, the key K' to the decryption unit 336 is $K' = f(M_1', M_2') \neq f(M_1, M_2)$, resulting in a mismatch between the decrypted content data CN" and the original content data CN. In other words, the original content data are not available on the user side. In such case, the user will suppose that the message given by the agent is not correct, that is, the agent would have committed some injustice, and then notifies the provider of that. As a result, the provider sends the correct mes-

sage directly to the user, thus allowing the user to decrypt and utilize the content data.

In the event that the agent has altered the submessage M1 to M1', in order to cause the decryption key K' (= $f(M_1', M_2')$) to match the encryption key K (= $f(M_1, M_2)$), it is required to change the other submessage M2 as well to M2' that satisfies $f(M_1', M_2') = K$. Since the function f is a unidirectional one, however, it is almost impossible for the malicious agent to find M_2' that satisfies $f(M_1', M_2') = K$, i.e., M_2 that allows K' to be changed to K.

As described so far, according to the third embodiment, by making a content data encryption key a unidirectional function or unidirectional hash function of the whole message or parts of the message (in this embodiment, two parts of the message) associated with the content data, that is, by correlating the encryption key with the message, the decryption of the content data becomes disabled when the message is forged. It therefore becomes possible to prevent unauthorized usage of content data and protect the digital rights of information providers. In addition, since the encryption key is never known to the users and agents, encrypted content data can never be decrypted illegally. Moreover, since the message containing at least two submessages used to generate an encryption key can be transmitted offline to the final user via the agent or other agents, there is no need of installing a key issuing server and hence the cost involved in providing information is reduced. This is suitable for an information providing service by individuals.

(Detailed Example)

Detailed example of a specific application of the third embodiment will be described next. As an example, it is assumed that the content data CN are MPEG-compressed video and the message M contains charging information for billing a user for video. The provider "A" describes in the submessage M1 charging information such that user should pay the provider "A" 1000yen for this video. The submessage M2 may contain the name of video information (content data) or a data pointer indicating the address of the video information, an application pointer indicating which of application programs is to be used for the content data, etc, as shown in FIG. 2. It should be noted that the number of submessages used to generate an encryption key is not limited to two, but may be three or more.

The provider "A" passes the message M to an agent "B" with its signature attached.

The agent verifies the signature of the message M and then sends it to a user directly or via another agent with its signature attached.

The user can decrypt the content data using the key generated from the submessages M1 and M2 of the message M thus received.

Suppose here that a malicious agent "B" alters the submessage M1 to a submessage M1' describing that user should pay the agent "B" 1,000yen and sends it to the user with its signature attached. In such a case, the user will know from this message M1' that it is to the agent "B" that he or she should pay 1,000yen for usage of content data. However, since the message has been forged, the correct encryption key cannot be obtained, so that the user fails to decrypt the encrypted content data. It turns out, on contact with the provider "A", that the agent "B" forged the message. Thus, it is possible to cause a malicious agent's plot to forge a message and take a charge to be paid to the provider "A" to end in failure.

(Fourth Embodiment)

In the third embodiment, it is not required that a digital signature is attached to the message. The fourth embodiment is a modification of the third embodiment in which the message is directly transmitted without a digital signature. FIG. 13 is a schematic representation of a server on the provider side, FIG. 14 is a schematic representation of a system on the agent side, and FIG. 15 is a schematic representation of a terminal on the user side. The provider does not attach the digital signature to the message. The agent does not perform a signature verification and merely relay the received message to the other agent or user. The user correlates the submessages included in the received message.

Additional advantages and modifications will readily occur to those skilled in the art. Therefore, the present invention in its broader aspects is not limited to the specific details, representative devices, and illustrated examples shown and described herein. Accordingly, various modifications may be made without departing from the spirit or scope of the general inventive concept as defined by the appended claims and their equivalents.

For example, in the first and second embodiments, the user cannot know the charging policy data without decryption and a security module is need for decryption. The user has no security module before service utilization. This is inconvenient for users. For this reason, it is desirable that a second charging policy identical to a charging policy to be transmitted in encrypted form be prepared separately and transmitted to the user site without encryption. The user can know the second charging policy data to decide whether to utilize the service or not. In this case, it is the charging policy contained in the decrypted service package that is transmitted to the service execution unit 62. However, a third party might alter the second charging policy in plaintext to render a charge for service free. This affords little user protection. It is therefore desirable that the user device be equipped with means for making a comparison between the decrypted charging policy and the second charging policy and disabling service utilization when the comparison indicates inequality.

In the first and second embodiments, in order to keep the decrypted service package unknown to the user, the decryption unit 60 and the service execution unit 62 are installed in the security module 50. That is, hardware is used to prevent alteration of information. Of course, the service package may be protected by software. A certificate that guarantees that the service package and the key K_1 are not output to outside nor retained may be attached to a service instance itself which is software for implementing the service package. In the absence of this certificate, the service package is disabled from being decrypted. In this case, the decryption unit 60 and the service execution unit 62 need not be installed in the security module 50. Further, if, when hardware is used to prevent alteration of information, the user's terminal is reliable, the decryption unit 60 and the service implementing unit 62 may not necessarily be provided in the security module 50. Although being implemented by a platform, the service package decryption unit and/or charging system may be implemented as an application program as with normal data processing.

It is not necessarily required that an agent or agents intervene between a provider and a user according to the above-mentioned embodiments. With no agent, a message may be transmitted from the provider to the user together with encrypted content data. In this case, the need of signature transfer processing through message is saved.

The transfers of content data and messages may be made online via the internet or offline through DVDs.

In the third embodiment, the signature encryption may be either public key-based or common key-based. The message may include not only charging information but also data described in accordance with a format such as SGML (Standard Generalized Markup Language), HTML (HyperText Markup Language), MHEG(Multimedia and Hypermedia Experts Group), XML(eXtensible Markup Language), and their extended or limited format or the like.

When the correlation unit 341 on the user side which calculates a unidirectional function or unidirectional hash function value is built into the security module, such as a semiconductor chip, a smart card, or the like, which is physically disabled against readout, as well as a decrypting unit for the content data, the security can be further increased because the key K is never read out to outside. The reason is that, if a user intervenes between the module in which the unidirectional function or unidirectional hash function is used to calculate the key K and the module in which the content data are decrypted, the user will be able to know the value for key K to thereby decrypt the content data, associate an entirely different message with the decrypted content data, and encrypt the content data using a different key. It must be avoided to make it possible to decrypt encrypted content data readily with no need of a message.

As described above, according to the present invention, there is provided an information providing system which has a facility of protecting content data and provider's rights at the user site including a transmitting path from an information provider to a user and permits information to be delivered readily.

Claims

1. An information providing system comprising:

   a provider device for providing information to users;
   a user device for utilizing information; and
   an information storage card adapted to be connected to the provider device and the user device and comprising means for storing a second key,
   characterized in that
   the provider device comprises means (42) for sending to the user device, a service package that describes information necessary for utilization of the provided information, the service package being encrypted in accordance with a first encryption system, and means (40) for sending to the user device, a first key used in the first encryption system, the first key being encrypted using the second key which is stored in the information storage card; and
   the user device comprises means (54) for decrypting the encrypted first key within the information storage card.

2. The information providing system according to claim 1, characterized in that the service package after decryption is disabled from being retained within the user device or being output from the user device to outside.

3. The information providing system according to claim 2, characterized in that the encrypted service package is decrypted within the information storage card and the decrypted service package is disabled from being output to outside of the information storage card.

4. The information providing system according to claim 2, characterized in that the user device comprises service package decryption means (60) for decrypting the encrypted service package and means for disabling the service package decryption means from decrypting the encrypted service package when it is not guaranteed that the decrypted service package should not be retained within the user device nor be output to the outside of the user device.

5. The information providing system according to

claim 1, characterized in that the service package comprises information (16) for identifying information to be provided, information (14) for identifying an application program that utilizes the information to be provided, and information (12) indicating a charging policy relating to the utilization of the information to be provided, and the user device comprises an application program execution unit (62) that operates in response to the decrypted application program identifying information, a charging unit that operates in response to the decrypted charging policy identifying information.

6. The information providing system according to claim 5, characterized in that the application program execution unit is implemented by an application program, and the charging unit is implemented by a platform that is different from the application program.

7. The information providing system according to claim 5, characterized in that said provider device comprises means for sending to the user device, a second charging policy identical to the charging policy contained in the encrypted service package without encryption.

8. The information providing system according to claim 1, characterized in that the first key used in the first encryption system is generated in the information storage card.

9. The information providing system according to claim 1, characterized in that the first key used in the first encryption system is generated by an authorized agent and is written into the information storage card.

10. The information providing system according to claim 1, characterized in that the provider device comprises means for sending a ticket to the user device, the ticket associating information identifying the service package with information identifying a key used to encrypt that service package and the user device comprises means for identifying a key associated with the service package to be utilized on the basis of the ticket.

11. The information providing system according to claim 1, which further comprises a repeater unit for receiving the message data from the provider device and transmitting the received message data to the user device.

12. An information providing system comprising:

a provider device for providing information to users;

a user device for utilizing information; and

a security module adapted to be connected to the provider device and the user device and comprising means for storing a second key in such a way that it cannot be read out to outside, characterized in that

the provider device comprises means (42) for sending to the user device, a service package that describes information necessary for utilization of information, the service package being encrypted in accordance with a first encryption system, a first key used in the first encryption system being encrypted using the second key stored in the security module; and

the user device comprises means (54) for decrypting the encrypted first key within the security module.

13. An information providing device for providing information to users with an information storage card for storing a second key, the device comprising:

means (42) for transmitting, a service package that describes information necessary for utilization of the provided information, the service package being encrypted in accordance with a first encryption system; and

means (40) for transmitting a first key used in the first encryption system, the first key being encrypted using the second key which is stored in the information storage card.

14. An information providing system for providing content data and message data in association with the content data, characterized by comprises:

means (314) for calculating an value of a unidirectional function or unidirectional hash function to which the whole message data or parts of the message data are input; and

means for encrypting the content data to be provided using the value of the unidirectional function or unidirectional hash function as a key.

15. The information providing system according to claim 14, characterized by further comprising means for attaching the message data with a digital signature.

16. The information providing system according to claim 14, characterized in that the message data contains charging information concerning a charge for usage of the content data.

17. The information providing system according to claim 14, characterized in that the message data contains data described in a format including

SGML, HTML, MHEG, or XML, and their extended or limited format.

18. An information utilization device for use with an information providing system in which content data and its associated message data are provided and the content data is encrypted, the device comprising:

means for calculating a first value of a unidirectional function or unidirectional hash function to which the whole message data or parts of the message data are input, a second value of a key generation function to which the first value is input, or a third value of a key generation function to which the first value and data included in the associated message or stored in the device are input; and
means for decrypting the encrypted content data using the value of the unidirectional function or unidirectional hash function as a key.

19. The information utilization device according to claim 18, characterized in that the message data is attached with a digital signature.

20. The information utilization device according to claim 18, characterized in that the message data contains charging information concerning a charge for usage of the content data.

21. The information utilization device according to claim 18, characterized in that the message data contains data described in a format including SGML, HTML, MHEG, or XML, and their extended or limited format.

22. An information providing system characterized by comprising:

an information providing device which, in encrypting content data using an encryption key, uses a first value of a unidirectional function or unidirectional hash function to which the whole message data or parts of the message data are input, a second value of a key generation function to which the first value is input, or a third value of a key generation function to which the first value and data included in the associated message or stored in the device are input, as the encryption key and transmitting the encrypted content data;
a repeater unit for receiving the message data from the information providing device and transmitting the received message data; and
an information utilization device which, in decrypting the encrypted content data transmitted from the information providing device using an decryption key, uses a value of a unidirectional function or unidirectional hash function to which the whole message data or parts of the message data associated with the content data and transmitted from the information providing device or the repeater unit as the decryption key.

23. The information providing system according to claim 22, characterized in that the information providing device encrypts the message data, and the repeater unit decrypts the received encrypted message data, encrypts the message data again and transmits the encrypted message data.

24. The information providing system according to claim 22, characterized in that the information providing device sends the message data with a provider's signature attached, and the repeater unit verifies the signature on the received message data and transmits the message data with a message data receiver's signature attached.

25. The information providing system according to claim 22, characterized in that the repeater unit is in the form of the information utilization device.

26. An encryption device for encrypting content data and its associated message data to be separately transmitted, characterized by comprising:

means for calculating an value of a unidirectional function or unidirectional hash function to which the whole message data or parts of the message data are input; and
means for encrypting the content data using the value of the unidirectional function or unidirectional hash function as a key.

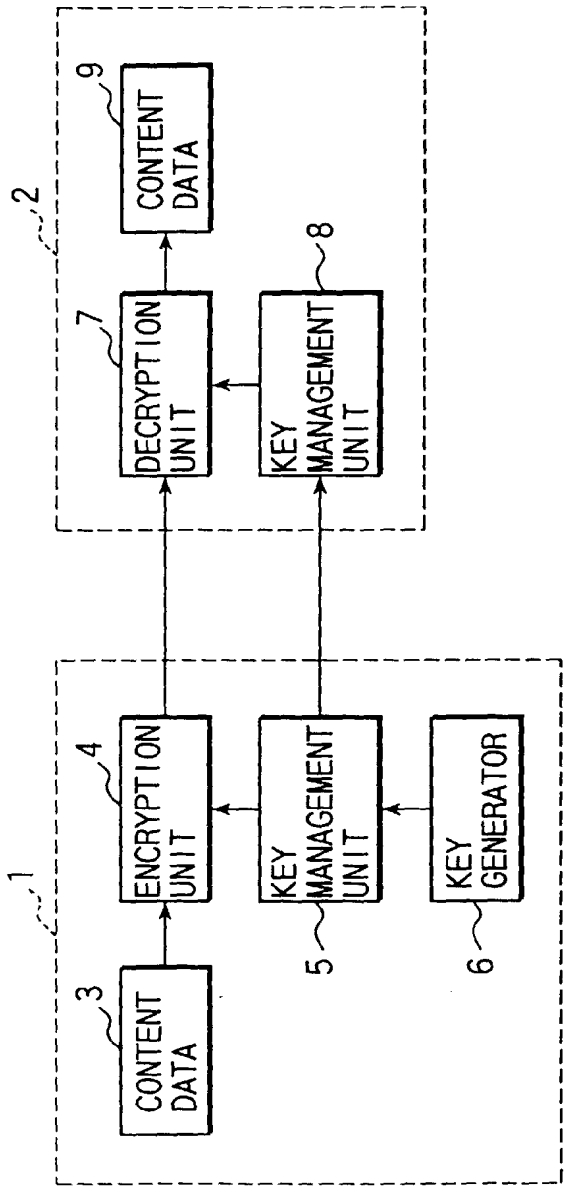


FIG. 1

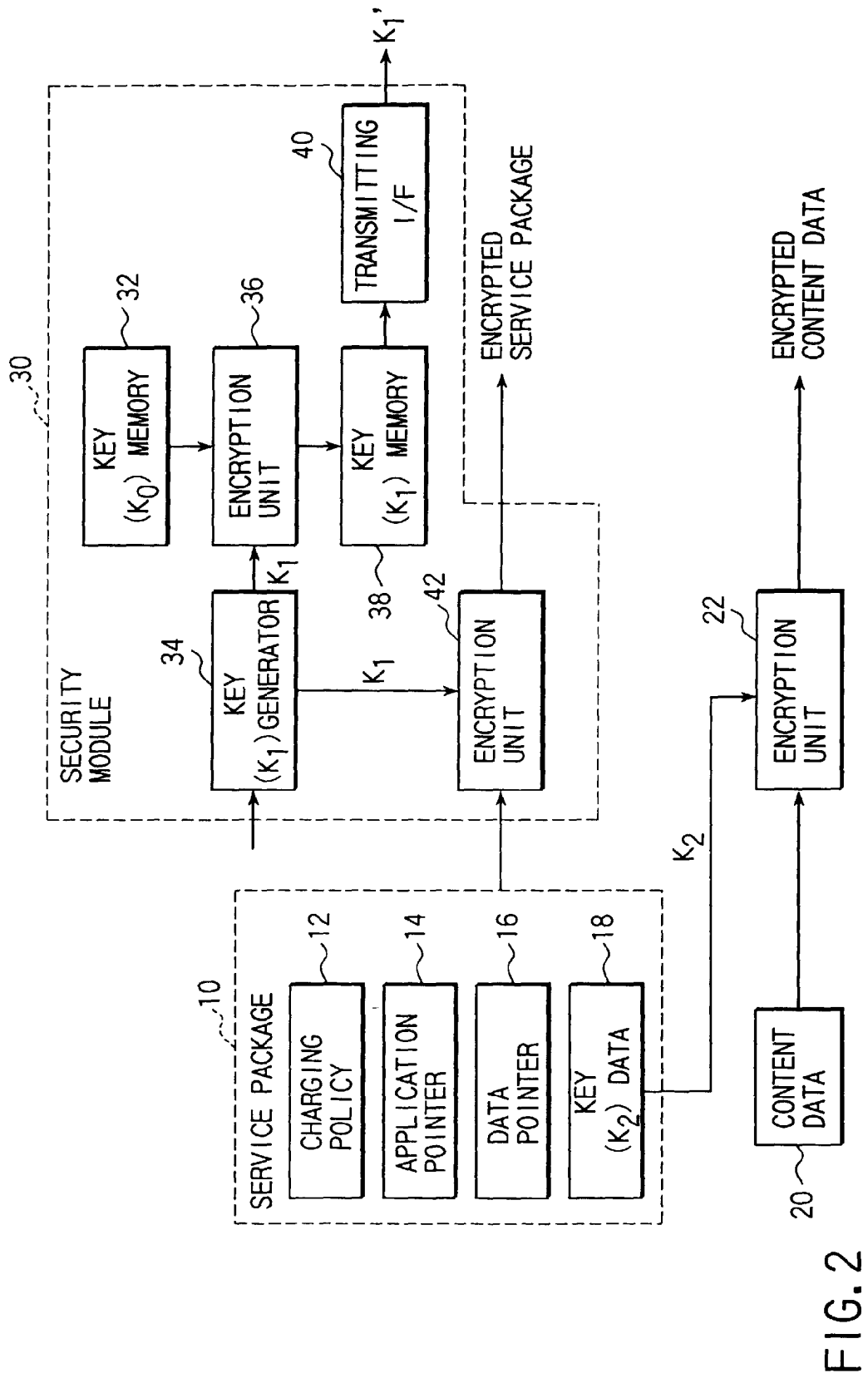


FIG. 2

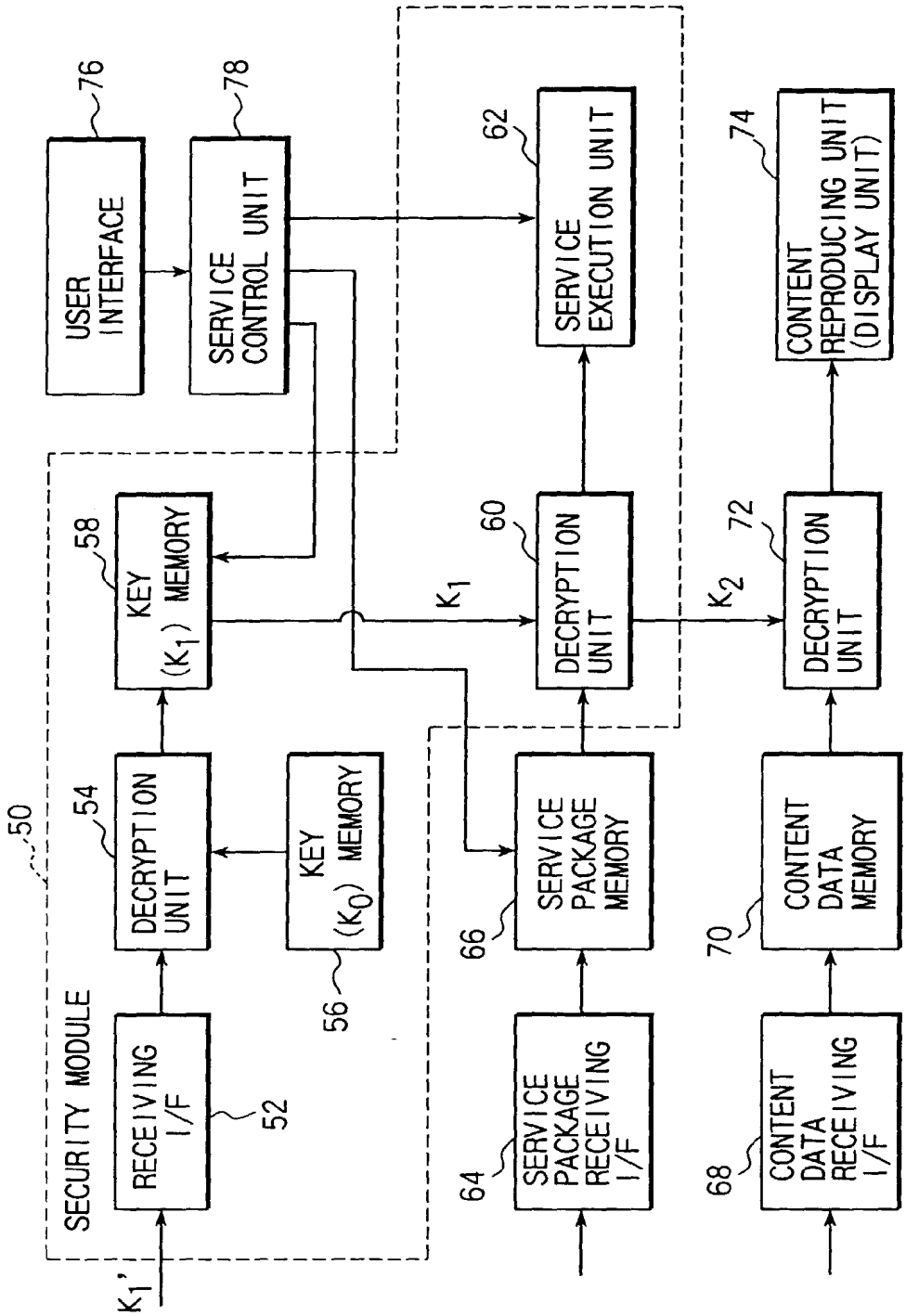


FIG. 3

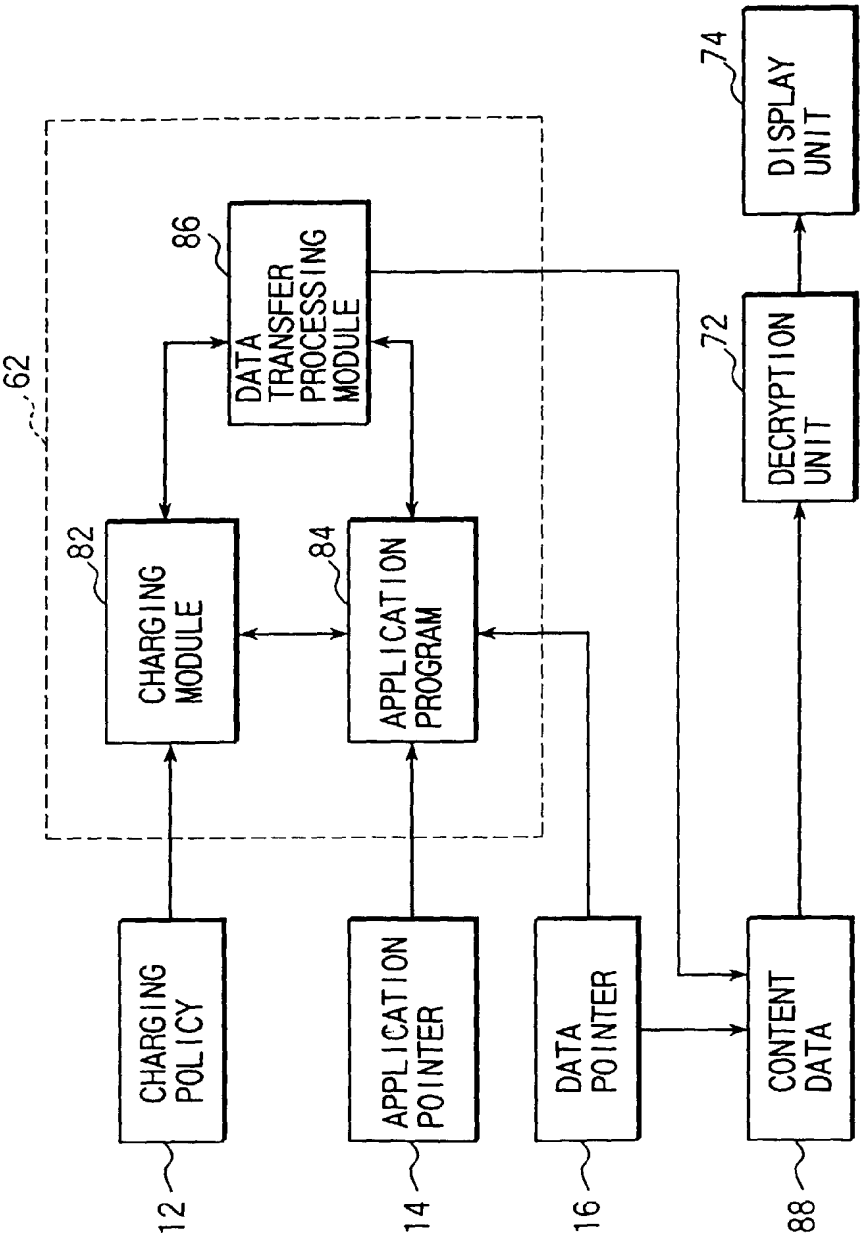


FIG. 4

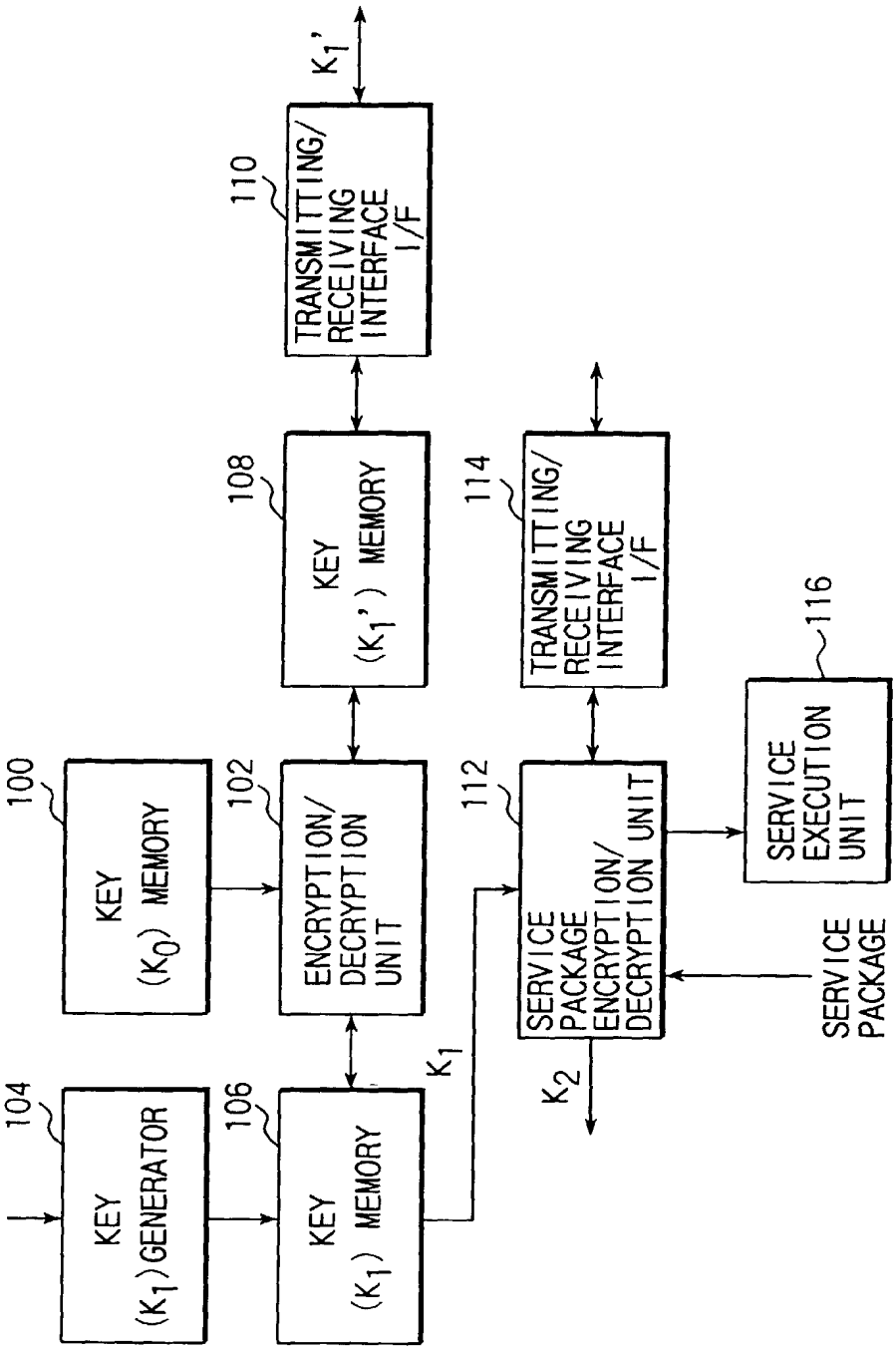


FIG. 5

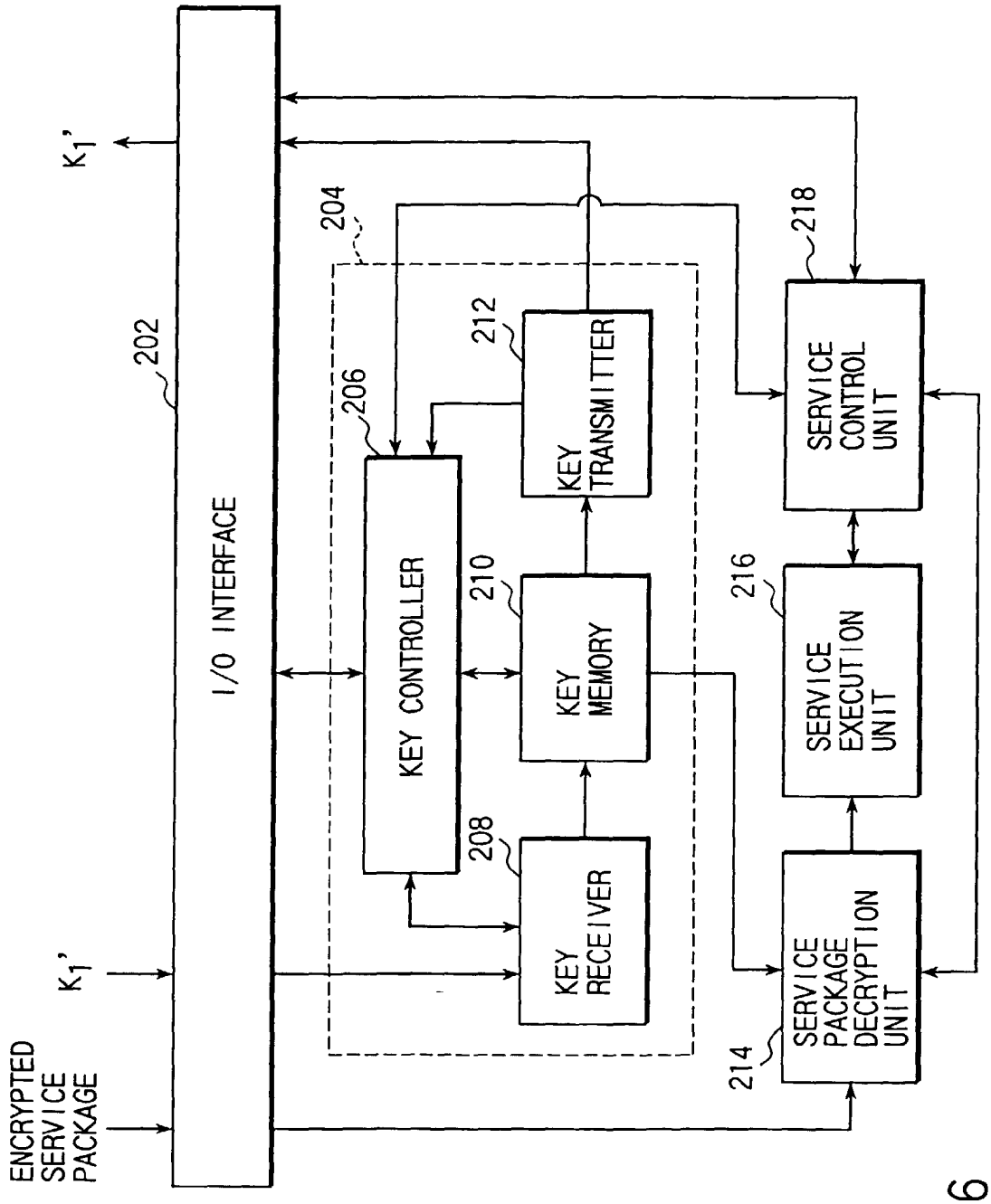


FIG. 6

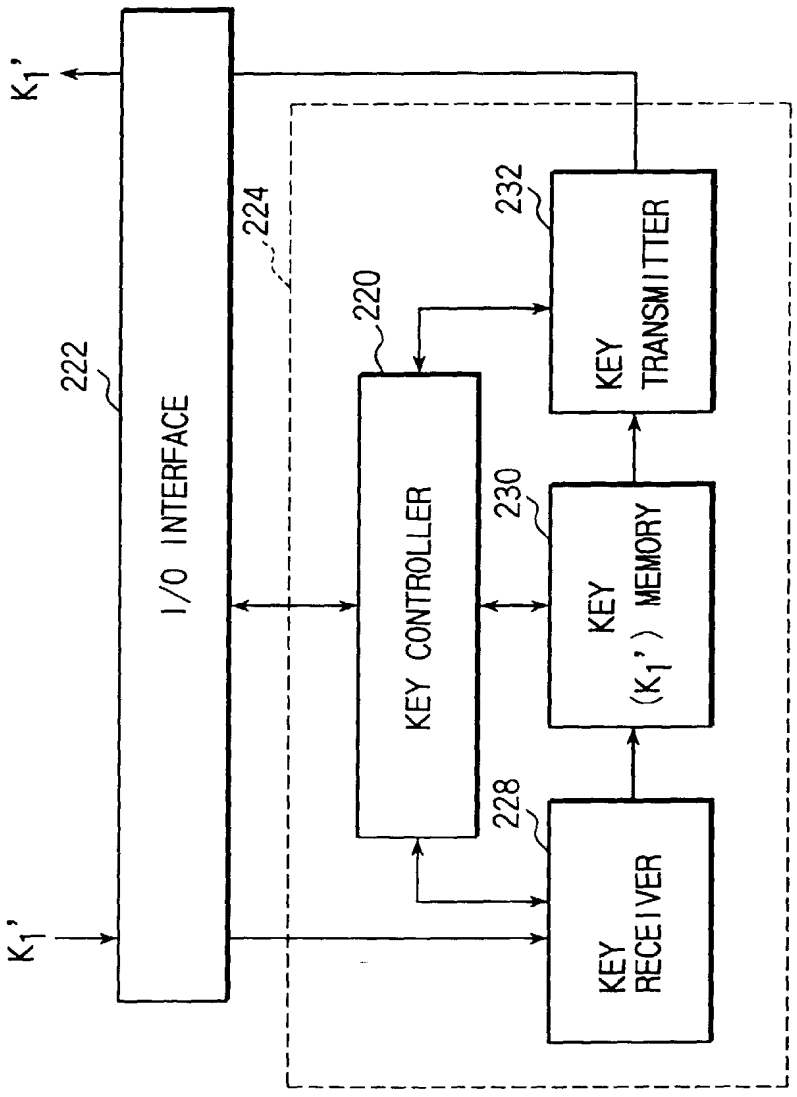


FIG. 7

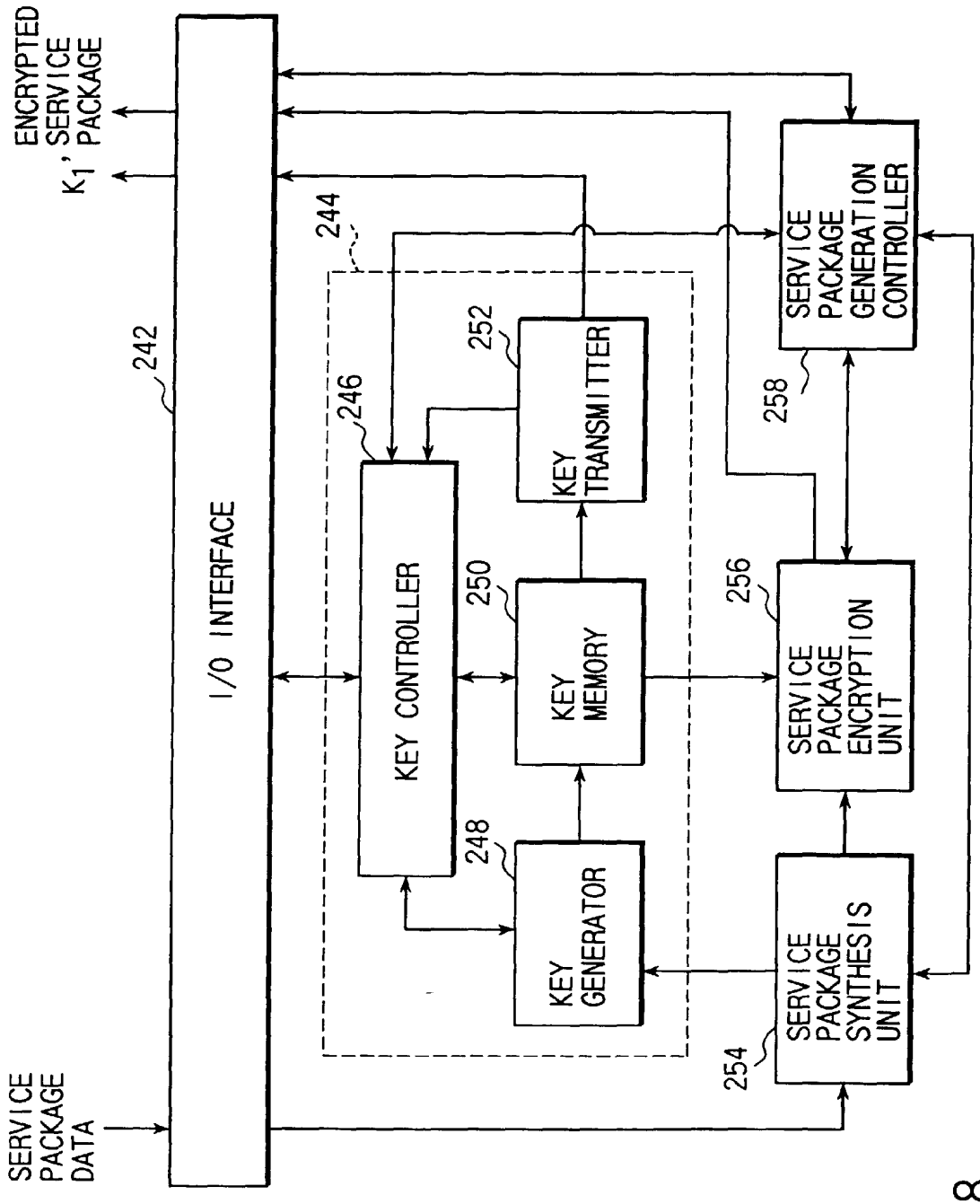


FIG. 8

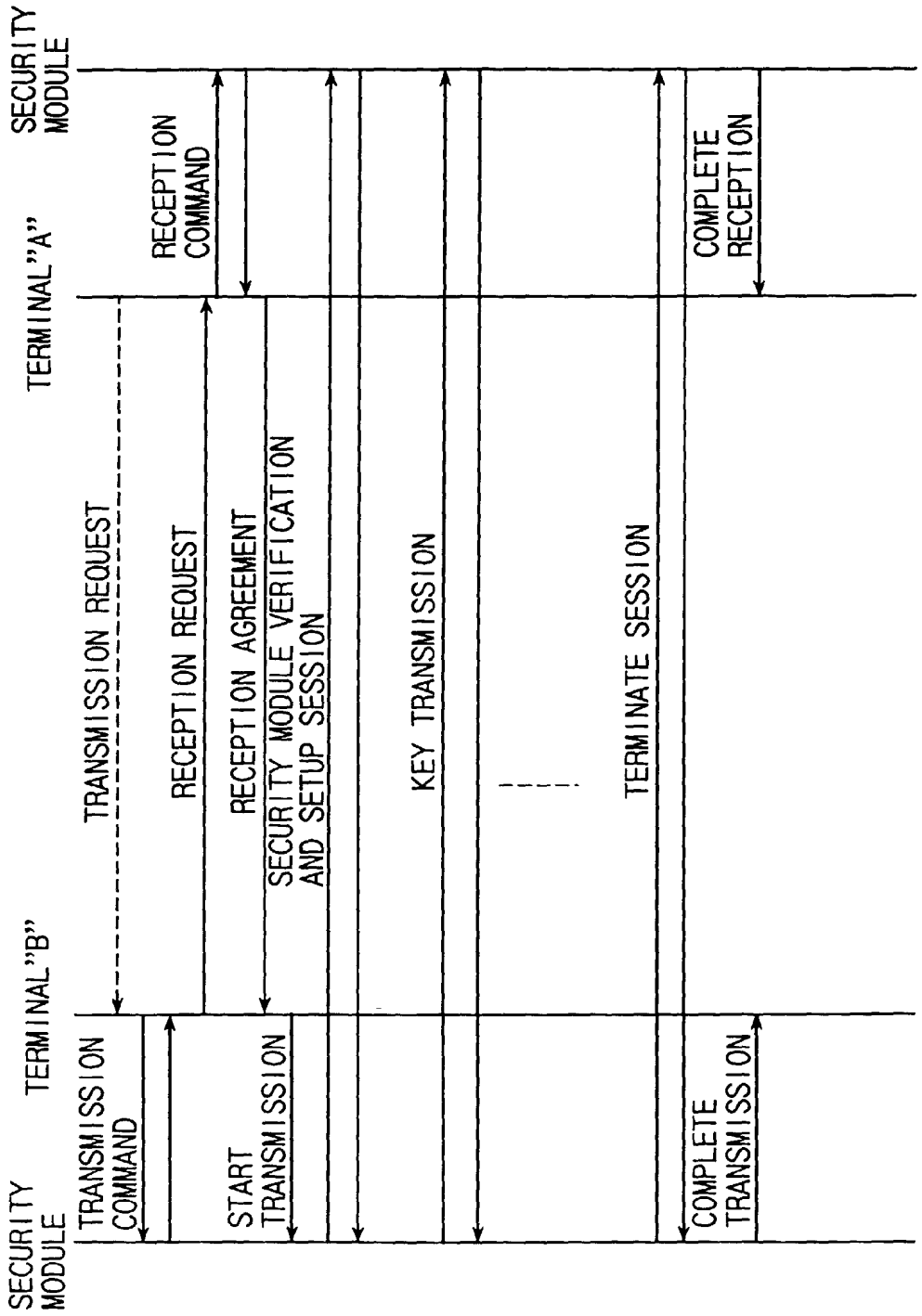


FIG. 9

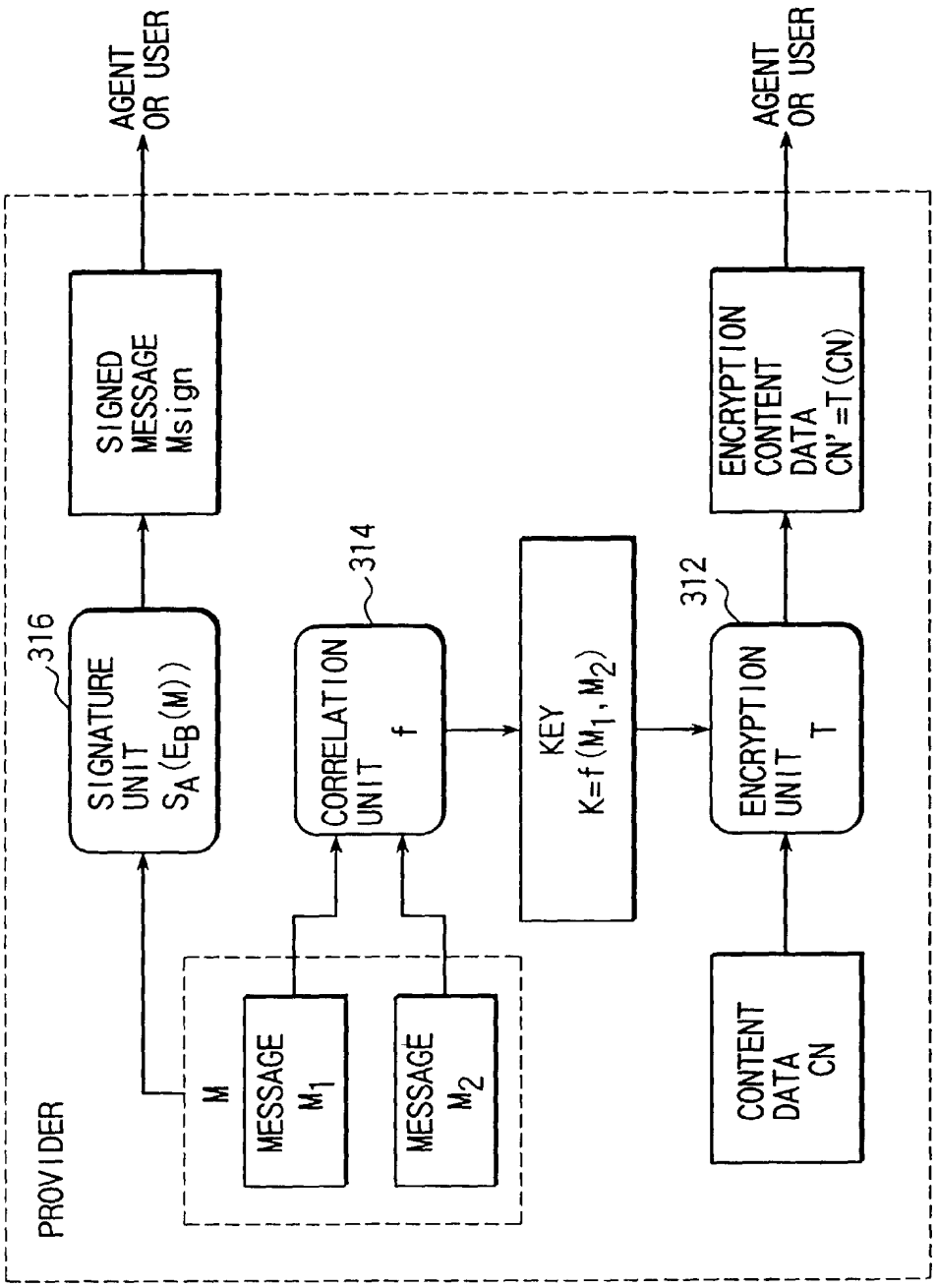

AGENT OR USER
AGENT OR USER
SIGNED MESSAGE Msign
ENCRYPTION CONTENT DATA CN' =T(CN)
316
SIGNATURE UNIT $S_A(E_B(M))$
314
CORRELATION UNIT f
KEY $K=f(M_1, M_2)$
312
ENCRYPTION UNIT T
PROVIDER
M
MESSAGE M_1
MESSAGE M_2
CONTENT DATA CN

FIG. 10

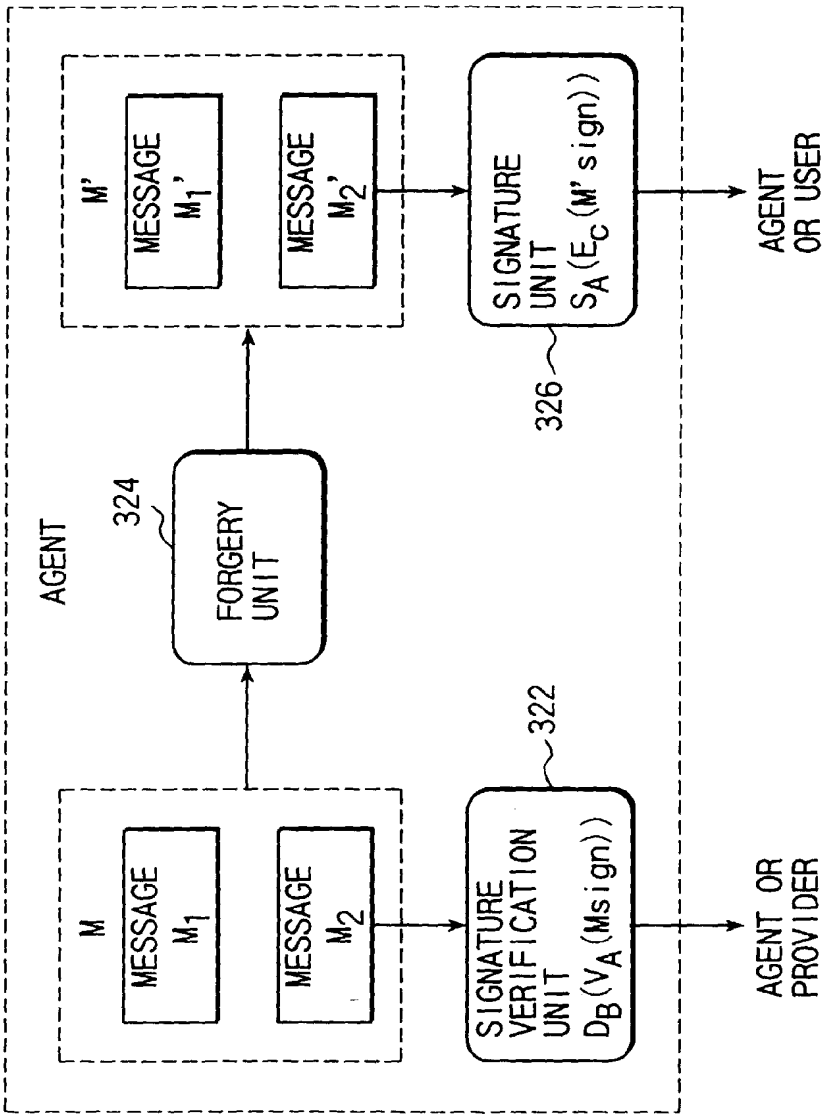


FIG. 11

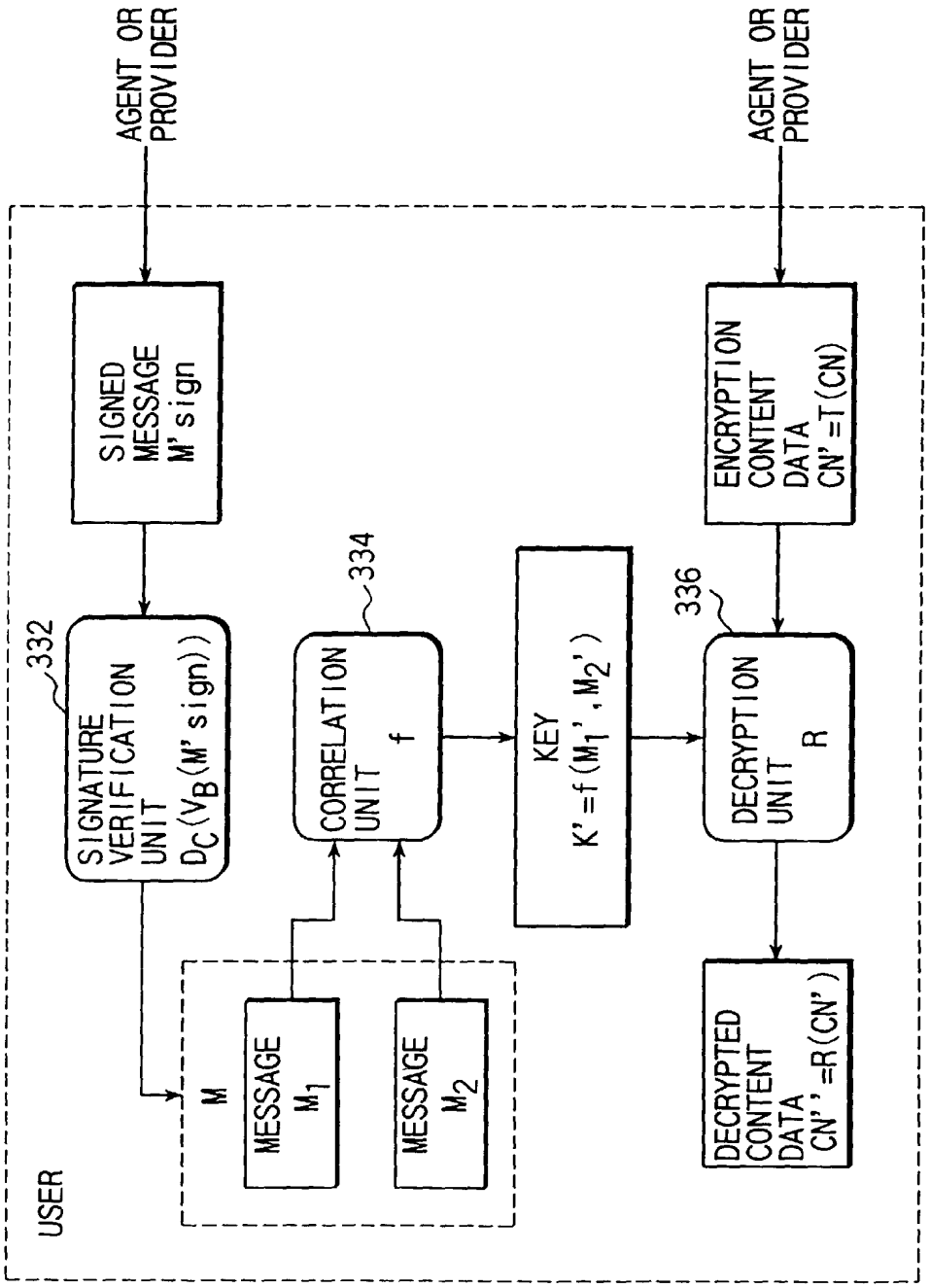


FIG. 12

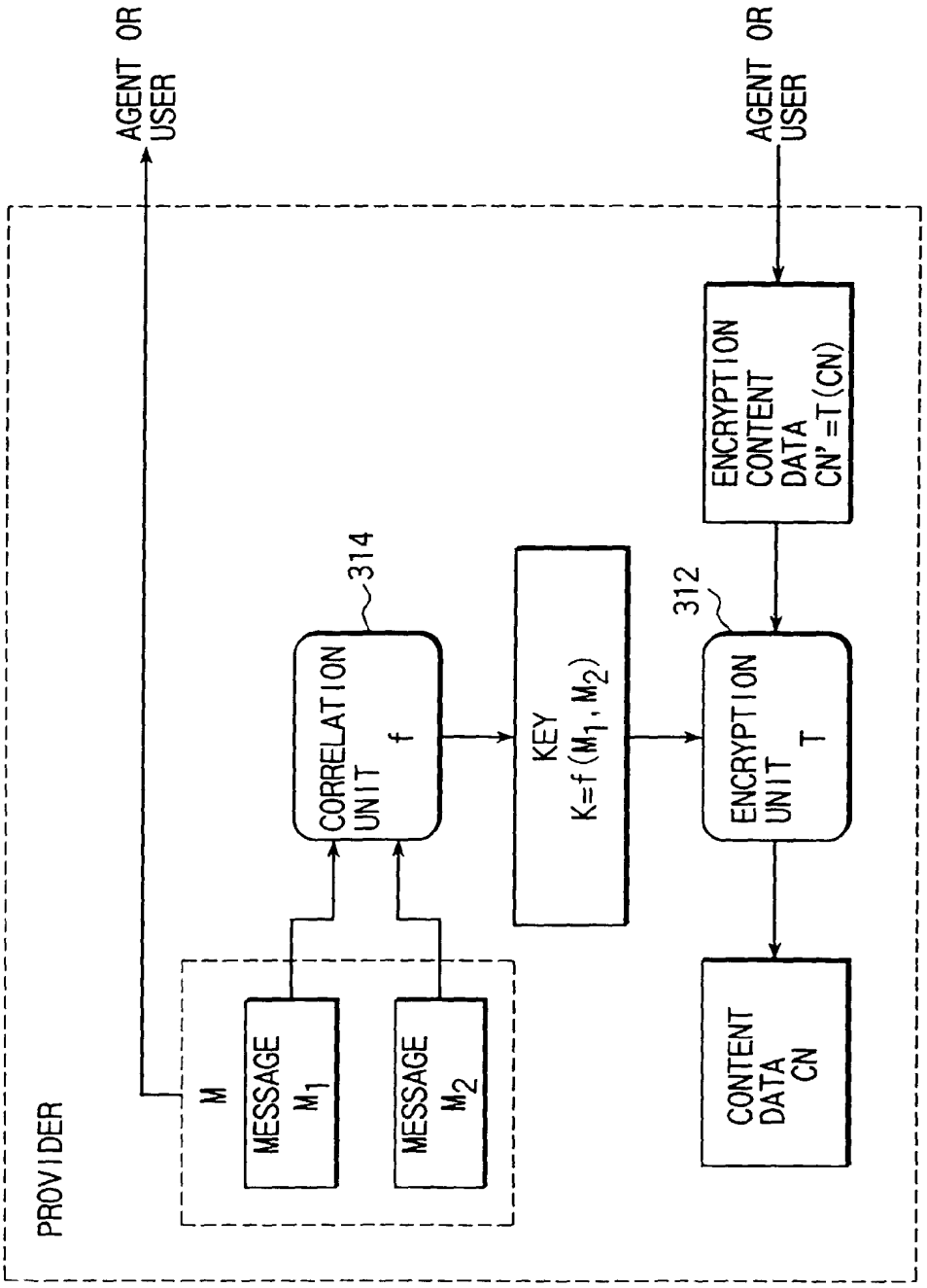


FIG. 13

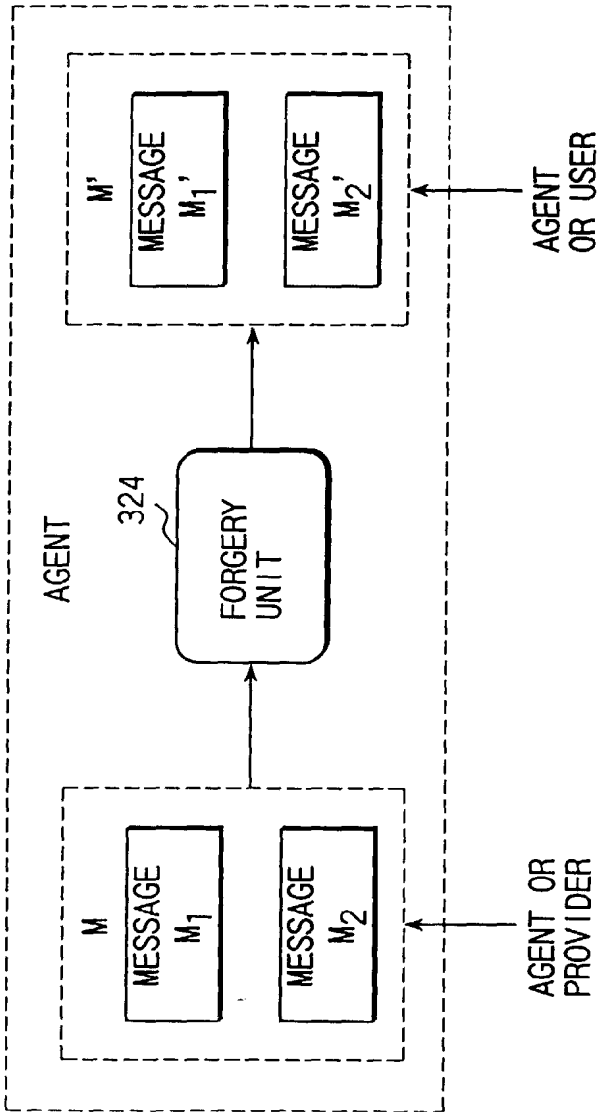


FIG. 14

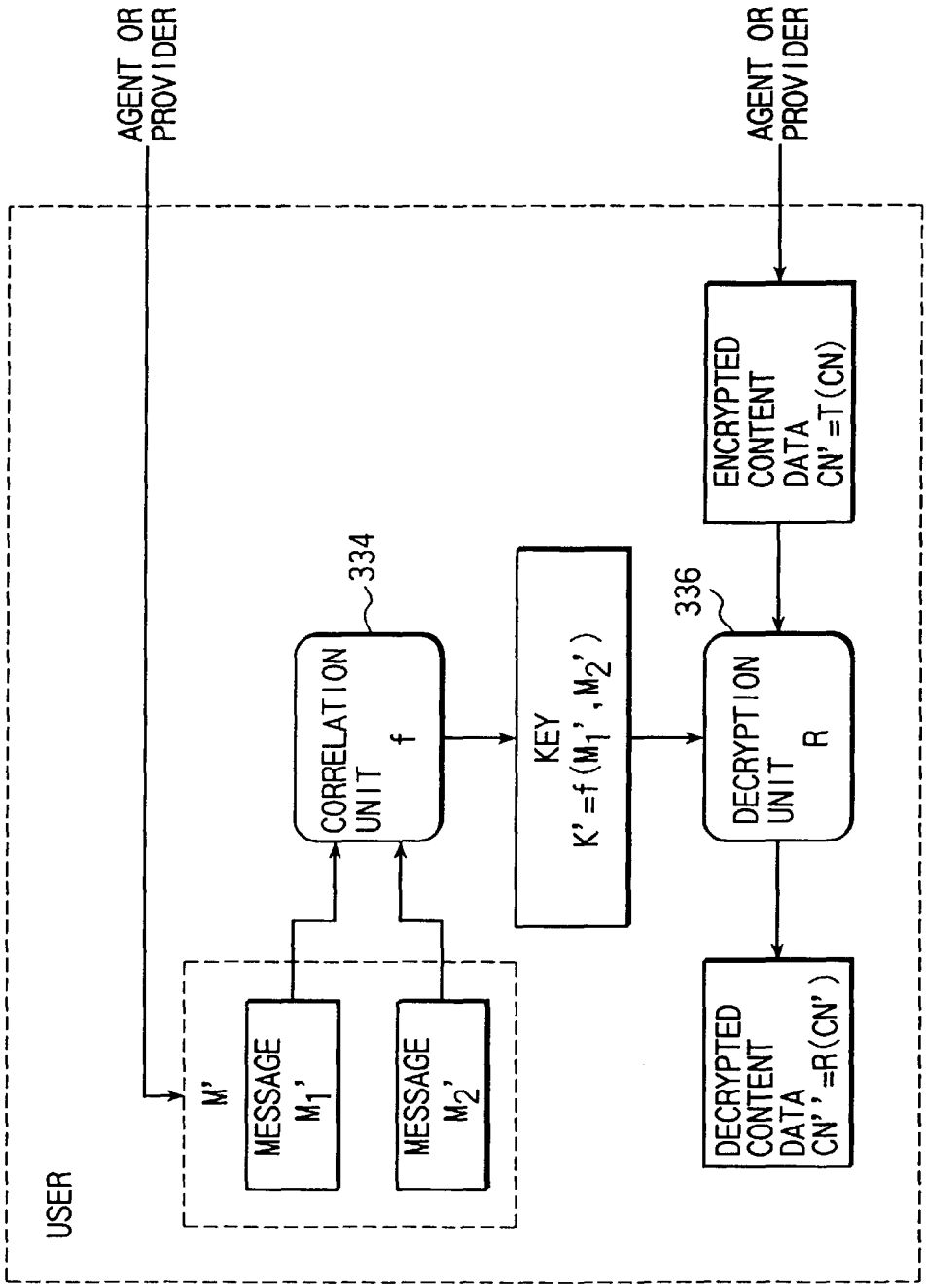


FIG. 15

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 895 171 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.02.1999 Bulletin 1999/05

(51) Int. Cl.[6]: **G06F 17/60**

(21) Application number: **98114099.9**

(22) Date of filing: **28.07.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **28.07.1997 US 901573**

(71) Applicant: **Neoforma, Inc.**
Mountain View, CA 94040 (US)

(72) Inventor: **Mcvicker, Wayne D.**
Mountain View, CA 94941 (US)

(74) Representative:
Schoppe, Fritz, Dipl.-Ing.
Schoppe & Zimmermann
Patentanwälte
Postfach 71 08 67
81458 München (DE)

(54) System for planning projects

(57) A computer aided planning tool provides a platform having a process-based structure including various planning templates for a project linked to information stored in a relational database related to the task performed utilizing that particular template. The planning tool enables users to navigate from information available in various information galleries to process tools utilized to design the components of the project. The database comprises product catalog information and a library of additional information for targeted access. The information stored in the database is unmodifiable or customizable by designation of the entity which submits that information by including the log-in name and unique password of that entity when the information is submitted. A particular page at a web site may be accessed based on information selected during utilization of a particular planning template to supplement database information, rather than requiring users to browse the web site for the relevant page. Additionally, automated two-way e-mail over the Internet is provided to facilitate the acquisition of additional information which is then accessible by users and can be automatically imported for use in the planning templates, including use in applications programs, such a spreadsheet for calculating costs for the project.

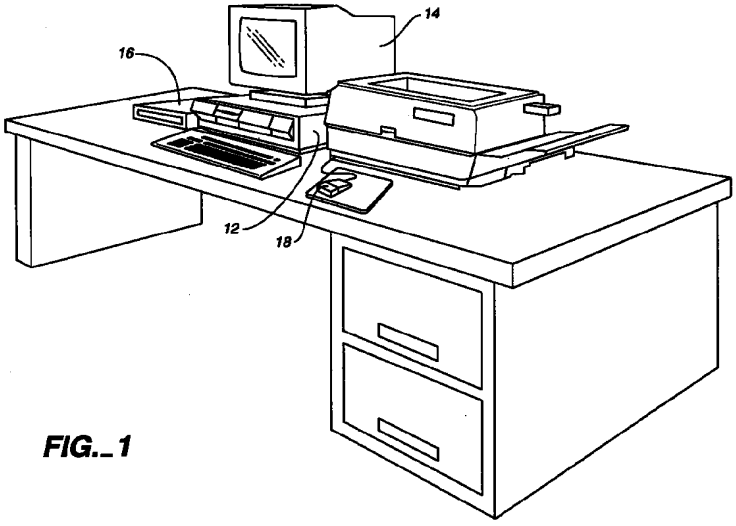


FIG._1

EP 0 895 171 A2

Description

Field of the Invention

[0001] The present invention relates to productivity tools and, more particularly, to computer aided design (CAD) productivity tools. Specifically, one embodiment of the invention provides a computer aided design tool for persons who plan and implement projects.

Background of the Invention

[0002] Persons who design projects are required to know or learn a great deal of information about a project for which they are assigned the responsibility to design. After that information is possessed, various productivity tools are known for designing projects. These productivity tools include various CAD tools which are commercially available for increasing design productivity. These productivity tools are generally provided for professional designers, such as architects and engineers.

[0003] For example, some projects require an in-depth current knowledge and expertise about the project, such as the design of a radiotherapy department for a healthcare facility, which necessitates knowledge of applicable regulations regarding acceptable dosage levels to radiotherapy technicians, materials and equipment utilized in the construction of a radiotherapy department, available commercial products, costs, etc. The design can require a number of complicated calculations, such as the thickness of the shielding that is needed to meet the regulations for radiation dosage levels received by technicians, and the thickness of the shielding depends upon the shielding material which in turn can depend upon the commercial availability of the material and trade-offs between the effectiveness of the material and the cost. This complicates the design process. Therefore, a need has existed for productivity tools for facilitating the design process and reducing the time to design a project, such as the radiotherapy department at a healthcare facility, as well as to enable a person to implement a project design through the purchase of products.

[0004] A useful productivity tool for designing a radiotherapy department is known. This productivity tool is commercially available as the Radiotherapy Department Toolkit from Neoforma, Inc. located in Mountain View, California. This toolkit includes a software-based planning tool and product catalog for radiotherapy departments resident on a database to enable facility planning to be efficient and fast by streamlining the complicated tasks of designing a radiotherapy department and purchasing needed equipment. Advantageously, in addition to providing a tool for configuring a radiotherapy department, the Radiotherapy Department Toolkit enables persons designing a radiotherapy department to access product information for selecting equipment for the department from the product catalog database. Additionally, this toolkit incorporates a web browser to enable persons to visit the web sites of various manufacturers whose equipment is included in the product catalog database. The Radiotherapy Department Toolkit enables persons to select shielding material manufacturers, to review current regulations, to generate standardized physics reports, and to simplify the design of complex shield barriers by automating tedious calculations. Therefore, this toolkit has proved to be a beneficial productivity tool for equipment planners, architects, physicists, physicians, administrators, and regulators.

[0005] Although the Radiotherapy Department Toolkit has various advantages, there are various limitations. Like CAD tools in the architectural and engineering fields that are limited to one aspect of a multi-faceted project, the Radiotherapy Department Toolkit is a specialized tool for the design of a single department. For example, a radiotherapy department may be only one of various departments at a healthcare facility, such as a hospital that additionally has emergency and operating rooms, a maternity ward, an intensive care unit, patient care and physical therapy departments, etc. Furthermore, the product catalog stored in the database of the Radiotherapy Department Toolkit is specialized, as are the other features, including calculation of shielding needs for radiotherapy equipment. Furthermore, the web browser of the Radiotherapy Department Toolkit simply enables persons to connect to the web sites of equipment manufacturers. Thereafter, persons must browse until the product information being sought is located at the web site.

[0006] Therefore, a project planting tool platform is needed that provides an expandable infrastructure that can be adapted as needed to far-ranging project planning applications. Such an adaptable productivity tool for planning a project must be flexible enough to accommodate expansion, such as the addition of planning modules to design various needed aspects of an overall project, and yet be efficiently configured to enhance productivity. Known productivity tools for planning the design of a project do not provide the infrastructure for substantially open-ended expansion and yet possess a level of integration and associational linkages, such as linkages and cross-linkages to and within a database and/or applications programs, such as spreadsheets, to serve as a platform for an efficient planning tool. Therefore, known productivity tools are inherently limited and constitute no more than special-purpose application programs that are not structured to be configured for other applications. At the same time, however, the project planning tool must also enable a person to implement the design for a project at the equipment level.

Summary of the Invention

[0007] One embodiment of the present invention pro-

vides an infrastructure for an efficient project planning tool as a platform for a planning tool for planning and implementing a project. The platform provided by the project planning tool can be tailored to any project planning application, from designing, outfitting, and determining the cost of a building project to planning an entire healthcare facility instead of a single department.

[0008] The computer aided planning tool in accordance with the invention provides a platform having a process-based structure. The process-based structure comprises various planning templates linked to information stored in a relational database related to the task performed utilizing that particular template in connection with the project being planned. The computer aided planning tool enables users to navigate from information available in various information galleries to process tools utilized to design the components of the project. The database comprises product catalog information and a library of additional information for targeted access. The information stored in the database is unmodifiable or customizable by designation of the entity which submits that information by including the log-in name and unique password of that entity when the information is submitted. For example, product vendors can designate their information unmodifiable and distribute their own files of information to users directly. Database information can be supplemented by enabling users to connect to web sites over the Internet, and in accordance with the invention a particular page at a web site may be accessed based on information selected during utilization of a particular planning template, rather than requiring users to browse the web site for the relevant page. Additionally, automated two-way e-mail over the Internet is provided to facilitate the acquisition of supplemental information which is then accessible by users and can be automatically imported for use in the planning templates, including use in applications programs, such as spreadsheets for calculating costs for the project.

[0009] As a result, the computer aided planning tool in accordance with the invention enables users to readily select processes applicable to planning projects for which they have been assigned responsibility and to efficiently navigate within planning templates and access product and library information that enables projects to be implemented. This significantly enhances user productivity.

Brief Description of the Drawings

[0010] The above and other objectives and features and the concomitant advantages of the present invention will be better understood and appreciated by those skilled in the art in view of the description of the preferred embodiments given below in conjunction with the accompanying drawings. In the drawings:

Fig. 1 illustrates one embodiment of the computer aided planning tool in accordance with the invention; and

Figs. 2-21 illustrate various screens that are displayed by the computer aided planning tool shown in Fig. 1 as a project is being planned.

Detailed Description of the Preferred Embodiments

[0011] The computer aided planning tool in accordance with the various embodiments of the invention is executed on a computer 12, as shown in Fig. 1. The computer aided planning tool in accordance with one embodiment of the invention is preferably a 32-bit CAD application compatible with a Microsoft Windows 95 or Microsoft NT 3.51 or later operating system available from Microsoft, Inc. located in Redmond, Washington. The computer 12 comprises a minimum of 16 MB of random access memory (RAM) and preferably includes 24 MB of RAM. The computer 12 also comprises a hard disk drive having 40 MB of free storage space available. The computer is also provided with an Internet connection, such as a modern, for connection to web sites of other entities and exchange of e-mail.

[0012] In an alternative embodiment, the computer aided planning tool can be available by connecting to the web site of the supplier of the computer aided planning tool. The computer aided planning tool can be ported to the web and executed on a web server. Therefore, the requirements for the computer 12 would be reduced.

[0013] Means for displaying information preferably in the form of a monitor 14 connected to computer 12 is also provided. The monitor 14 can be a 640 x 480, 8-bit (256 colors) VGA monitor and is preferably an 800 x 600, 24-bit (16 million colors) SVGA monitor. The computer 12 is also preferably connected to a CD-ROM drive 16. As shown in Fig. 1, a mouse 18 is provided for mouse-driven navigation between design processes comprising the computer aided planning tool, such as navigation from room to room within one of a plurality of department planning templates, for example, a radiotherapy department for a healthcare facility. The mouse 18 also enables persons utilizing the computer aided planning tool (referred to hereafter as "users") to review and select products and services (collectively referred to hereafter as "products") available from various vendors from a plurality of different categories of products for implementing a project, such as radiotherapy equipment for the radiotherapy department of a healthcare facility that is being planned.

[0014] The computer aided planning tool preferably provides three-dimensional visualization of images on monitor 14, such as an entire radiotherapy department for a healthcare facility and the individual rooms in the radiotherapy department being planned. The computer aided planning tool also comprises a relational database having an easy-to-use graphical interface, so that stored product information can be accessed. Three-

dimensional visualizations are also preferably provided for specific implementing products. The three-dimensional visualizations of implementing products are linked to screens displayed by the computer aided planning tool on monitor 14 featuring product specifications, drawings, and manufacturer details. All product screens are interrelated for easy navigation between views so users can learn more about the product, vendor, and how to contact the vendor, which are available to users at the click of mouse 18.

[0015] The computer aided planning tool also comprises functional modules linked to the information displayable in a planning template and other screens. For example, a spreadsheet application executed in the background can process information related to the project design and use product information accessed from the product catalog stored in the relational database to generate items in a table for determining the cost of various implementing components of a design, as well as the overall cost of the project design. For example, a shielding functional feature for planning a healthcare facility can automate the complicated calculation of shielding barriers for a radiotherapy department to facilitate the design process for the radiotherapy department for a healthcare facility.

[0016] The computer aided planning tool in accordance with the various embodiments of the invention comprises a blend of software tools consisting of code executed by a computer, such as computer 12 or a web server connected with computer 12, and a relational database accessible by the computer. The software tools can be encoded on a CD-ROM and downloaded into the computer to execute the computer aided planning routine in accordance with the invention. Data can also be initially provided on a CD-ROM and updated by providing revised data on another CD-ROM or by allowing data to be downloaded over the Internet as the need for updating the data arises. In one exemplary implementation to be described in more detail below, the data stored on the CD-ROM can provide a library or catalog of information related to healthcare for use in planning a healthcare facility, such as products available from various vendors of products to the healthcare industry. Additionally, the computer aided planning tool in accordance with the invention provides access to the Internet so that a web site can be browsed for information related to the project that the computer aided planning tool in accordance with the invention is being utilized to plan. The browser software is integrated into the software planning tool in accordance with the invention in a seamless fashion so that access to web sites of interest can be effected at various times while a project is being planned. In one exemplary implementation to be described in more detail below, web sites of various vendors of products to the healthcare industry are accessible by the computer aided planning tool in accordance with the invention by clicking mouse 18 on an entity name or logo for obtaining additional information relating to planning a healthcare facility, such as supplemental information about products available from various vendors of products to the healthcare industry. The integration of various sources of data for access is a salient feature of the computer aided planning tool in accordance with the invention.

[0017] Furthermore, the integration of a local database of product information with Internet access to the web sites of product vendors provided by the computer aided planning tool in accordance with the invention affords a targeted, reliable, easy, and timesaving approach to selection of products needed for the project being planned. This is based on a two-tiered approach. The first tier is that the database provided by the computer aided planning tool provides a general product catalog, but access to that general product catalog is based on the particular selection from among the set of targeted processes that has been selected and using one of various planning templates comprising the computer aided planning tool. Based on the particular planning template that is selected, knowledge-based access is provided to limit the portions of the database, that is, to limit the portions of the general product catalog, that are accessible for product information. The particular portions of the database that are accessible are limited to the particular targeted planning process that has been selected, which means that unlike the prior art, the entire general product catalog in the local database does not have to be browsed for applicable product information. Moreover, the information stored in the particular portions of the relational database becomes immediately accessible in the context of the process within the planning template, which obviates the need for users to browse for the information.

[0018] The general product catalog can be updated by providing revised data on another CD-ROM or preferably by allowing data to be downloaded over the Internet as the need for updating the data arises. For example, an updated CD-ROM can be created by downloading general product information from product vendors and providing the updated product information to users of the computer aided planning tool in accordance with the invention. Preferably, however, the general product catalog is updated by users directly downloading general product catalog information over the Internet. In any event, the general product catalog stored in the database accessible by the computer aided planning tool in accordance with the invention is preferably maintained by the product vendors. Therefore, the responsibility for providing the data to update the general product catalog resides with the vendors whose products appear in the database.

[0019] In this regard, each vendor is assigned a log-in name and a unique password by the supplier of the computer aided planning tool in accordance with the invention. The unique password enables the product vendor to access the portion of the archival database maintained by the supplier of the computer aided plan-

ning tool in accordance with the invention dedicated to the particular products of that product vendor included in the general product catalog. Access is preferably provided over the Internet. This enables product vendors to update information regarding their products that appear in the general product catalog as the need arises for updating that information, such as discontinuing a product, adding a new product to a product line, etc. One advantage of the computer aided planning tool in accordance with the invention is that the responsibility for updating general product information preferably resides with the set of product vendors, so that the best interests of the product vendors are served by their diligently updating their general product information as business circumstances dictate. This can be easily and quickly accomplished by downloading data over the Internet with the assurance that the integrity of the general product catalog can be maintained by the assignment of unique passwords. This also reduces the burden of database maintenance on the supplier of the computer aided planning tool in accordance with the invention.

[0020] One embodiment of the computer aided planning tool in accordance with the invention provides a set of targeted processes. In one exemplary implementation to be described in more detail below, the target processes relate to planning a healthcare facility, such as selecting the departments that will comprise the healthcare facility, architecting the physical layout of the facility including the various rooms from operating rooms to patient rooms to waiting rooms, selecting actual equipment to deploy in the architected rooms, such as medical equipment, furniture, and even decorations, and other products available from various vendors of products to the healthcare industry. The targeted processes are frilly enabled by the computer aided planning tool in accordance with the invention. A relational database is provided for providing information needed in the planning process. Access to applicable information stored in the database is determined by the particular targeted process that is selected. A spreadsheet is executed in the background for performing calculations needed in the planning process. In one exemplary implementation to be described in more detail below, the spreadsheet application software can compute costs for the healthcare facility being planned, such as cost information depending upon the materials, equipment, accessories, and other costs involved in constructing and furnishing a healthcare facility. A suite of conventional software application tools can be provided to enable planning, such as architectural application software.

[0021] In order to implement the targeted processes provided by the computer aided planning tool in accordance with the invention, a process-based graphic user interface (GUI) is provided. The process-based interface comprises a menu of available selections corresponding to the targeted processes that are provided by the computer aided planning tool in accordance with the invention. Additionally, the GUI of the computer aided planning tool in accordance with the invention is intuitive, and a sufficient number of help screens are provided that the need for hardcopy documentation in the form of a user manual is obviated.

[0022] Preferably, each of the targeted planning processes is connected to a portion of the general product catalog stored in the database related to that planning process, so that the general product catalog provides the backbone of the computer aided planning tool in accordance with the invention. This enables users of the computer aided planning tool in accordance with the invention to migrate easily from a targeted planning process directly to specific product information, which substantially reduces the time needed to fully plan a project and obtain concrete information, such as cost information, for project implementation. Tables generated by users including the results of any calculations performed by the spreadsheet can be utilized to generate an order report utilizing the reporting features of the computer aided planning tool.

[0023] The computer aided planning tool in accordance with the invention provides a specialized design tool integrated with the product database backbone. The computer aided planning tool is also integrated with community provided information in a library database for enabling entities provided with the computer aided planning tool to design and implement a project including complicated technical aspects of the project. In one embodiment of the invention, the library information stored in the database comprises information which can only be modified by the entity which has submitted the information and, alternatively, the information can be customizable. Typically, product information submitted by vendors is submitted with the log-in name and unique password of the vendor so that users cannot modify the information. An entity which submits library information can also include its log-in name and unique password with library information to render that information unmodifiable. Otherwise, information in the database can be imported by users and modified. Enabling product and library information to be designated as unmodifiable can better assure the integrity of the database.

[0024] The computer aided planning tool preferably includes integrated spreadsheet applications software integrated with the computer aided planning tool, which is linked to predetermined selections from the planning templates and executes in the background for calculating the quantity and cost of components needed to implement a project For example, the spreadsheet applications software can be configured to calculate the amount of shielding material based on a selected type of material using conventional equations for determining the amount of a material based on the shielding properties of the particular material and then calculates the associated cost of utilizing that material in a radiol-

ogy department of a healthcare facility. The spreadsheet is integrated with the database and linked by selection of the planning template to product information and library information to import the information needed to perform the needed calculations to determine quantity and cost which are then displayed. Preferably, the computer aided planning tool displays the particular product, together with the quantity and cost, in tabular form. Therefore, complicated aspects of planning a project are processed in the background by the computer aided planning tool, so that complicated procedures connected with a selected planning template and various procedures that require expert knowledge are performed in the background simply through the selection of a template and straightforward selection of available components, such as products available to implement those components. This facilitates planning a project, particularly a complicated project that may require expert knowledge in a particular field. Advantageously, even an expert in the particular field is able to save a great deal of time, since tedious calculations are performed automatically by the computer aided planning tool.

[0025] The library of information stored in the database is also preferably updatable through connection over the Internet to the web site of the entity which published the information. The library of information comprises submittals of information by the universe, or community, of entities which are provided with the computer aided planning tool. Any person who has authorized access to the computer aided planning tool at an entity provided with the computer aided planning tool can author and submit information for access by the entire community of entities which also are provided with the computer aided planning tool and users at those entities. Any such entity or person who desires to author information for inclusion in the library can add the information to the web site of the entity where the information is authored and/or can submit the information to the supplier of the computer aided planning tool. The submitted information is published through the library available to entities provided with the computer aided planning tool. The information published by the supplier of the computer aided planning tool and/or not submitted but simply accessible through browsing the web site of the entity at which the information was authored enables direct access by users of the computer aided planning tool. Preferably, the computer aided planning tool includes linkages to the addresses of the web sites of entities which publish in the subject areas related to the selected planning template as part of the relational database, so that users can easily determine the web site addresses which they can browse if they desire to seek additional information. The interests of all are served by the submission of information for publication in the library.

[0026] For example, one group of entities provided with the computer aided planning tool is, of course, persons who desire to plan a project, such as a healthcare facility. It is important that those persons are informed regarding any applicable regulatory requirements for the project, such as the maximum level of exposure to radiation that is permitted to technicians who work in a radiation therapy or radiology department at a healthcare facility. Moreover, these persons are also well-served to learn from the experience of other entities which have implemented a similar project and to also determine whether or not an industry or trade association or other professional group has published guidelines or standards that facilitate the implementation of a project, such as guidelines by the American Medical Association for equipping a hospital emergency room or intensive care unit. Moreover, some entities may voluntarily publish information about a project that may be invaluable to another entity planning a project at a later time and the prestige that can attach to the publishing entity when other entities re-use part or all of an implementation developed by the publishing entity, such as one healthcare planner implementing an intensive care unit based on the implementation of such a department submitted by Stanford University Hospital.

[0027] Another group of entities provided with the computer aided planning tool and well-served by submission and publication of information is any governmental agencies which oversee regulation of the implemented project, such as a healthcare facility. The fact that there are regulations regarding an implementation that incorporates regulated equipment, such as radiation equipment in a healthcare facility, has been mentioned earlier. Other regulations may also apply to a project being planned, such as building codes that apply to a healthcare facility. Government agencies and other regulatory or oversight agencies can easily and effectively disseminate needed information by submitting the information to the provider of the computer aided planning tool for publication or having that information available at their web site for access by users of the computer aided planning tool.

[0028] An additional group of entities provided with the computer aided planning tool and well-served by submission and publication of information is professional consultants who specialize in the planning and/or implementation of various aspects of the given type of project and who are available to be retained by the entity accessing the library for information, for example, a hospital architectural firm which could be retained to render actual plans for construction of a hospital. The library accessible by the computer aided planning tool provides a vehicle for a professional consultant to publicize expertise and prior experience by having that information available at the web site of the consultant for access by users of the computer aided planning tool.

[0029] As a further and quite obvious example, a group of entities provided with the computer aided planning tool and motivated to submit information for publication is vendors of products for actual implementation

of the various components of the project that is the subject of the planning. The library accessible by the computer aided planning tool provides a mechanism for a product vendor to publicize important product development and product announcements, such as the development of new oncology equipment for the radiotherapy department at a healthcare facility, by having that information available at its web site for access by users of the computer aided planning tool.

[0030] The groups of entities that would publish in the library accessible through the computer aided planning tool in accordance with the invention vary from one type of project being planned to another. Therefore, there is no limit to the community of entities which are provided with the computer aided planning tool and persons at those entities who would author and submit information.

[0031] Considered in more detail, the product database stores product information. The product information comprises graphic and/or text information relating to products. The product graphics and/or text for the various product information stored in the database is preferably provided and updated as needed (i.e., maintained) by the respective vendors of those products. Similarly, the library information comprises graphics and/or text information relating to products or other aspects of a project. The graphics and/or text for the various library information stored in the database is also preferably provided and updated as needed (i.e., maintained) by the entities of the community which submit that information for publication. The computer aided planning tool displays this graphic and/or text when a planning template is active and the product is selected for implementing a component of the project being planned.

[0032] When a particular component is selected within a planning template, one embodiment of the computer aided planning tool in accordance with the invention enables additional information relating to that component, such as a product, to be obtained, that is, the computer aided planning tool provides a mechanism to obtain information to supplement product information stored in the database. The additional information is obtained by selecting a specific product, for example, by clicking mouse 18 on a product listed on a screen displaying one or more products of a vendor. In response to selection of the specific product, the computer aided planning tool connects over the Internet to the vendor and directly accesses the page at the web site of the vendor where the additional information about that particular product appears. This obviates the need for users of the computer aided planning tool to browse the web sites of vendors for additional product information when specific products have been selected.

[0033] In another embodiment, the additional information is obtained by a two-way e-mail exchange between the computer aided planning tool and the vendor of the selected implementing product or entity which authored the library information. To facilitate the exchange of e-mail, the computer aided planning tool preferably provides check boxes within the planning template, which can be selected to request a quote or supplemental information or references relating to a product. If a check box is not selected, the request is preferably defaulted to a request for supplemental information relating to the product. In response to selection of a check box, the computer aided planning tool preferably automatically assembles a pre-formatted e-mail message by the computer aided planning tool to the e-mail address of the vendor or publishing entity stored in the database together with the product or library information. The assembled request message contains: an identification of the product or publication about which supplemental information is requested, the e-mail address of the vendor or product entity to which the query is to be sent as well as the e-mail address of the computer aided planning tool which sends the request, and a pre-formatted text message dependent upon the selected check box, such as "Please reply with pricing information for the product identified in the subject of this message" or "Please reply with all available product information respecting the product identified in the subject of this message." Preferably, the computer aided planning tool enables customized notes to be appended to the pre-formatted text, for example "Please also specify earliest delivery date and method of shipment." If the vendor or publishing entity does not have a current e-mail address, the assembled e-mail request preferably additionally incorporates the facsimile transmission number of the requesting entity, and the e-mail address is preferably defaulted to the e-mail address of the supplier of the computer aided planning tool. Upon receipt of the e-mail message with the default e-mail address by the supplier of the computer aided planning tool, the supplier can convert the e-mail message to a facsimile transmission and send the facsimile transmission to the vendor or publishing entity with a request to reply by facsimile transmission to the telephone number of the computer aided planning tool which sent the request. For example, if additional information is desired about a large number of products needed for implementation of a project being planned, an e-mail message is assembled for each product as to which additional information is sought and sent to each of the product vendors, which may result in a substantial volume of e-mail messages being sent to product vendors by the computer aided planning tool.

[0034] If the vendor or publishing entity to which the request is addressed also possesses e-mail capability, a pre-formatted reply is assembled. The assembled reply message contains: an identification of the product or publication about which supplemental information was requested, the e-mail address of the computer aided planning tool which sent the request as well as the e-mail address of the vendor or publishing entity which sends the reply, and a pre-formatted text message dependent upon the selected check box, such as

"The price of the product identified in the subject of this message is $...." or "Supplemental product information respecting the product identified in the subject of this message is...." Preferably, the reply can also include a separate section for reply to any customized note appended to the pre-formatted text, for example, "The earliest delivery date is ... and method of shipment is by overland freight."

[0035] Two-way e-mail over the Internet provides significant advantages. One advantage is that e-mail affords an effective mechanism for communicating product and published information. Another advantage is that e-mail obviates the need to interrupt the planning process to telephone or draft a letter request for supplemental information. E-mail is particularly convenient when the vendor or publishing entity is located In another time zone or another country. Consequently, international products and published library information from abroad can more logically be included in the database. Therefore, the computer aided planning tool has global geographical application.

[0036] Also, a preferred embodiment of the computer aided planning tool in accordance with the invention automatically processes replies by vendors and publishing entities to e-mail requests. Users route e-mail replies to an e-mail queue when the e-mail application of the computer on which the computer aided planning tool executes is active. When the computer aided planning tool is thereafter executed, the e-mail replies in the e-mail queue are automatically parsed, and the pre-formatted replies are disassembled and the supplemental information is de-embedded and entered in the tables of the planning templates that existed at the time when the check boxes of those templates were checked to generate the corresponding requests. The format of the de-embedded information is compatible with the other information contained in the tables that have been generated within the planning template. This provides a very powerful and automated mechanism for inclusion of supplemental information for planning a project and implementing components of that project. This supplemental information is also processed by any spreadsheet application program executing in the background to perform calculations, such as the total cost of implementing a project.

[0037] One implementation of the computer aided planning tool in accordance with the invention will now be described. It is to be understood that the implementation to be described is by way of example and not by way of limitation. The features of the computer aided planning tool in accordance with the invention apply generally to planning a project and are not restricted in any manner to the specific implementation to be described below. Other exemplary potential implementations will be mentioned following a detailed description of the one exemplary implementation.

[0038] One exemplary implementation provides a computer aided planning tool for healthcare. In one implementation, for example, the computer aided planning tool can be a healthcare facility planning tool.

[0039] The computer aided planning tool comprises a relational database comprising product information from vendors. The database also comprises a library comprising worldwide regulatory agency regulations and other information, such as files generated by record-and-verify systems like VARiS, which can be accessed from the library stored in the database. Also, the library can store information relating to architectural solutions that present a sampling of architectural drawings and graphics of architectural design implementations from around the world for evaluation by an architect involved in a planning project for a healthcare facility. The computer aided planning tool in accordance with one implementation of the invention can include a spreadsheet application, such as Lotus 1-2-3 available from Lotus Development Corporation, which is pre-configured to perform various calculations within a selected planning template such as the cost of a project being planned. A spreadsheet can be also be configured to calculate shielding in connection with a process which comprises designing a radiotherapy department. Automated shielding calculations are provided by the known Radiotherapy Toolkit and therefore do not in and of themselves constitute the present invention. Products available from vendors, such as commercially available doors ad wall assemblies, can be selected, and parameters included in the information relating to those implementing components stored in the product database are called by the spreadsheet and used in the calculations performed by the spreadsheet. Regulatory agency defaults can also be used in the spreadsheet calculations.

[0040] The computer aided planning tool, which comprises a program or set of coded instructions which is executed by computer 12 or a web server to which computer 12 is connected, will now be described in detail with reference to Figs. 2-21. Fig. 2 illustrates an opening screen, which can be referred to as the process screen, that is displayed by monitor 14 when the computer aided planning tool is opened or accessed. In the event that the computer aided planning tool is executed by a web server connected to computer 12, the process screen can be denoted as the "Home Page," as indicated at the lower right hand corner of Fig. 2.

[0041] The process screen shown in Fig. 2 is the screen in which the interface to the various processes throughout the computer aided planning tool is defined. Generally, the computer aided planning tool comprises many different screens associated, or linked, in many ways. Each screen is treated as a unique object, which enables screens to be bundled through linkages in different orders to create different modules. The process-based platform provided by the computer aided planning tool provides navigation through a variety of tools and a variety of screens of information accessed from the associated relational database, and the process-

based interface facilitates navigation. The user can select next and previous buttons, move a slider to jump ahead and behind, and select specific icons. All icons are defined on the process screen, and the user can jump to the process screen multiple ways. The pointer can be positioned on an icon to cause the definition of that icon to appear. There are many ways for the user to jump around as well as to be guided through a process, unlike web interfaces which are random interfaces that are not for designed for a particular process, such as found in the known Radiotherapy Department Toolkit. A selected process of the computer aided planning tool guides the user through steps of that particular process as a part of process.

[0042] The process-based interface of the process screen shown in Fig. 2 enables the user to build a process based on a set of menu commands. The process-based interface of the computer aided planning tool enables the user to focus on a selected process to plan and implement a project. In contrast, the known Radiotherapy Department Toolkit lacks the guided, directed, and customizable tour of screens, for example, approximately 45 to 60 screens, of which only a portion, for example, 20 screens, are pertinent to any particular user, that is provided by the computer aided planning tool.

[0043] The processes which are listed on the process screen shown in Fig. 2 are user definable so that the processes can be configured on a profession-specific basis, such as processes for planning an entire healthcare facility. Furthermore, a hospital administrator is likely to utilize different processes that an architect or a physicist. Consequently, the process-based interface can be redesigned. The resulting process screen is also a file which can be rendered unmodifiable, display-only information. This provides a flexible process-based platform for the computer aided planning tool.

[0044] The process screen shown in Fig. 2 enables a user to position the mouse pointer to highlight one of the identified processes under "Select a process..." on the right hand side of the screen and view a description of that process below. Moreover, the features of the highlighted process appear on the left hand side of the screen. In the exemplary implementation of the computer aided planning tool for planning a healthcare facility, the process screen can include the processes "Edit a department or room template," Explore Neoforma Healthcare," "Linac Shielding - New Construction," "Linac Shielding - Retrofit," "Plan a new room or department," and "Select a product," for example. The user can define other processes or delete one or more of the process selections listed on the process screen shown in Fig. 2 to provide a process-based interface that can be customized at the home page level.

[0045] The processes can vary in complexity. In this regard, "Plan a new room or department" is a comprehensive process tool for planning and implementing departments and individual rooms of a healthcare facility and "Linac Shielding - New Construction" is a specialized process tool for designing and implementing shielding for a radiotherapy room at a healthcare facility. On the other hand, "Select a product" is a streamlined process for identifying products for implementation of a component of a project.

[0046] Clicking mouse 18 on the highlighted process on which the pointer is positioned selects the identified process. As shown in Fig. 2, the highlighted process is "Explore Neoforma Healthcare." This can be designated as the default process and provides access to various screen galleries for the user to browse. Clicking mouse 18 causes "Explore Neoforma Healthcare" to be selected. This process enables the user to simply access and view screens contained in various screen galleries associated with the features on the left hand side of the process screen shown in Fig. 2. This process allows the user to navigate through screen galleries, rather than the screen galleries being tied to one of the other processes listed in the process screen and thus browse the program in overview fashion. The screen galleries comprise the following galleries.

[0047] A department galley enables a user to view the overall scope of one component of a planning project. The screens contained in the department gallery provide the user a bird's-eye view of an entire department, such as a radiotherapy department. For example, Fig. 3 illustrates such a department gallery screen accessed through the "Explore Neoforma Healthcare" process and displayed on monitor 14. The department gallery screen shown in Fig. 3 lists available room choices in the lower right portion of the screen which are linked to detailed descriptions and requirements of each room. The process from which the screen is entered, that is, "Explore Neoforma Healthcare," appears at the upper left hand corner of the screen shown in Fig. 3, while the screen being viewed by the user is identified in the lower right hand corner ("Department Gallery"). The department gallery enables easy, mouse-driven navigation from room to room within the department. The department gallery provides three-dimensional department visualization as shown in Fig. 3, as well as specific information regarding the location and relationship of rooms for design optimization and implementation.

[0048] As shown in the upper left portion of Fig. 3, the screen was submitted by a product vendor. The information contained in the screen is therefore identifiable with a specific source and is unmodifiable, display-only information.

[0049] A rooms gallery is enabled by the user selecting any room from the department gallery. The rooms gallery enables the user to access information stored in the relational database that provides an in-depth description of the equipment recommended for that room. The screens contained in the rooms gallery displays only product information relevant to the selected room, so that the product selection and evaluation process is focused. The screens contained in the rooms gal-

lery also present product categories sorted in logical, functional, and room-determined groupings.

[0050] A products gallery is enabled when the user selects any product category either by clicking the mouse on the three-dimensional representation of the category or by choosing from the categories listing. The screens contained in the products gallery display listings of vendors and their products. The products gallery provides detailed descriptive information submitted by the vendor of the product for each of the listed products. The screens contained in the products gallery also enable the user to review product highlights and choose vendors for further evaluation. The products gallery provides all of the relevant information needed for full evaluation of a product family or specific product for implementation of a component of the project being planned and potential purchase by the user. The screens contained in the products gallery provide detailed descriptions, features, benefits, and specifications relating to the products. The screens contained in the products gallery also preferably provide precise product drawings, photos, and/or graphics so that the user can gain an accurate understanding of the product. The products gallery also preferably directs the user to the product vendor and provides contact information for the vendor by the user clicking mouse 18 on the vendor logo or name.

[0051] A company information gallery enables the vendor to promote its products to the user. The company information gallery contains screens that include corporate profile or background information, awarded certifications, and listings of the international markets in which the vendor is active. The company information gallery links to the products gallery screens for all products that the vendor has listed in the product catalog stored in the database. The screens contained in the company information gallery provide information regarding how the user can directly contact the vendor.

[0052] Unlike other processes listed on the process screen, the process invoked by selection of "Explore Neoforma Healthcare" is a substantially unguided process. This process also enables the user to browse tools available in connection with other listed processes, as well.

[0053] A streamlined process selectable from the process screen shown in Fig. 2 is "Select a product." The user first positions the pointer on "Select a product" to highlight that process, as shown in Fig. 4, and clicks mouse 18 to select that process. Selection of the product selection process causes the computer aided planning tool to access the company information gallery and display a screen, such as the product vendor screen shown in Fig. 5, which lists various products available from that vendor in the lower left portion of the screen. The user next performs the task of selecting a product by pointing to a desired product and clicking mouse 18. The computer aided planning tool user then accesses the product catalog in the database and accesses the product gallery linked to the screen in the company information gallery shown in Fig. 5. This causes a screen in the product gallery, such as the screen shown in Fig. 6, to be displayed from which the user can complete the task of implementing a component of the project, for example, designation of the treatment couch shown in Fig. 6 for a radiotherapy department of a healthcare facility.

[0054] "Linac Shielding - Retrofit" and "Linac Shielding - New Construction" are examples of more complicated processes selectable from the process screen shown in Fig. 2. The user first positions the pointer on "Linac Shielding - New Construction" to highlight that process, as shown in Fig. 7, and clicks mouse 18 to select that process. Alternatively, the user can position the pointer on "Linac Shielding - Retrofit" to highlight that process, as shown in Fig. 7, and clicks mouse 18 to select that process. There are slight differences between these two processes, but a majority of the screens utilized by these two processes are substantially similar. That is, the processes are both linked to shielding screens.

[0055] For example, selection of the linac shielding process for new construction causes the computer aided planning tool to access product information stored in the database for all products in the healthcare industry related to a linear accelerator room, bound intricately into the process of designing a linear accelerator room, and to jump immediately into a specialized tool for designing shielding in part based on access to applicable regulatory information contained in the library information stored in the database. The user can set up project information for the shielding project.

[0056] Additionally, the tool for designing shielding is linked to library information useful in the context of a shielding project, for example, information contained in the library stored on the database relating to similar projects implemented by others. That is, the user can access the room gallery and browse linear accelerator room information to obtain information within the process of designing shielding for a new construction project. This enables the user to explore a community-based and edited library of information, while utilizing the specialized shielding tool.

[0057] Library information stored in the database can also be accessed for used in calculations performed by the specialized shielding tool comprising the linac shielding process for new construction, such as regulations issued by governmental agencies or other organizations which have oversight responsibility for a project that involves shielding. For example, the screen shown in Fig. 9 lists common barriers found in a radiation therapy room, such as walls and doors, as well as links to regulations relating to dose limits that apply to the design of barriers. As shown in the upper left hand portion of the screen shown in Fig. 9, regulations promulgated by Taiwan can be linked to the barrier design for shielding in the "Linac Shielding - New Construction" process.

[0058] The screen shown in Fig. 9 accessed through the linac shielding process for new construction is linked to regulatory information contained in the library information stored in the database. As described above, the particular screen shown in Fig. 9 is linked to regulations promulgated by Taiwan. This evidences the flexibility of the computer aided planning tool to plan projects, such as radiation therapy rooms, worldwide. The regulations that apply to a specific project being planned by the user can also be selected by the user. The user can compare one project design to another as a function of the applicable regulations in one part of world to another. For example, the applicable regulations can be regulations promulgated by the United States of America, instead of Taiwan, as shown in the radiation safety screen shown in Fig. 10. The screen shown in Fig. 10 displays the regulatory information with respect to radiation safety requirements in the United States contained in the library information stored in the database. The information contained in the library also contains information submitted by the community of entities which is supplied with the computer aided planning tool with respect to previous shielding designs, such as barrier walls, that the user can view for suggestions on how to create a wall for the project being planned.

[0059] As in the case of product information stored in the database, library information can also be unmodifiable, display-only information or alternatively can be customizable information which can be imported by the user utilizing a file manager program to be edited for use by the user. In this regard, a new file can be created through a files menu. The creator of the file can then determine whether the file is unmodifiable, display-only information or customizable information. In order to designate the file that is created as unmodifiable, display-only information, the entity that creates the file includes the log-in name and unique password assigned to that entity with the file. For example, the source of the regulatory information displayed in the screen shown in Fig. 10 is identified with the source of that information and is therefore unmodifiable, display-only information. By default, if no such log-in name or password is included, the information is customizable and can be imported by a user and modified for use. If the library information is designated as unmodifiable, display-only information, only the entity that created the file can access that file and update the information in the file. Therefore, the maintenance responsibility for information in the library that is designated unmodifiable, display-only information resides with the entity that created the file.

[0060] The screen for the specialized shielding tool which comprises the linac shielding process for new construction is shown in Fig. 11. The shielding tool can be utilized by the user to optimize the design of barriers to obtain cost savings through linkage of parameters of various materials that can be utilized for a shielding design project to the shielding tool for calculating material thicknesses for barriers, such as walls. As shown in Fig. 11, the shielding tool has not been enabled by the user to calculate shielding requirements, since the default is the "Calculation Off" button at the top of the table. The screen also illustrates that no specific vendor material has been selected for inclusion in the design for the shielding barrier being designed by the user in connection with the project being planned.

[0061] The links to materials from vendors, for example, specific components of material, such as modular walls, or shielding designs utilizing materials provided by various vendors in a design created by the user, are linked to the shielding tool for implementing a project. In this regard, the shielding tool is linked to a gallery of materials information stored in the database and linked to the shielding tool. The materials, such as lead and concrete blocks, can be generic and vendor-specific materials. The materials information includes parameters of various materials, which are used by the shielding tool to calculate required thicknesses of materials utilized in the design of shielding for a radiation therapy room. The screen shown in Fig. 12 indicates selection of a vendor-specific concrete block material by positioning the pointer on the top "Concrete (normal)" line in the table shown in Fig. 11 and scrolling utilizing the up/down arrows to a vendor-specific concrete block material from the product catalog stored in the database and clicking mouse 18 on that material. The selection is linked to the "Linac Wall Materials" gallery in the gallery of materials information. The user can access the vendor-specific information for the wall material by clicking on the selected material in the table shown in Fig. 12, which causes material information to be displayed. The information displayed in response to clicking on the material shown in the table in Fig. 12 is displayed in the screen shown in Fig. 13.

[0062] The shielding tool automatically imports the parameters of the material selected by the user for calculating the shielding thickness requirements for the selected materials. The calculations are initiated by the user selecting the "Auto" check box on the left hand side of the table shown in Fig. 12 and also selecting the "Calculation On" button above the center of the table. In the example calculation illustrated by the screen shown in Fig. 12, the thickness of lead is set by the user, and the shielding tool calculates the thickness of the selected concrete block material that is needed to satisfy the given dosage regulations. The shielding tool utilizes conventional shielding computations to determine thicknesses for materials and calculates thicknesses using the parameters of the selected material, as indicated by the partial product identification and density parameter which also appear in the screen shown in Fig. 13.

[0063] The user can also access specific product information that is used in radiation therapy rooms. The information accessible to the user includes information from the materials gallery. For example, the screen shown in Fig. 13 is linked to the shielding tool shown in Figs. 11 and 12. If the user had simply browsed the

materials gallery for a material for designing a barrier wall and selected the vendor-specific wall material displayed in the screen shown in Fig. 13, the user can import the material parameters contained in the file for the material through the file manager by invoking "Insert Wall" which appears at the left central portion of the screen shown in Fig. 12 and thus select the vendor-specific wall material for calculating the thickness of that material for the shielding project being planned.

[0064] Also, as indicated above, the product gallery for radiation therapy rooms is accessible to the user through the linac shielding process for new construction. For example, vendor-specific equipment from the product gallery can be accessed through the "Linac Shielding - New Construction" process, as shown in Fig. 14. The screen shown in Fig. 14 displays product information with respect to vendor-specific equipment utilized in a radiation therapy room at a healthcare facility. Consequently, there is also a link to the company information gallery, as well, through the linac shielding process for new construction.

[0065] In summary, the linac shielding process for new construction enables the user to plan a design based on community provided information, regulations, generic and vendor-specific materials that can be utilized, and vendor-specific products. Then, the shielding tool calculates material thicknesses for barriers when "Auto" and "Calculation On" which appear in the screen shown in Fig. 12 are selected by the user using the parameters of the materials selected by the user. The user can then access the company information gallery for more information if the shielding design for the project being planned is satisfactory.

[0066] Another process selectable from the process screen shown in Fig. 2 is "Edit a department or room template." The user first positions the pointer on "Edit a department or room template" to highlight that process, as shown in Fig. 15 and then clicks mouse 18 to select that process. Selection of the department or room template editing process causes the computer aided planning tool to access the screens linked to department templates. For example, the department template screens include templates for different departments, such as different templates for a radiation therapy department at a healthcare facility. A screen which provides one exemplary department template for a radiation therapy department is shown in Fig. 16. Since this screen identifies the source of the displayed information in the upper right portion of the screen, the department template shown in Fig. 16 comprises unmodifiable, display-only information. In contrast, any department template which is modifiable, such as a department template in a file created by the user or customizable because the source of the information is not identified, can be imported by the user and modified to design a department relating to the project.

[0067] As shown in Fig. 16, the screen lists various rooms included in the radiation therapy department included in the particular department template that appears in the screen. The user can position the pointer on an individual room identified in the list of rooms and click mouse 18 to access the screen that corresponds to the selected room. Therefore, the user can navigate among rooms of the department shown in the department level screen shown in Fig. 16.

[0068] Finally, another process selectable from the process screen shown in Fig. 2 is "Plan a new room or department." The user first positions the pointer on "Plan a new room or department" to highlight that process, as shown in Fig. 17, and clicks mouse 18 to select that process. Selection of the new room or department planning process causes the computer aided planning tool to access the department planning template screens, such as the department planning template screen for a radiation therapy department shown in Fig. 18. The templates enable the user to view the designs and implementations of similar projects and learn from the prior experience of others who have planned those projects. These templates enable the user to design a department based on gleaning an understanding of information displayed in screens which consist of unmodifiable, display-only information indicated when the particular template is identified with the source that submitted the template or by selecting a screen that displays customizable information and importing that template so that the design embodied in the department template can be edited as needed to create a design for the department being planned. Therefore, the user can create a design or import a template if not designated unmodifiable, display-only information.

[0069] The rooms which comprise the department, such as the radiation therapy department profiled in the screen shown in Fig. 18, are linked to the department planning template. The rooms templates linked to the department planning template can be selected by the user locating the pointer to one or more rooms listed in the department planning template and clicking mouse 18 on the individual rooms. One of the room planning templates accessible upon selection of the "Administration Office" from the department planning template shown in Fig. 18 is illustrated by the room planning template for the "Administration Office" shown in Fig. 19.

[0070] The department and room planning templates which are accessed by the "Plan a new room or department process" can be created by any entity which is supplied with the computer aided planning tool. The entity can create a file that becomes a department and/or room template and designate those/that template unmodifiable, display-only information by including the log-in name and unique password of that entity or customizable information by not including the entity log-in name and unique password. If the source of the template is identified, the responsibility for maintaining the template resides with the identified source, which is the only entity that has access to the information through log-in name and unique password to update or other-

wise modify the information. For example, the department planning and room planning templates shown in Figs. 18 and 19 are customizable department and room information which can be imported by the user and edited to create a design for a project being planned by the user. Templates can be individually or collectively incorporated into the project being planned by the user.

[0071] As illustrated by the screen shown in Fig. 19, the room planning template can be further linked to product information in the products gallery. The user can access the products gallery to complete the process of planning a project by performing the task of implementing the components of the project, such as selecting the equipment, furniture, and accessories, such as interior decorating appointments, based on positioning the pointer on the products listed in the room planning template listed in the screen shown in Fig. 19 and double clicking on the products.

[0072] If department planning and/or room planning templates are unmodifiable, display-only information identified with the entity which is the source of that information or vendor-specific products are linked to the products listed in the room planning templates for implementing components of the room, a link to the company information gallery exists. Therefore, the user can access information for contacting the identified entity or vendor, for example, the web and e-mail addresses, the telephone and facsimile transmission numbers, and street addresses, as well as the company profile, of the identified entity or vendor.

[0073] The user can use an accessed web site address of a vendor, for example, to contact that vendor. The computer aided planning tool enables the user to connect to the web site of a vendor by positioning the pointer on the name or logo of the vendor in the screen which profiles that company in the company information gallery. For example, the vendor identified in the screen shown in Fig. 5 can be accessed through any product screen or unmodifiable, display-only information sourced by that vendor, and the user can position the pointer on the company name or logo and connect to the home page at the web site of that particular company by clicking mouse 18 from that screen. On the other hand, if the user positions the pointer on the company name or logo in a screen in the products gallery which includes a specific product reference, for example, on the name of the company identified in Fig. 13, and clicks mouse 18, the user is connected to the relevant page at the vendor web site that contains information pertinent to the specific product. Therefore, the applicable web site page is directly accessible from the screen in the products gallery relating to that product, which obviates the need for the user to browse the web site of the vendor for specific product information.

[0074] Once the user has planned the design for a project, including a department and room design, the user can utilize the computer aided planning tool to perform the task of actually planning the implementation of the various components of the project being planned, for example, the selection of products for implementation of the components incorporated into the design for the project. For example, the user can select a customizable template for a department for a healthcare facility and select that template as the department planning template for the project being planned, as illustrated in the screen shown in Fig. 20. The user can then access the room planning template linked to that department planning template, as shown in Fig. 21. The user can then select products to implement the components of the room. In this regard, the user can browse the products gallery, and browsed product information can be actively added to the project being planned, as indicated by the selection of a vendor-specific product in the first line of the table which appears in the screen shown in Fig. 21. In order to complete the task of implementing the components of a project being planned, as well as determine the cost of the project, the user may need additional information from various vendors of products. This task can be facilitated by two-way connectivity to the various vendors by user broadcasted e-mail.

[0075] After the user has completed the task of assembling a table of vendor-specific products for implementing the components of a project being planned, the user decides what, if any, additional product information is needed. The user then generates pre-formatted e-mail messages, for example, "I am [the user profiled under the user options]. I am requesting [check box(es)] about the [product from the products listed in the table shown in Fig. 21]." The check boxes appear in the lower left hand corner of the screen shown in Fig. 21. These check boxes include "Information," "Quote," and "References." The user can also elect to add specific comments or requests which are appended to the pre-formatted e-mail message, for example, "Please quote in Yen." The additional information sought by the user positioning the pointer to one or more check boxes and clicking mouse 18 is then assembled into a pre-formatted e-mail message to the vendor of each product which appears in the table in the screen shown in Fig. 21.

[0076] The table which appears in the screen shown in Fig. 21 is linked to the company information gallery. Therefore, the e-mail addresses of the vendors which have e-mail addresses are accessed and assembled into the respective pre-formatted e-mail messages. If the vendor does not have an e-mail address, the request is e-mailed to the supplier of the computer aided planning tool, which in turn generates a facsimile transmission incorporating the request to the vendor.

[0077] All of the e-mail requests that are to be broadcast are routed to an e-mail queue, which is accessible from the project being planned and mailed as part of the process. The e-mail requests are sent through a standard e-mail connection over the Internet.

[0078] Each pre-formatted e-mail message also con-

tains "This is from [a user]. Please reply by filling in between the brackets with the requested information." The vendor then replies by entering the requested information between the brackets and responding to the user by way of an e-mail reply over the Internet.

[0079] When e-mail is checked by the user, the user identifies pertinent e-mail which is generically titled "Response to Healthcare Inquiry," and the user saves this e-mail to the e-mail queue. Every time that the computer aided planning tool is opened, the e-mail queue is checked. Replies from vendors that have been transferred to the e-mail queue are parsed, and the additional information that was requested by the user is converted to vendor responses. The information is also entered into the line of the table which appears in the screen shown in Fig. 21 for the particular product to which the additional information relates. As shown in the screen which appears in Fig. 21, an indication is displayed whether or not the vendor has responded. Therefore, status reports can be generated by the user. The information listed in the table which appears in the screen shown in Fig. 21 is also preferable directly linked to a spreadsheet application program for calculating the cost of the project being planned. Once the table which appears in the screen shown in Fig. 21 is complete, an equipment list can be generated for implementing the components of the project being planned. In another embodiment, compatibility templates can be utilized to determine which products are compatible with other specific products based on the information linked to the table which appears in the screen shown in Fig. 21.

[0080] The computer aided planning tool in accordance with the invention provides a platform having a process-based structure unlike known CAD productivity tools or the Radiotherapy Department Toolkit. The process-based structure comprises various planning templates linked to information stored in a relational database related to the task performed utilizing that particular template. The computer aided planning tool enables users to navigate smoothly back and forth from information available in various information galleries to process tools utilized to design the components of the project. The database comprises a product catalog backbone and a library of additional information for targeted access. Unlike known productivity tools, the information is unmodifiable or customizable by designation of the entity which submits that information. For example, product vendors can designate their information unmodifiable and distribute their own files of information to users directly. Database information can be supplemented by enabling users to connect to web sites over the Internet, and unlike the known prior art, a particular page at a web site may be accessed based on information selected during utilization of a particular planning template, rather than requiring users to browse the web site for the relevant page. Finally, automated two-way e-mail over the Internet is provided to facilitate the acquisition of supplemental information which is then accessible by users and can be automatically imported for use in the planning templates, including use in applications programs, such as spreadsheets for calculating costs for the project.

[0081] It will be understood and appreciated that the embodiments of the present invention described above are susceptible to various modifications, changes, and adaptations. For example, the computer aided planning tool in accordance with the invention can also be provided for planning projects in the hospitality (e.g., hotel, restaurant, entertainment), education, retail, and service industry (hair salons, dentists, etc.). All is intended to be comprehended within the meaning and range of equivalents of the appended claims.

Claims

1. A computer-implemented tool for planning a project comprising:

   means for providing at least one planning process tool for planning a given type of project comprising a plurality of planning templates, each planning template for planning a component of the given type of project;
   means connected to the means for providing the planning process tool for providing a database of stored information related to the given type of project;
   means connected to the means for providing the planning process tool for displaying the planning templates for enabling selection of a planning template;
   means for selecting the planning template;
   means responsive to selection of the planning template for displaying at least one planning task for implementing the component of the given type of project;
   means for selecting the planning task;
   means connected to the database for enabling access to at least a portion of the stored information in the database based on the selected planning task, whereby the planning template is linked to information stored in the database that is related to planning the component of the given type of project;
   means connected to the planning template for selecting access to information stored in the database for implementing the component: and
   means responsive to information accessed from the database for displaying the accessed information.

2. A computer-implemented planning tool comprising:

   means incorporated into a computer and executed by the computer to provide a knowledge-based tool for performing at least one planning

process for planning a given type of project, the tool comprising at least one planning template for planning a component of the given type of project;

means connected to the computer for providing a database of stored information related to the given type of project, the database comprising a library of published information, the published information comprising unmodifiable, display-only information and customizable information, the unmodifiable, display-only information being associated with an identifiable source of the published information and the customizable information being associated with an anonymous information source;

means connected to the computer for displaying the planning template for enabling selection of the planning template;

means for selecting the planning template;

means responsive to selection of the planning template for displaying at least one planning task for implementing the component of the given type of project;

means connected to the database for enabling access to the library in the database based on the selected planning task, whereby the planning template is linked to the library stored in the database related to planning the component of the given type of project;

means connected to the planning template for selecting access to the library stored in the database for providing information from the library stored in the database relating to the component; and

means responsive to information from the library accessed from the database for displaying the accessed information.

3. A computer-implemented planning tool comprising:

means incorporated into a computer and executed by the computer to provide a knowledge-based tool for performing at least one planning process for planning a given type of project, the tool comprising at least one planning template for planning a component of the given type of project;

means connected to the means for providing the planning process tool for providing a database of stored information related to the given type of project;

means connected to the computer for displaying the planning template for enabling selection of the planning template;

means for selecting the planning template;

means responsive to selection of the planning template for displaying at least one planning task for implementing the component of the given type of project;

means for selecting the planning task;

means connected to the database for enabling access to at least a portion of the stored information in the database based on the selected planning task, whereby the planning template is linked to information stored in the database that is related to planning the component of the given type of project;

means connected to the planning template for selecting access to information stored in the database for implementing the component; and

means responsive to information accessed from the database for displaying the accessed information.

4. A computer-implement tool comprising:

means incorporated into a computer and executed by the computer to provide a knowledge-based tool for a selection process for a given type of project for selecting a component of the given type of project;

means connected to the means for providing the process tool for providing a database of stored information related to the given type of project;

means responsive to selection of the process for displaying at least one task for implementing the component of the given type of project;

means for selecting the task;

means connected to the database for enabling access to at least a portion of the stored information in the database based on the selected task, whereby the process is linked to information stored in the database that is related to the component of the given type of project;

means responsive to information accessed from the database for displaying the accessed information;

means external to the process tool and database, the external means comprising means for maintaining the information stored in the database for implementing the component;

an Internet connection between the process tool and the external means;

means connected to the process tool for sending a request for component implementing information to a multiple page web site of the external means over the Internet; and

means connected to the external means and responsive to the request for accessing a particular page of pertinent information relating to the component implementing information and communicating the pertinent information on the particular page of the web site to the process tool over the Internet.

5. A computer-implemented planning tool comprising:

means incorporated into a computer and executed by the computer to provide a knowledge-based tool for performing at least one planning process for planning a given type of project, the tool comprising at least one planning template for planning a component of the given type of project;
means connected to the means for providing the planning process tool for providing a database of stored information related to the given type of project;
means connected to the computer for displaying the planning template for enabling selection of the planning template;
means for selecting the planning template;
means responsive to selection of the planning template for displaying at least one planning task for implementing the component of the given type of project;
means for selecting the planning task;
means connected to the database for enabling access to at least a portion of the stored information in the database based on the selected planning task, whereby the planning template is linked to information stored in the database that is related to planning the component of the given type of project;
means connected to the planning template for selecting access to information stored in the database for implementing the component;
means responsive to information accessed from the database for displaying the accessed information;
means external to the planning process tool and database, the external means comprising means for maintaining the information stored in the database for implementing the component;
an Internet connection between the planning process tool and the external means;
means connected to the planning process tool for sending a request for component implementing information to a web site of the external means over the Internet; and
means connected to the external means and responsive to the request for accessing a particular page of pertinent information relating to the component implementing information and communicating the pertinent information on the particular page of the web site to the process tool over the Internet.

6. A computer-implemented tool for planning a project comprising:

means for providing at least one planning process tool for planning a given type of project comprising at least one planning template for planning a component of the given type of project;
means connected to the means for providing the planning process tool for providing a database of stored information related to the given type of project;
means connected to the means for providing the planning process tool for displaying the planning template for enabling selection of the planning template;
means for selecting the planning template;
means responsive to selection of the planning template for displaying at least one planning task for implementing the component of the given type of project;
means for selecting the planning task;
means connected to the database for enabling access to at least a portion of the stored information in the database based on the selected planning task, whereby the planning template is linked to information stored in the database that is related to planning the component of the given type of project;
means connected to the planning template for selecting access to information stored in the database for implementing the component;
means responsive to information accessed from the database for displaying the accessed information and enabling selection of at least one query regarding the displayed accessed information;
means for selecting the query;
means responsive to selecting the query for formatting a request based on the selected query;
means responsive to the request for communicating the request to means external to the planning process tool and database, the external means comprising means responsive to the request for processing the communicated request comprising at least additional information representative of a response to the query and means for communicating the response to the request to the planning process tool; and
means responsive to the response to the communicated request for displaying the additional information related to implementation of the component.

7. A computer-implemented planning tool as defined in one of the claims 1 to 6 wherein the database comprises product information, whereby the database comprises a product catalog.

8. A computer-implemented planning tool as defined in one of the claims 1 to 7 wherein the given type of project comprises a facility comprising rooms and

the planning template comprises means for configuring at least one room design.

9. A computer-implemented planning tool as defined in claim 8 wherein the database comprises product information, whereby the database comprises a product catalog, and wherein the component comprises an item selected from among the group of items consisting of equipment, furniture, and accessories.

10. A computer-implemented planning tool as defined in claim 9 wherein the facility is a healthcare facility and wherein the equipment comprises medical equipment.

11. A computer-implemented planning tool as defined in one of the claims 1 to 10 wherein the database comprises a library of published information, the published information comprising unmodifiable, display-only information and customizable information, the unmodifiable, display-only information being associated with an identifiable source of the published information and the customizable information being associated with an anonymous information source and wherein the means connected to the database for enabling access to at least a portion of the stored information in the database based on the selected planning task enables access to the library in the database based on the selected planning task, whereby the planning template is linked to the library stored in the database related to planning the component of the given type of project, and further comprising:

means connected to the planning template for selecting access to the library stored in the database for providing information from the library stored in the database relating to the component; and
means responsive to information from the library accessed from the database for displaying the accessed information.

12. A computer-implemented planning tool as defined in claim 11, further comprising means connected to the means responsive to information from the library accessed from the database for displaying the accessed information for importing only customizable information from the accessed library information for editing and saving in an edited file.

13. A computer-implemented planning tool as defined in claim 11 wherein the unmodifiable, display-only information corresponds to library information submitted by the identifiable source together with a password assigned to the identifiable source and the unmodifiable, display-only information is maintained by the identifiable source.

14. A computer-implemented planning tool as defined in claim 13, further comprising an Internet connection between the planning process tool and the identifiable and anonymous sources, and wherein library information is accessible from the identifiable and anonymous sources by the planning process tool over the Internet.

15. A computer-implemented planning tool as defined in one of the claims 1 to 14, further comprising means for calculating a quantity of the implementing component.

16. A computer-implemented planning tool as defined in one of the claims 1 to 15, further comprising means for calculating a cost for the quantity of the implementing component.

17. A computer-implemented planning tool as defined in one of the claims 1 to 11, further comprising means for calculating a quantity of the implementing component.

18. A computer-implemented planning tool as defined in claim 3, further comprising means external to the planning process tool and database and an Internet connection between the planning process tool and the external means, the external means comprising means for maintaining the information stored in the database for implementing the component and submitting the maintenance information over the Internet together with a password assigned to the external means.

19. A computer-implemented tool as defined in claim 4 or 5, further comprising means connected to the external means for maintaining the pertinent information relating to the component implementing information.

20. A computer-implemented planning tool as defined in claim 6 wherein the means responsive to information accessed from the database for displaying the accessed information and enabling selection of at least one query regarding the displayed accessed information comprises means for enabling selection of a query selected from among the group of queries consisting of information and quotation and the means responsive to selecting the query for formatting a request based on the selected query comprises means for assembling an e-mail message comprising the selected query and header data containing one of the e-mail address corresponding to the external means and a supplier of the planning process tool and a return e-mail address corresponding to the address of the plan-

ning process tool and the means responsive to the request for communicating the request to means external to the planning process tool and database comprises means connectable to the Internet for transmitting the e-mail message to one of the external means and the supplier of the planning process tool.

21. A computer-implemented planning tool as defined in claim 20 wherein the means responsive to information accessed from the database for displaying the accessed information and enabling selection of at least one query regarding the displayed accessed information comprises displaying a respective check box for enabling selection of a query selected from among the group of queries consisting of information and quotation and wherein information is requested as a default for failure of selection of a check box.

22. A computer-implemented planning tool as defined in claim 20 wherein the e-mail address of the request corresponds to the external means and the request is transmitted to the external means and wherein the means responsive to the request for processing the communicated request comprising at least additional information representative of a response to the query comprises means for assembling an e-mail message comprising the additional information and the return e-mail address corresponding to the address of the planning process tool and the means for communicating the response to the request to the planning process tool comprises means connectable to the Internet for transmitting the e-mail message response to the planning process tool.

23. A computer-implemented planning tool as defined in claim 22, further comprising:

means connected to the planning process tool responsive to the additional information for storing the additional information in an e-mail queue and wherein the means connected to the planning template for selecting access to information stored in the database for implementing the component also enables access to the additional information stored in the e-mail queue for implementing the component and the means responsive to information accessed from the database for displaying the accessed information also is responsive to the additional information accessed from the e-mail queue for displaying the accessed additional information.

24. A computer-implemented planning tool as defined in claim 20 wherein the e-mail address of the request corresponds to the supplier of the planning process tool and the request is transmitted to the supplier, and further comprising means at the address of the supplier for responding to the request by converting the message to a facsimile transmission and sending the facsimile transmission to means for receiving a facsimile transmission at a telephone number corresponding to the external means.

25. A computer-implemented planning tool as defined in claim 6, further comprising means connected to the planning process tool for enabling incorporation of additional customizable information into the query for communication to the external means.

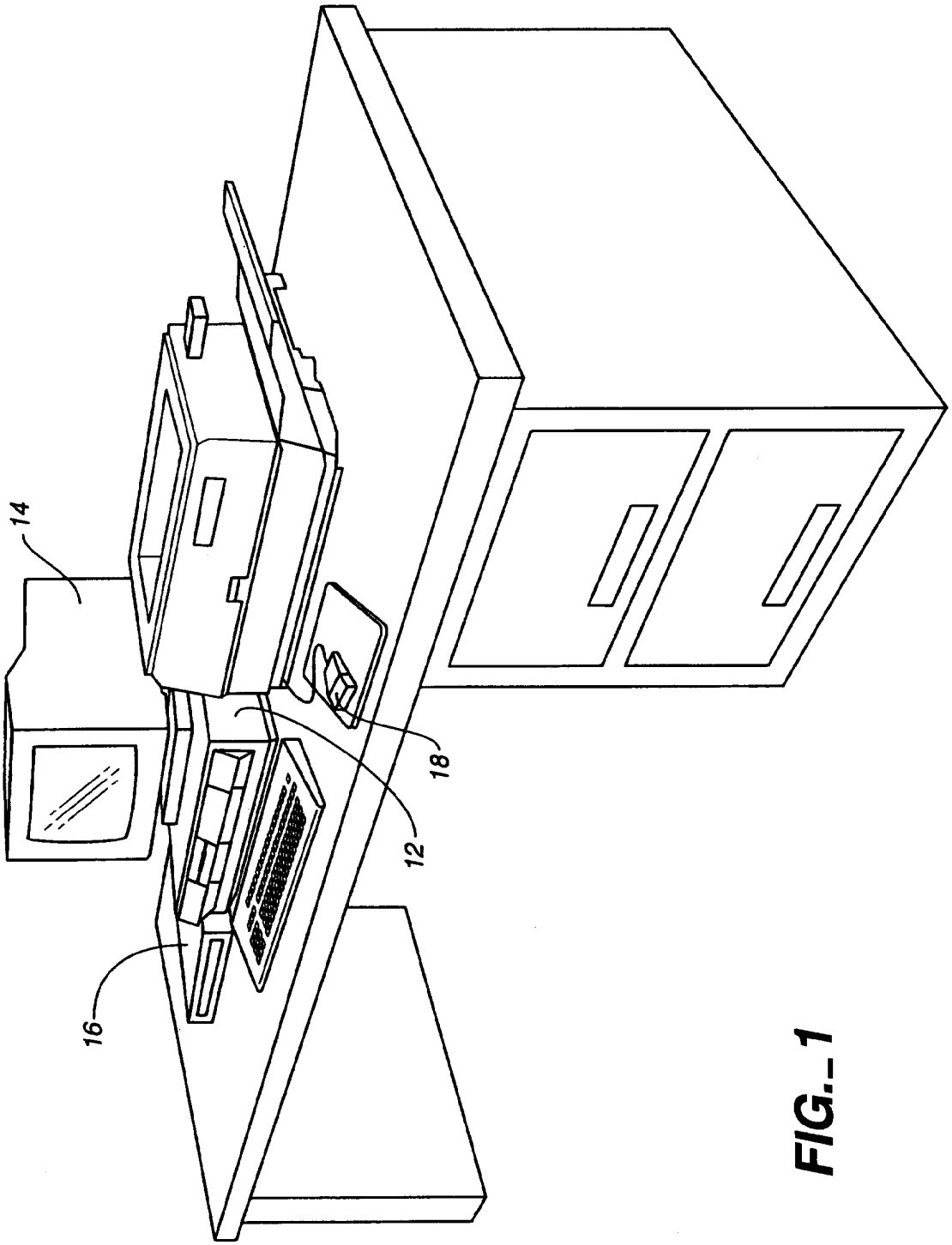


FIG._1

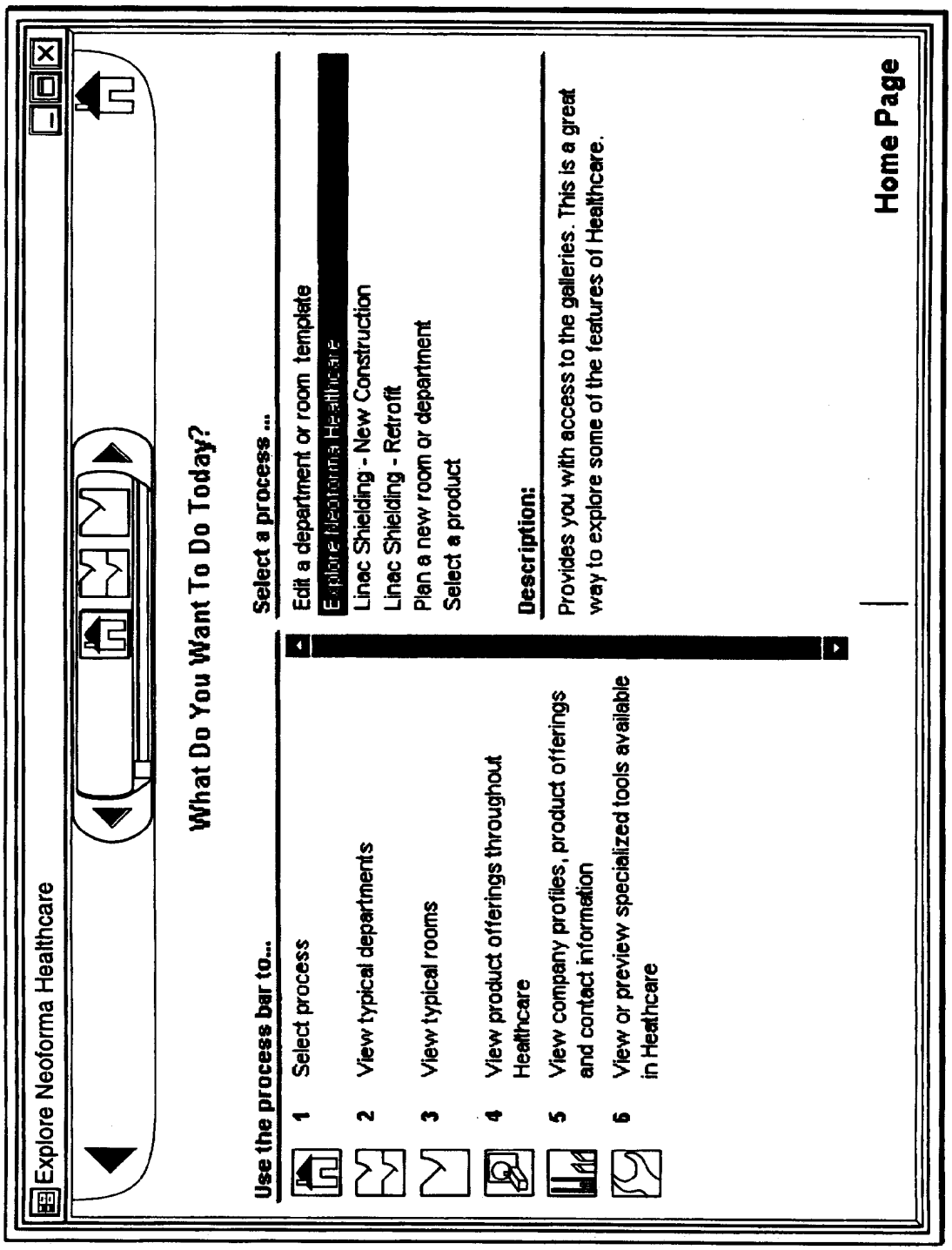


FIG._2

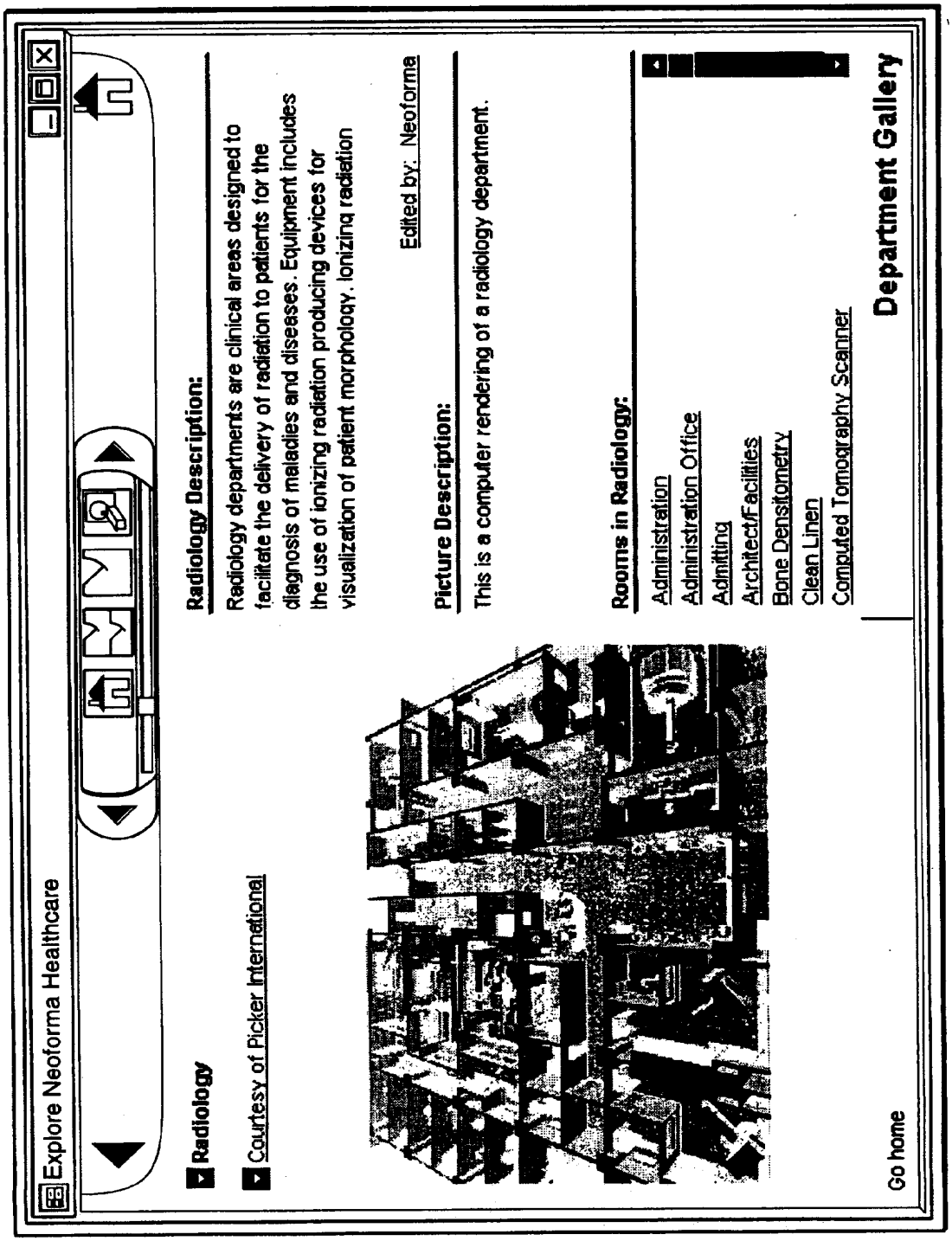


FIG._3

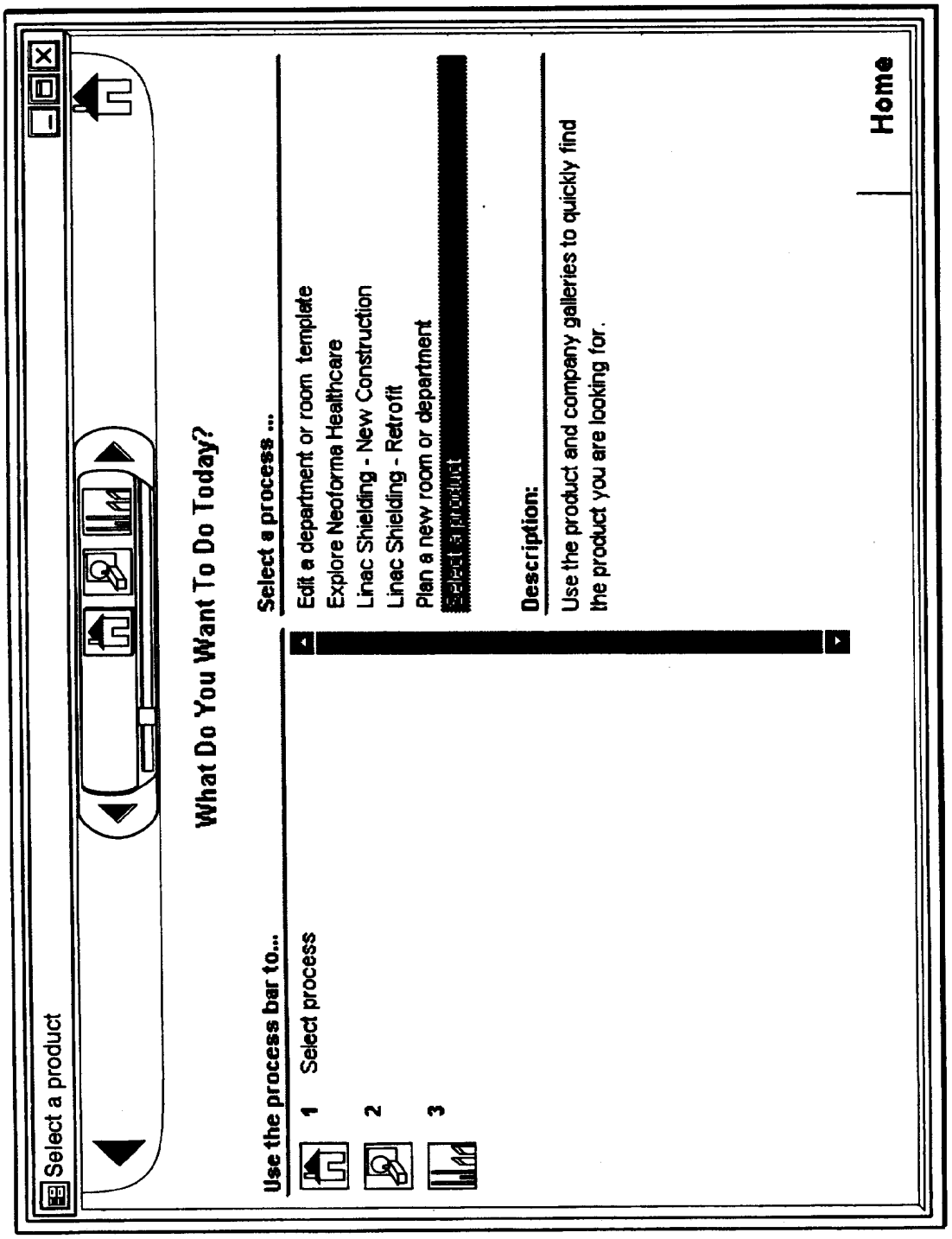


FIG._4

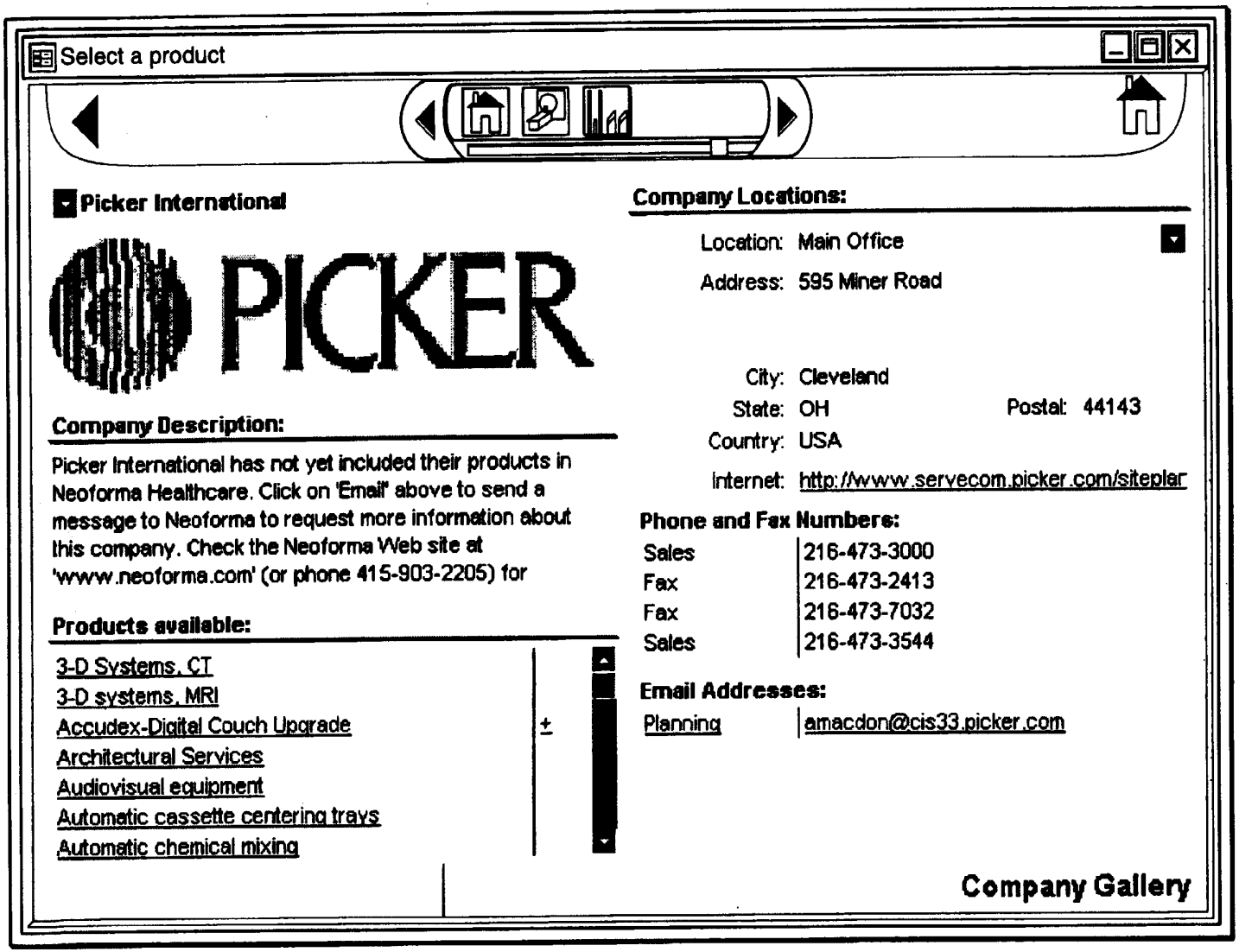


FIG._5

EP 0 895 171 A2

FIG._6

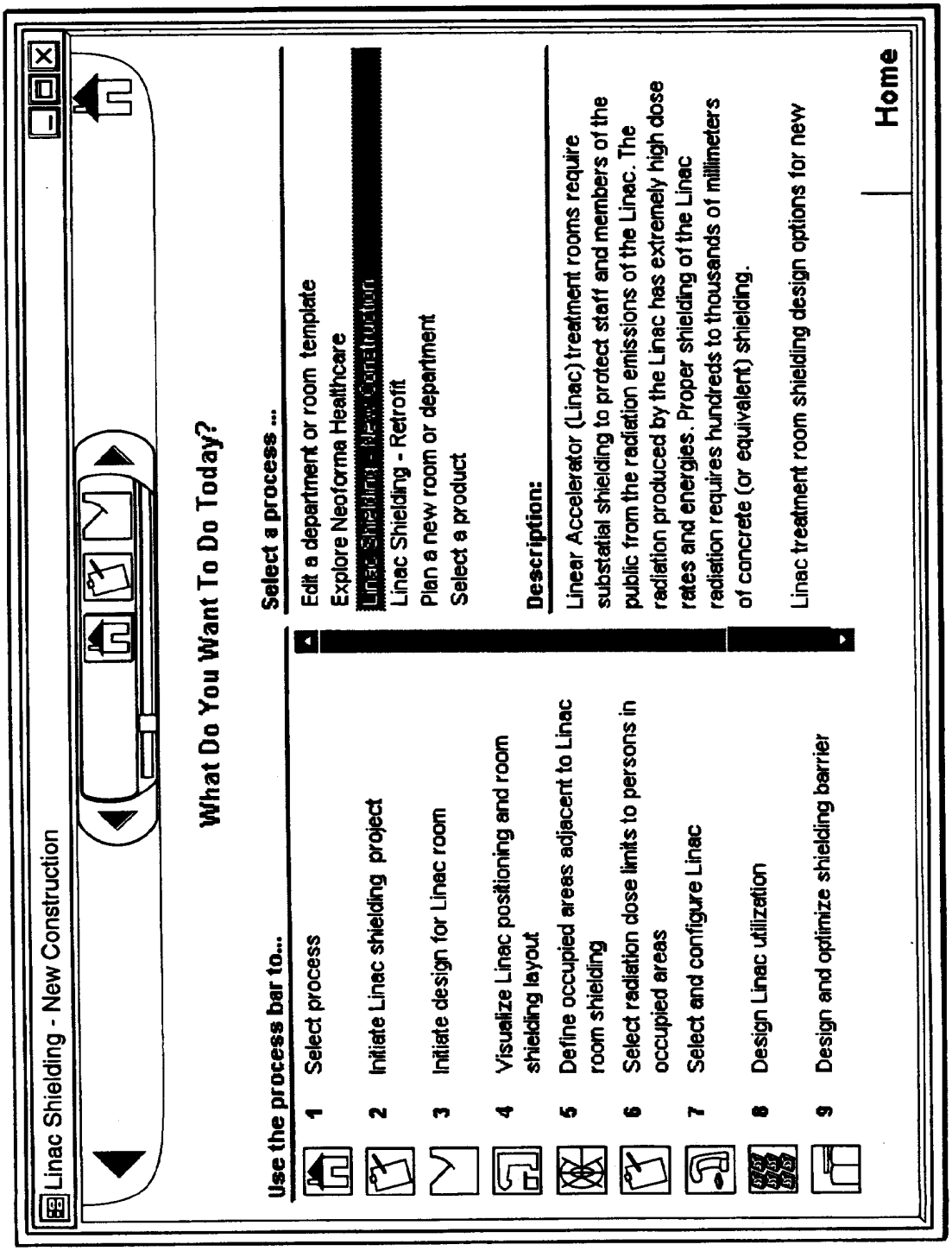


FIG._7

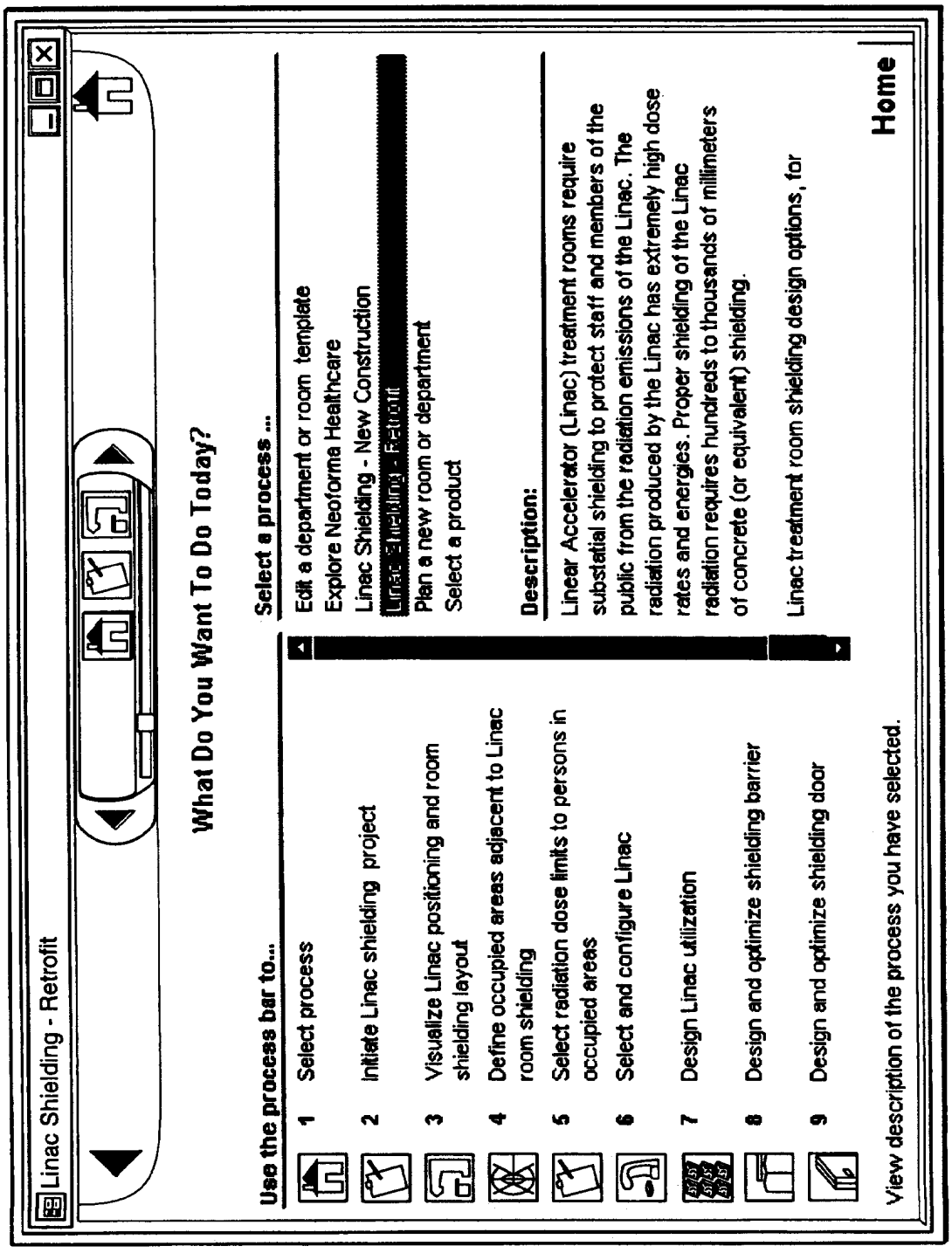


FIG._8

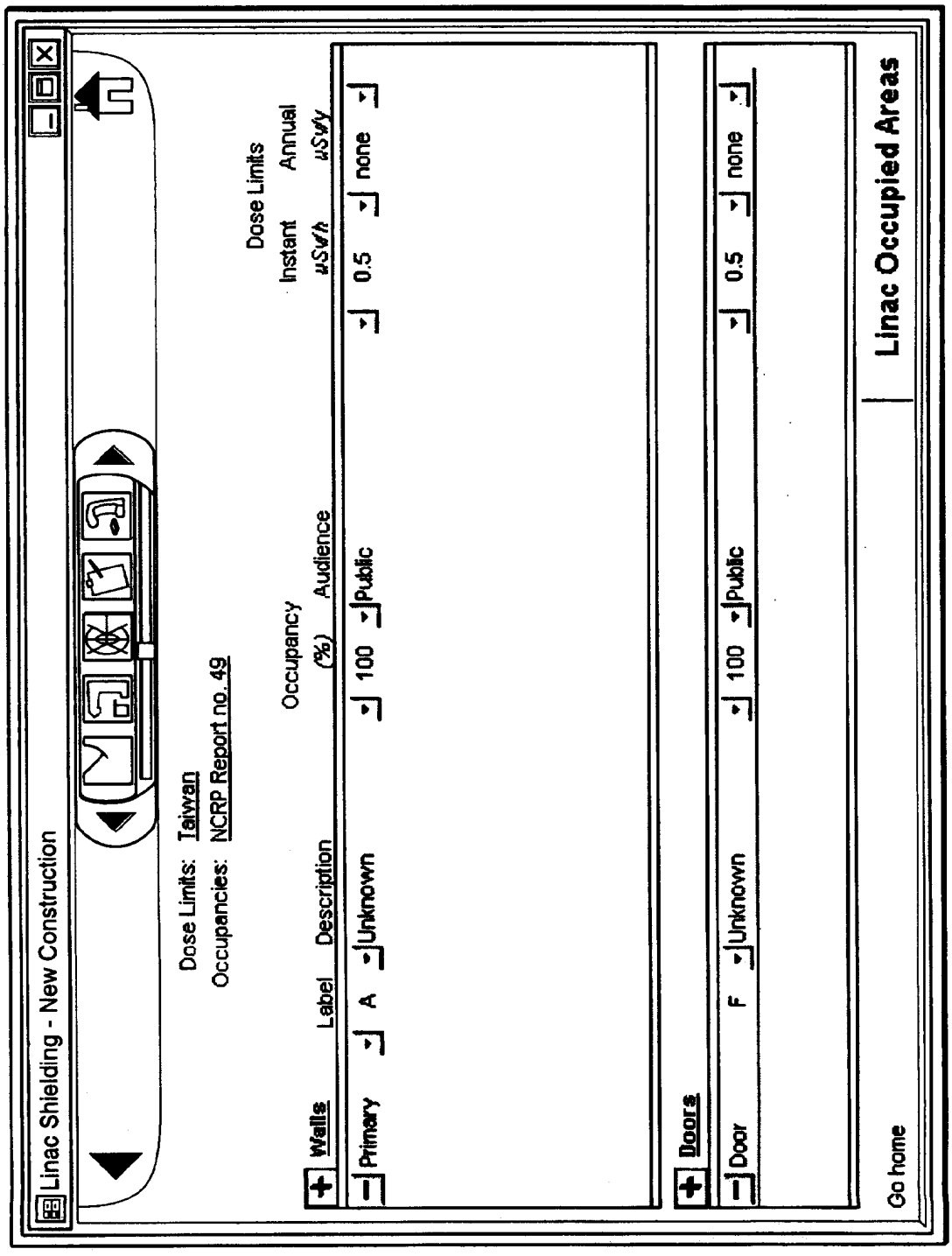


FIG._9

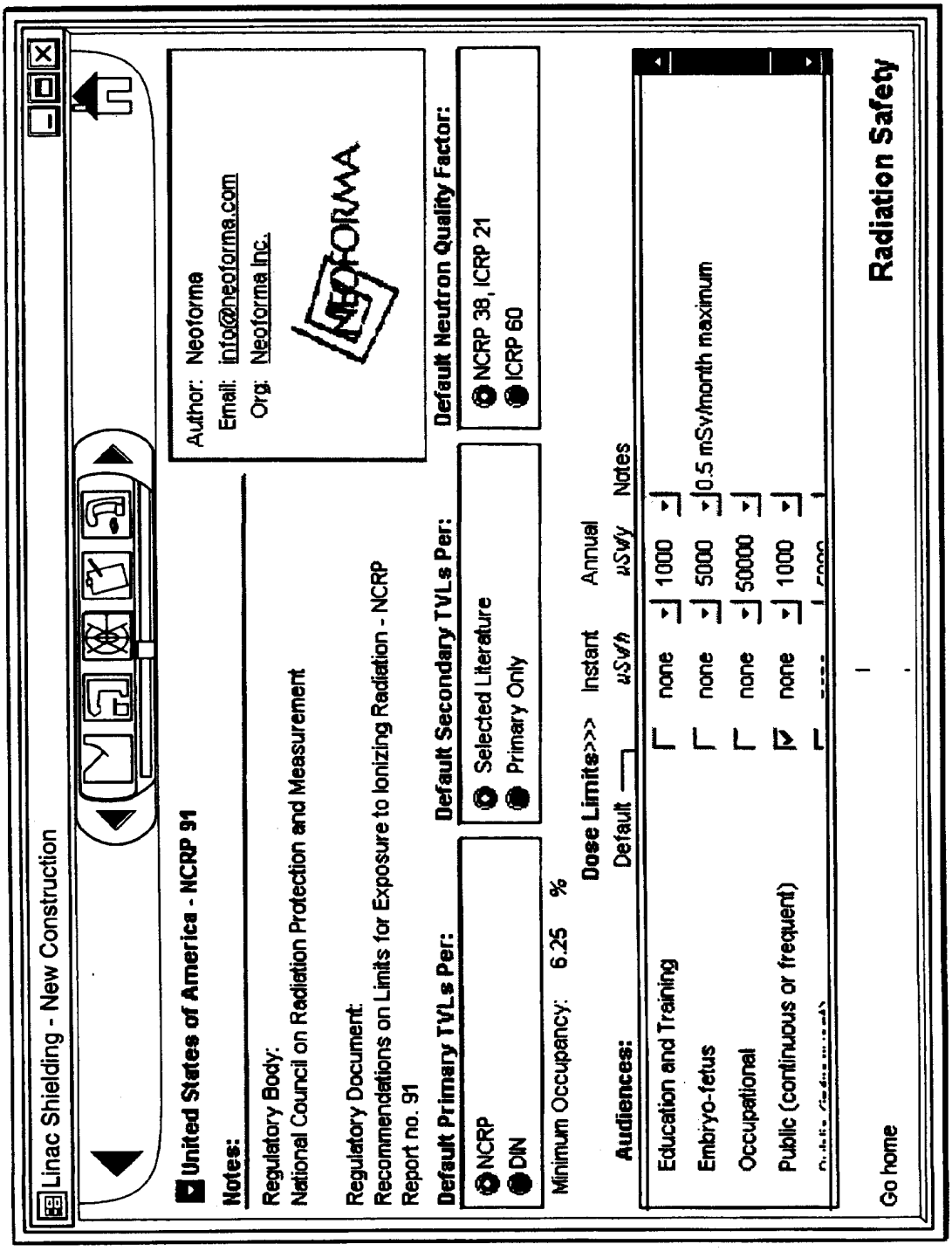


FIG._10

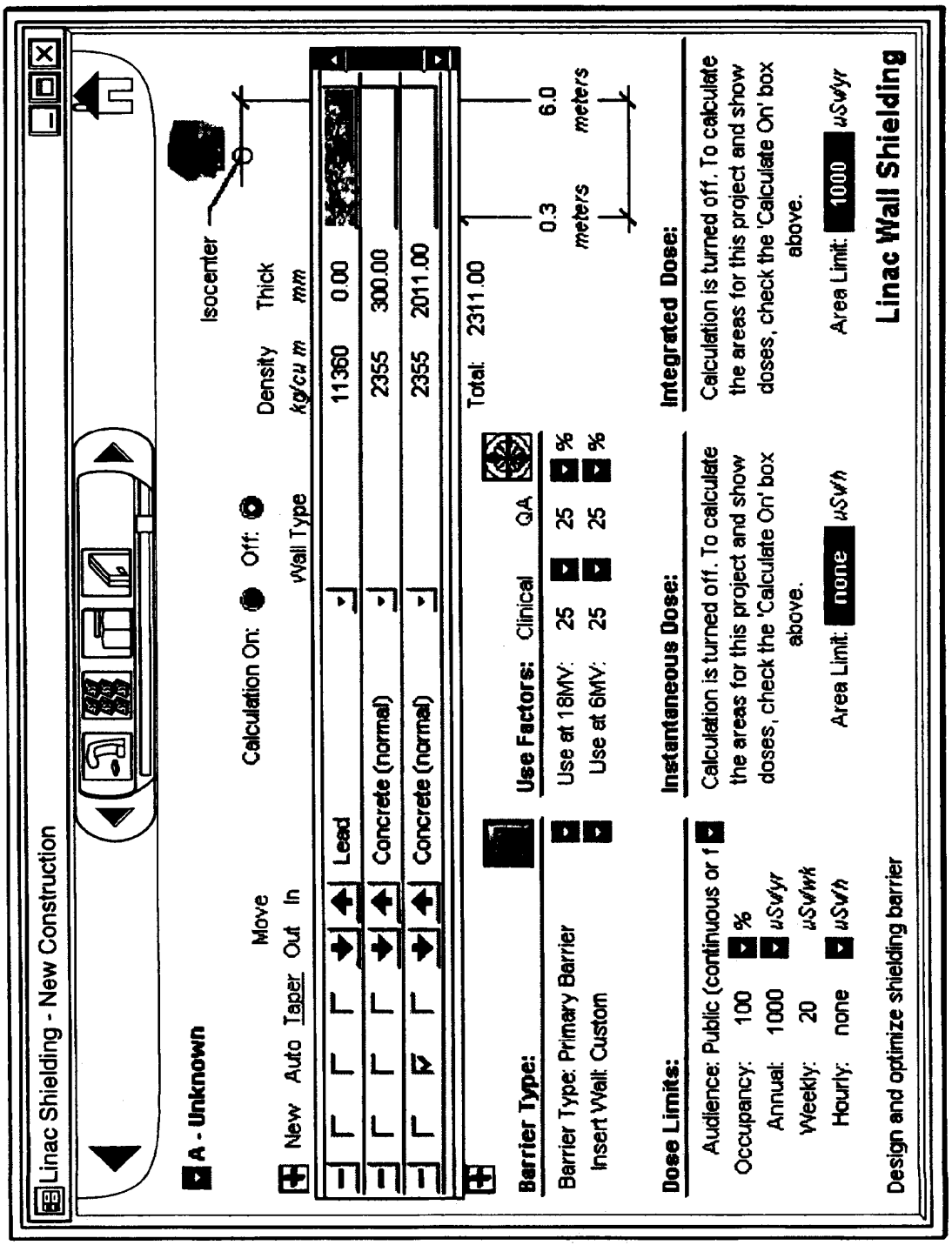


FIG._11

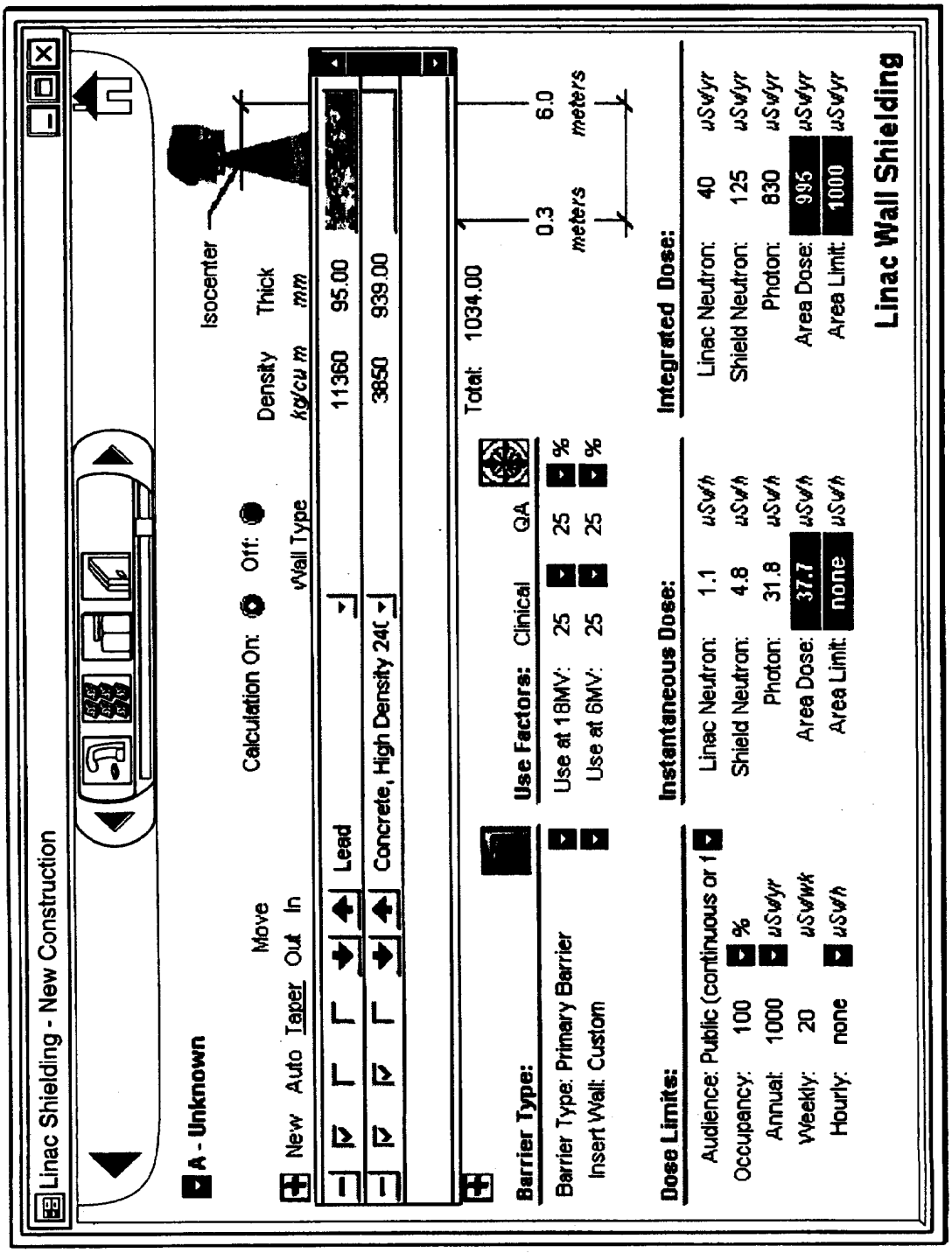


FIG._12

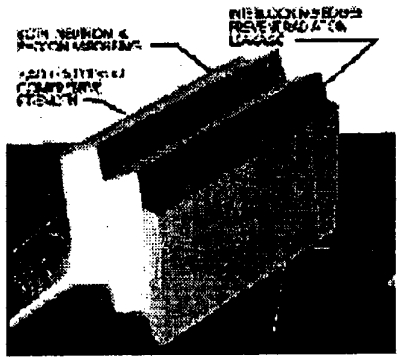
Linac Shielding - New Construction

Concrete Block, Ledite, 240 PCF (3850 kg/ cubic m)

Description (from company):

LEDITEr Accelerator Rooms - Freestanding, modular designed, interior or exterior radiation protective rooms housing Accelerators or High Dose

Shielding Calculation Notes:



Author:
Email:
Org: Atomic International

Material Shielding Properties (The TVLs below are based on the material's standard density):

Density:	3850.0	*kg/cu m*			Thickness Increment:		152.400	*mm*	
Minimum Density:	3465.0	*kg/cu m*							
Maximum Density:	4235.0	*kg/cu m*							
Photon Energy (MV - BJR11):	4	6	8	10	15	18	20	24	
Primary X-ray First TVL (NCRP):	184.0	215.0	230.0	246.0	276.0	292.0	307.0	307.0	*mm*
Primary X-ray >First TVL (NCRP):	184.0	215.0	230.0	246.0	261.0	269.0	276.0	276.0	*mm*
Primary X-ray TVL (DIN):	160.0	193.0	233.0	233.0	248.0	260.0	269.0	269.0	*mm*
Leakage X-ray TVL (Literature):	169.0	184.0	200.0	215.0	230.0	230.0	230.0	246.0	*mm*

Go home

Linac Wall Materials

FIG._13

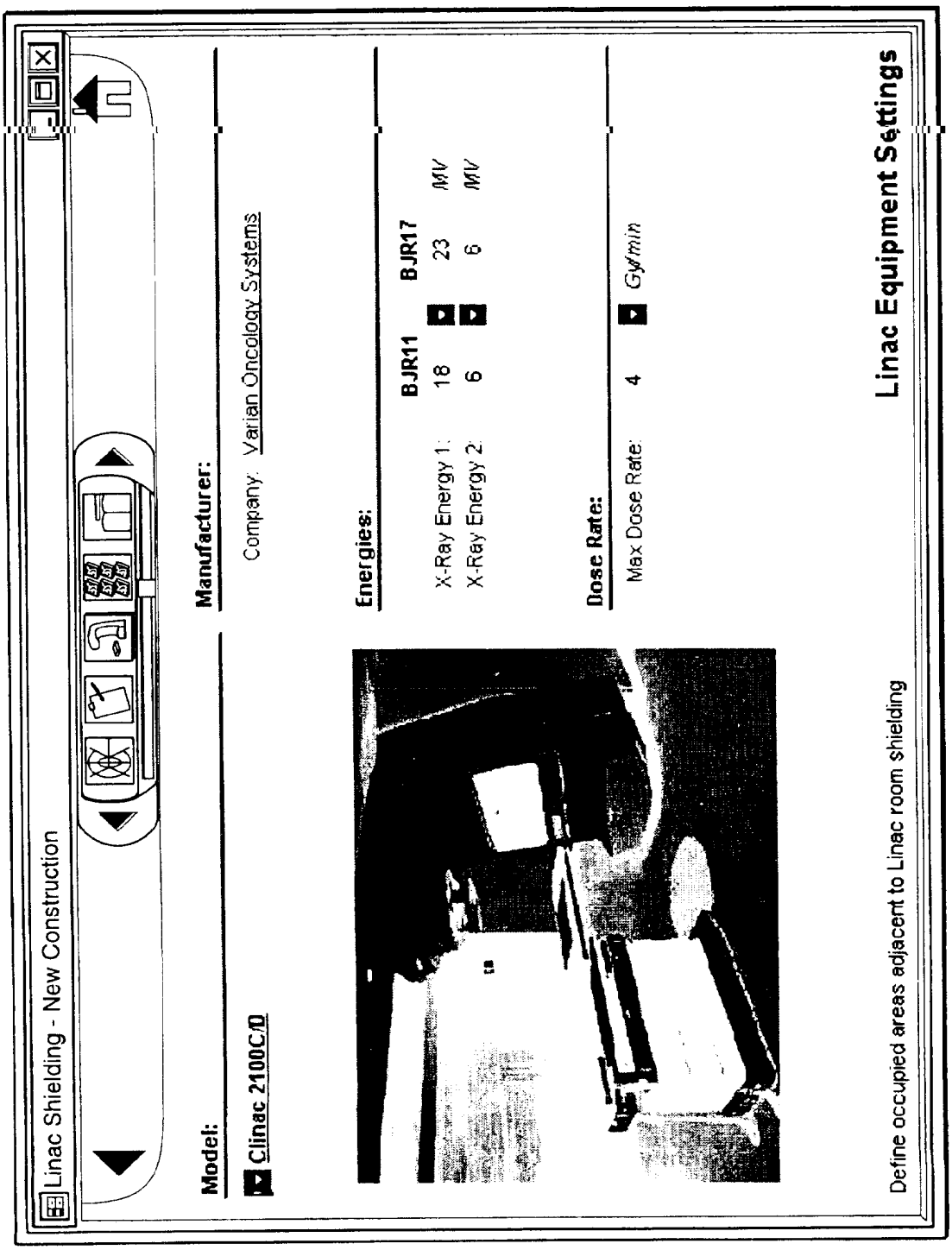


FIG._14

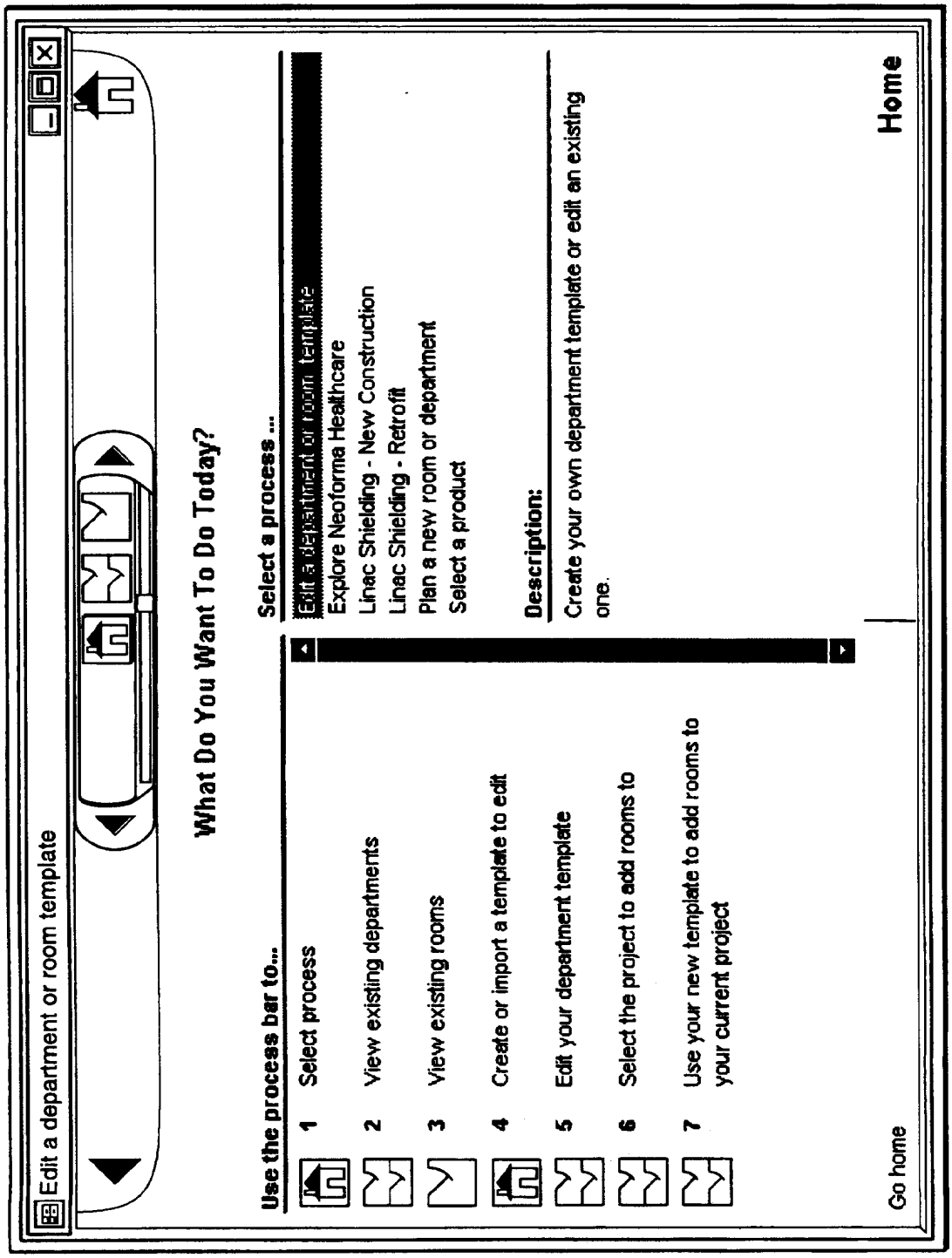


FIG._15

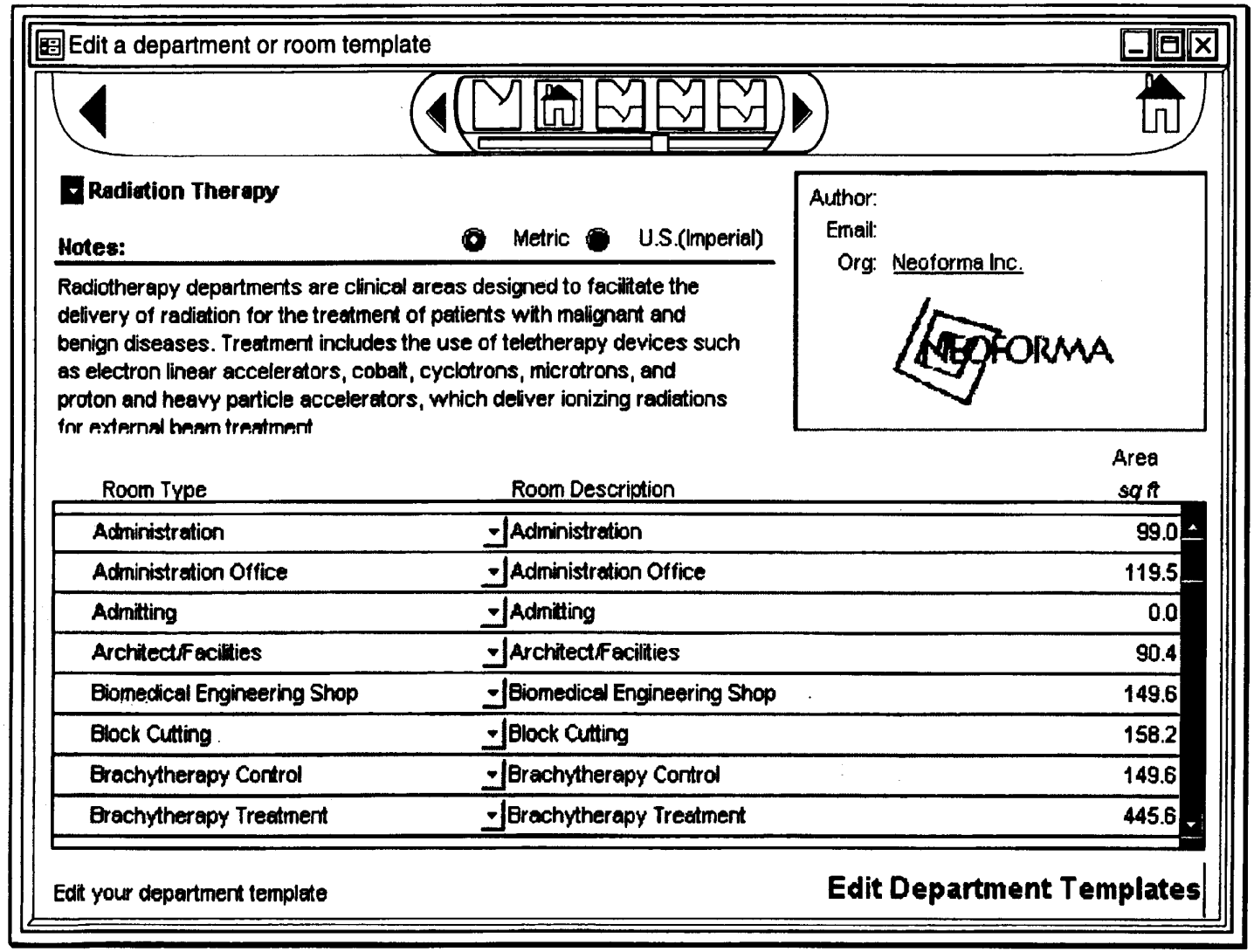


FIG._16

EP 0 895 171 A2

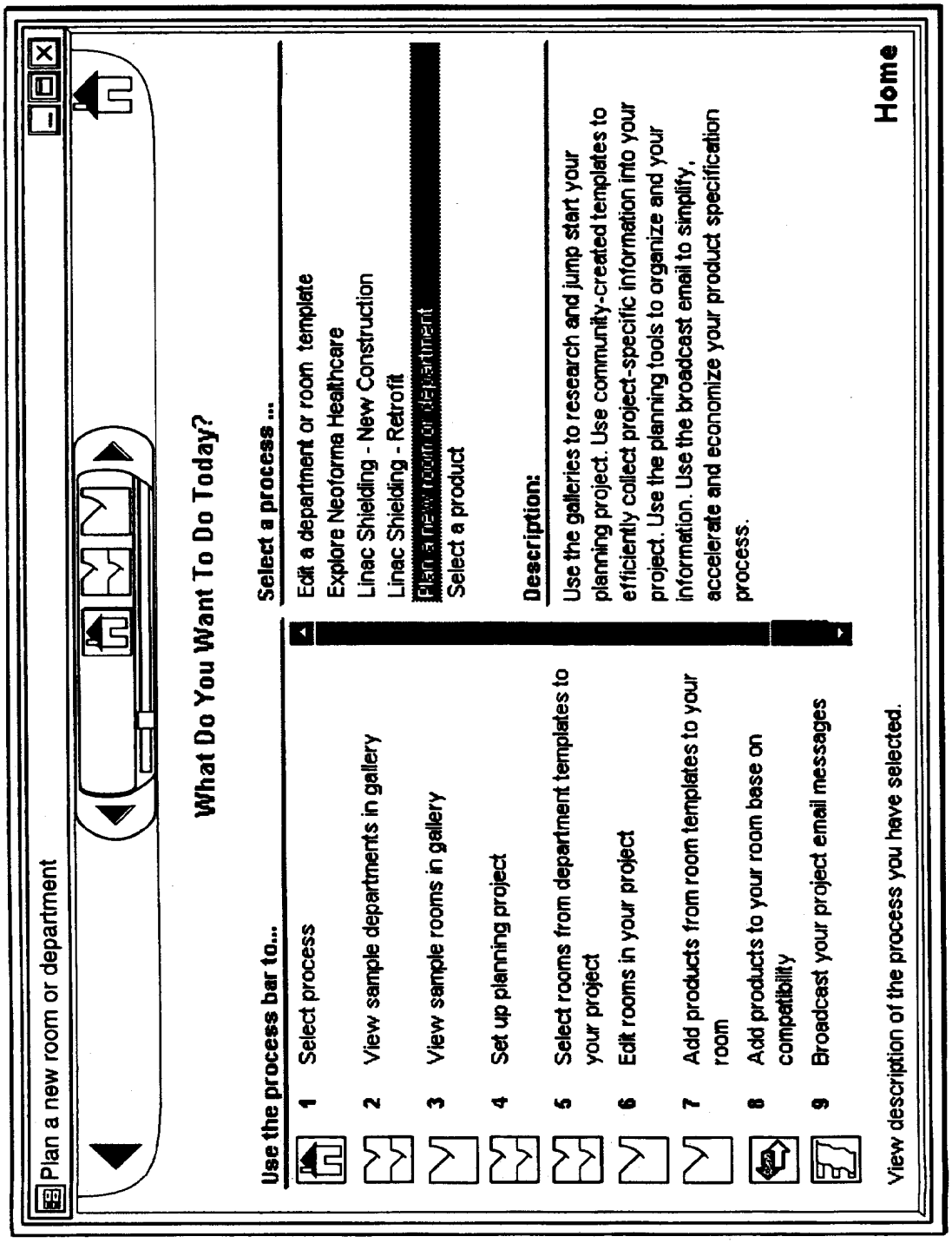


FIG._17

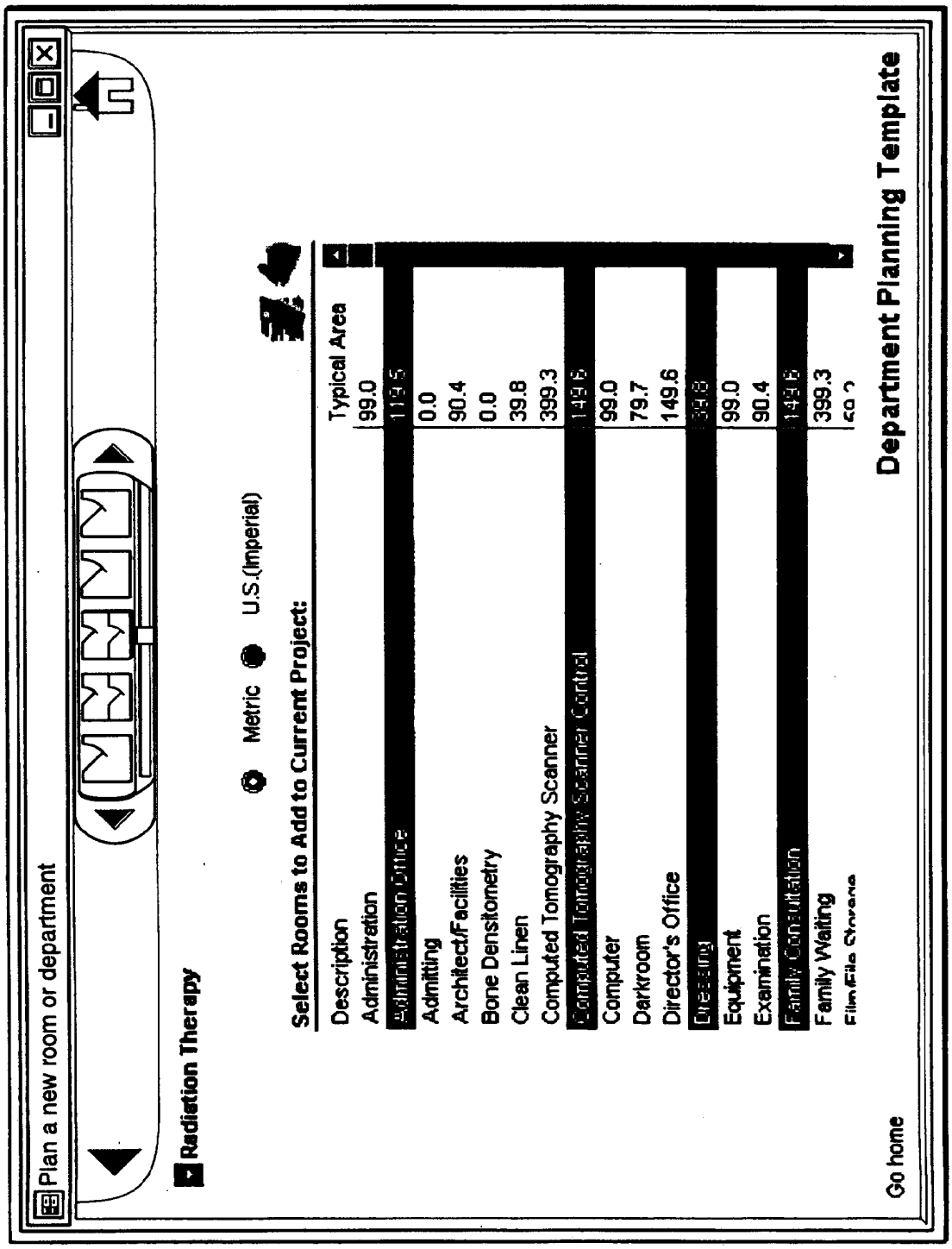


FIG._18

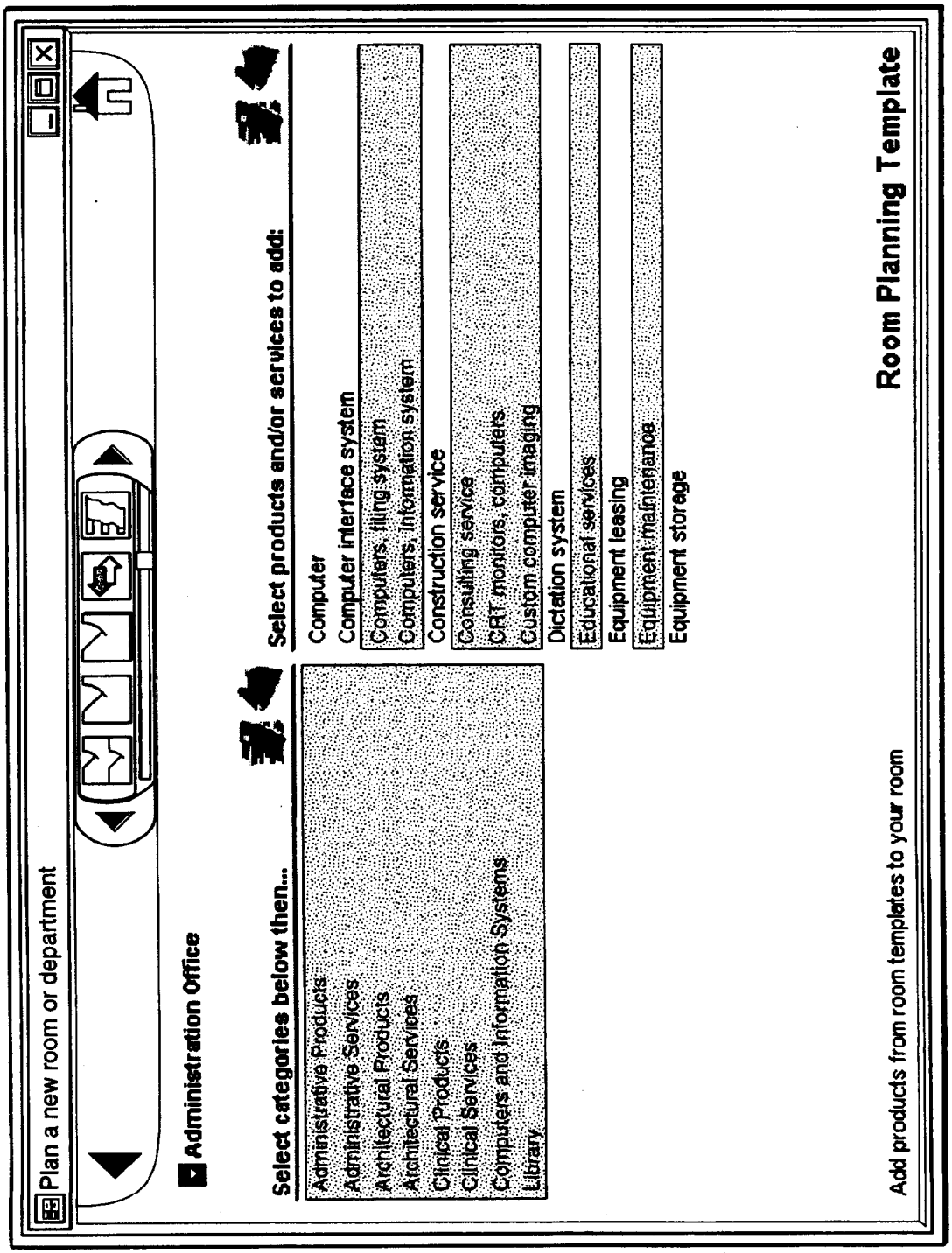


FIG._19

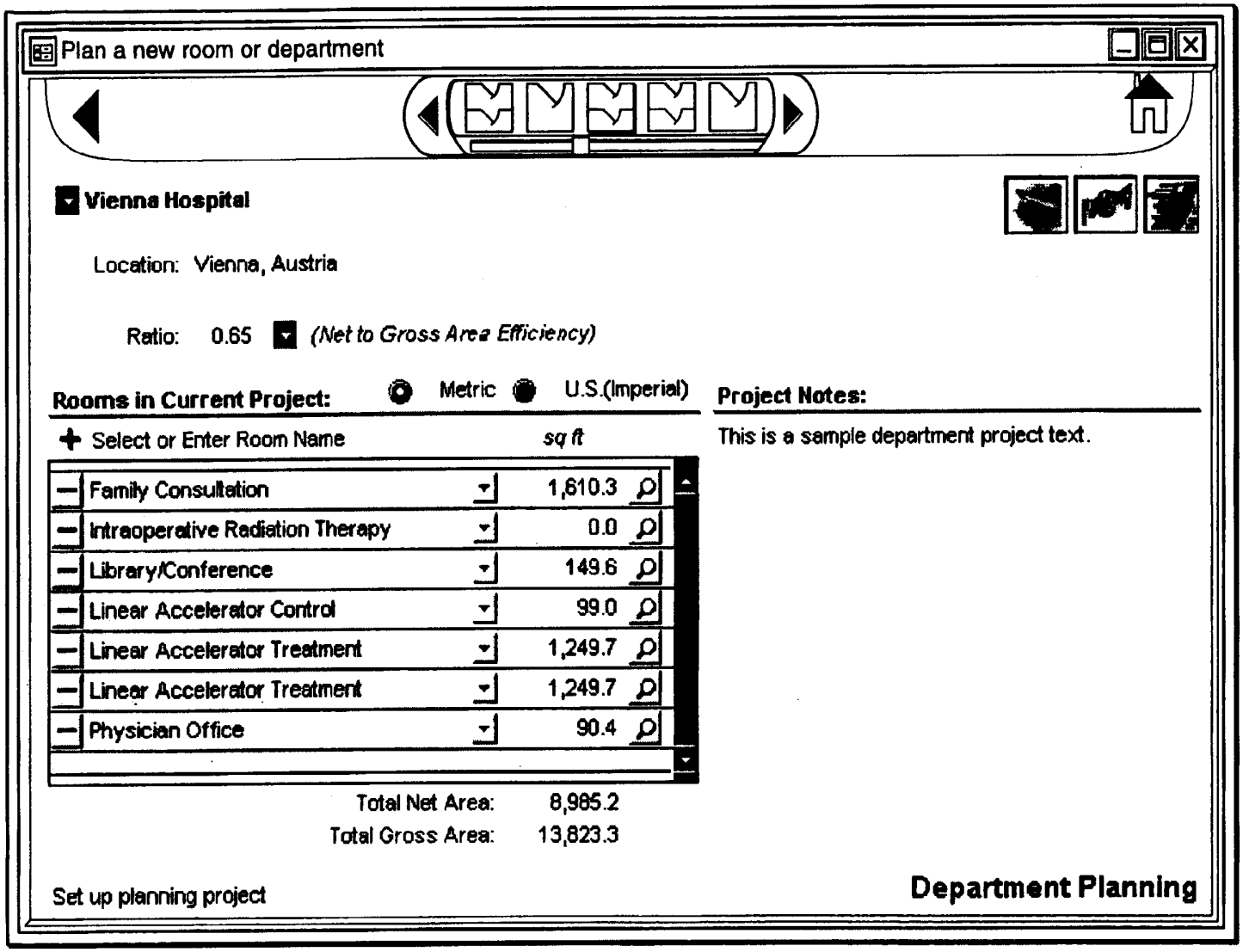


FIG._20

EP 0 895 171 A2

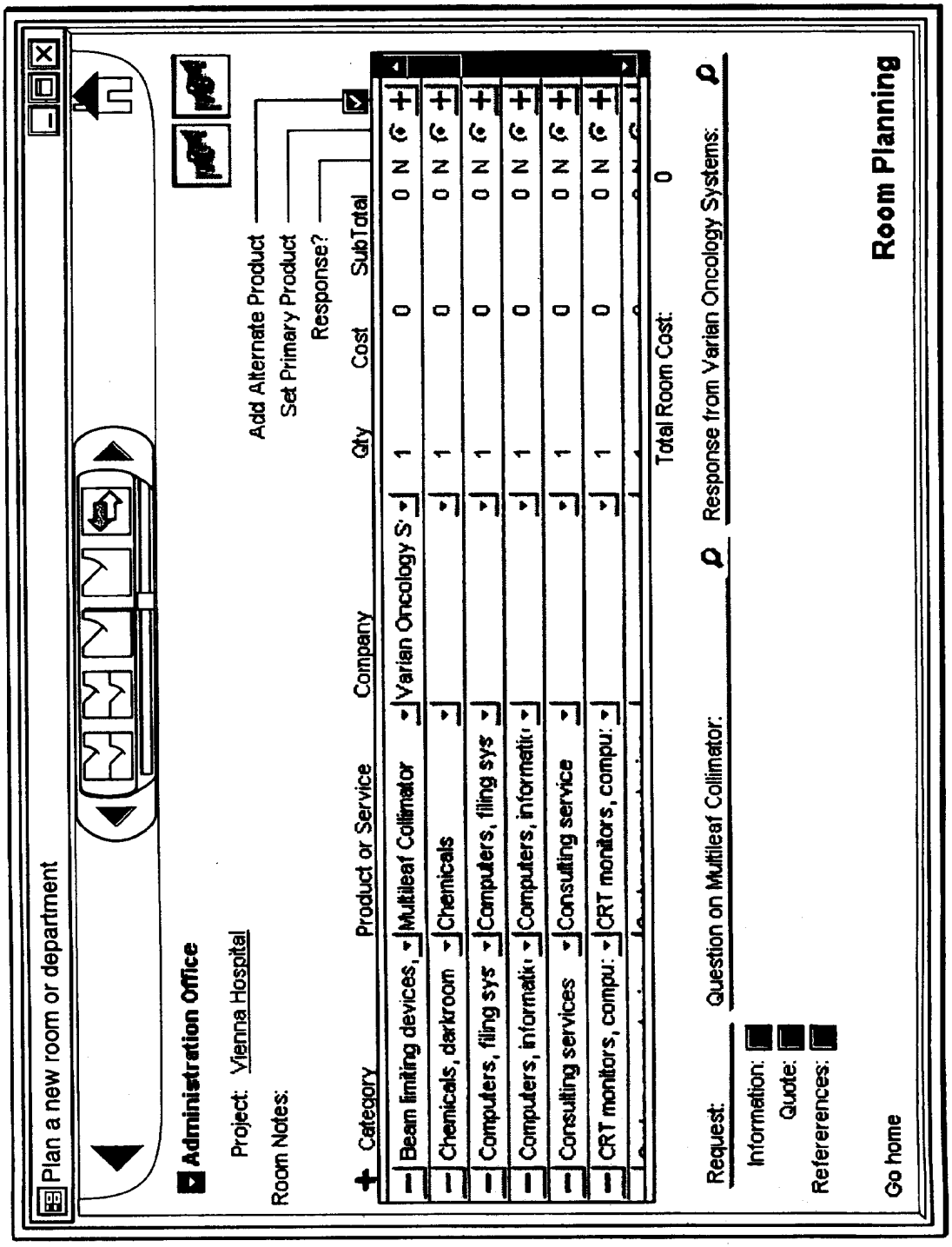


FIG._21

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 926 607 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
30.06.1999 Bulletin 1999/26

(51) Int. Cl.[6]: **G06F 17/30**

(21) Application number: **98124276.1**

(22) Date of filing: **18.12.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **23.12.1997 US 997482**
23.12.1997 US 997705

(71) Applicant: **Ricoh Company**
Tokyo 143-8555 (JP)

(72) Inventors:
- **Motoyama, Tetsuro,**
c/o RICOH CORPORATION,
Systems
San Jose, CA 95134-2088 (US)
- **Fong, Avery,**
c/o RICOH CORPORATION,
Systems
San Jose, CA 95134-2088 (US)
- **Bhatnagar,Anurag,**
c/o RICOH CORPORATION,
Systems
San Jose, CA 95134-2088 (US)

(74) Representative:
Schwabe - Sandmair - Marx
Stuntzstrasse 16
81677 München (DE)

(54) Object-oriented system for mapping structered information to different structured information

(57) An object-oriented system and computer program product for mapping structured information to different structured information, which allows a user to interactively define the mapping. The present invention operates as an object-oriented user tool by accepting interactive input from a user of a source input, by processing the input to display the source input in a format for accepting and processing user commands to create or edit a transformation map of source components to target components. Interactive user input is then accepted and processed for selection of an input file to be transformed and selection of a transformation map to be used for the requested transformation. Interactive user input is accepted and processed for selection of individual components of the first structured information format for mapping, and for selection of options for the target components. Exemplary options for the target components are a null value, the source component itself, a single selected target component, or plural selected target components. Interactive user input is accepted for processing to assign attribute values to components of the second structured information format. Exemplary options for the sources of attribute values are attribute values obtained from the source components, system attribute values, no value, attribute values input interactively by the user, and content of element. Interactive user input is then accepted and processed to initiate processing of a transformation of the source input file in the first structured information format to a target output file in the second structured information format.

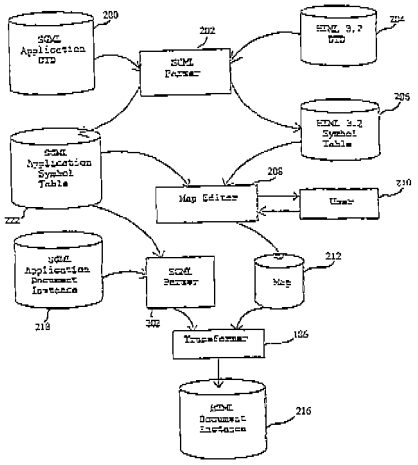


Fig. 5

EP 0 926 607 A2

Description

CROSS-REFERENCES TO RELATED APPLICATION

[0001] This application is related to and being concurrently filed with another patent application: U.S. Patent Application S/N 08/XXX,XXX, Attorney Docket No. 5244-0063-2X, entitled "Method and Apparatus For Mapping Structured information to Different Structured Information" filed on _____, 1997, and incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] This invention relates generally to mapping structured information to different structured information in an object-oriented framework. The present invention relates more specifically to processing a document encoded in a markup language format, a database information format, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, or a DOS file name scheme, transforming it into another markup language format, another database information format, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, or a DOS file name scheme, in an object-oriented framework. The invention is more specifically related to a system and computer program product for mapping in which a user interactively defines the mapping for the transformation in an object-oriented framework.

[0003] This invention also relates generally to providing a user interface for mapping structured information to different structured information. The present invention relates more specifically to providing a user interface for processing a document encoded in a markup language format, a database information format, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, or a DOS file name scheme, transforming it into another markup language format, another database information format, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, or a DOS file name scheme. The invention is more specifically related to a method and apparatus for providing a user interface for mapping in which a user interactively defines the mapping for the transformation.

Discussion of the Background

[0004] Standard Generalized Markup Language ("SGML") is an information management standard adopted by the International Organization for Standardization ("ISO"), as ISO 8879:1986, as a means for providing platform-independent and application-independent documents that retain content, indexing, and linked information. SGML provides a grammarlike mechanism for users to define the structure of their documents and the tags they will use to denote the structure in individual documents. A complete description of SGML is provided in Goldfarb, C. F., *The SGML Handbook*, Oxford University Press, Oxford, 1990, and McGrath, S., *Parseme.1st: SGML for Software Developers*, Prentice Hall PTR, New Jersey, 1998, which are incorporated herein by reference.

[0005] HyperText Markup Language ("HTML") is an application of SGML that uses tags to mark elements, such as text or graphics, in a document to indicate how Web browsers should display these elements to the user and should respond to user actions such as activation of a link by means of a key press or mouse click. HTML is used for documents on the World Wide Web. HTML 2.0, defined by the Internet Engineering Task Force ("IETF"), includes features of HTML common to all Web browsers as of 1995, and was the first version of HTML widely used on the World Wide Web. Future HTML development will be carried out by the World Wide Web Consortium ("W3C"). HTML 3.2, the latest proposed standard, incorporates features widely implemented as of early 1996. A description of SGML and HTML features is given in Bradley, N., *The Concise (SGML) Companion*, Addison Wesley Longman, New York, 1997, which is incorporated herein by reference.

[0006] Object Oriented Programming ("OOP") is a programming methodology in which a program is viewed as a collection of discrete objects that are self-contained collections of data structures and routines that interact with other objects. Many high-level languages, including C++, support the declaration of a class. The class is typically a template, or detailed description, of the objects, or instances of objects, which will be created, or instantiated, by a constructor function during program execution and destroyed by a destructor function when the object is no longer needed. A conversational reference to a class includes all of the objects currently in existence as a result of constructor calls. A class is made up of data items, structures, and methods. Data items correspond to variables of prior programming art. Structures are named groupings of related data items and other structures. Methods correspond to functions and subroutines of prior programming art.

[0007] An object-oriented framework is a reusable basic design structure, consisting of abstract and concrete classes, that assists in building applications.

[0008] Pointers, used for accessing specific objects, data items, and methods, are data items which contain system equivalents of absolute addresses in computer memory. Null pointers, or zero pointers, are pointer variables or literals

which have been assigned a system value, for example, zero, denoting that a specific pointer is currently pointing to a null, or non-existent item. References and reference variables are generally data items which contain system equivalents of absolute addresses in computer memory.

[0009] A string variable or a string literal is a data structure composed of a sequence of characters of the character set of a particular application. A null string, a nil string, or an empty string is a string which contains no characters.

[0010] The three main features of object oriented programming are inheritance, encapsulation, and polymorphism. Inheritance allows a programmer to establish a general class with features which are desirable for a wide range of objects. For example, if a programmer designs a class polygon having certain features such as a closed convex shape made up of plural straight lines joined pairwise at vertices, it is then possible to construct polygon subclasses such as triangles, quadrilaterals, pentagons, and hexagons, all having the shared properties of the parent class polygon, with additional constraints on the number of sides to be allowed for the objects generated. It is also possible, for example, to have subclasses of class quadrilateral such as rectangle and rhombus. A class square inherits all features of the class rectangle and additionally has all of its sides equal in length. The class polygon is considered an abstract class, in that instantiations of actual objects is performed only in its subclasses. However, the class polygon establishes certain properties inherent to all of the non-abstract, or concrete subclasses for inheritance purposes.

[0011] Encapsulation and polymorphism have already been described, and are already well known, in patents relating to object oriented systems. A comprehensive discussion of OOP is provided in Coad, P. and Yourdon, E., *Object-Oriented Analysis, Second Edition*, Prentice-Hall, Inc., New Jersey, 1991, and in Booch, G., *Object-Oriented Analysis and Design with Applications, Second Edition*, Addison Wesley Longman, California, 1994, which are incorporated herein by reference.

[0012] A Graphical User Interface ("GUI") is an environment that represents programs, files, and options by means of icons, menus, and dialog boxes on a screen. An icon is an image, displayed on a screen or other output device, that can be manipulated by a user. By serving as a visual pictorial representation of a function that is available, an icon generates a user-friendly interface by freeing the user of the burden of having to remember commands or type them on a keyboard. A menu is a list of options from which the user can make a selection to perform a desired action. A dialog box is a special window, or area, displayed on a screen or other output device, to solicit a response from the user. In a GUI, the user can select and activate options by pointing and clicking with a mouse or by keystrokes on the keyboard. The preceding descriptions were derived from definitions given in the *Computer Dictionary, Third Edition*, Microsoft Press, Washington, 1997.

[0013] ISO and International Electrotechnical Commission ("IEC") form a specialized system for worldwide standardization. ISO/IEC 9070:1991(E) is an international standard which is applied to an assignment or unique owner prefixes to owners of public text conforming to ISO 8879. The standard describes the procedures for making an assignment and the method for constructing registered owner names from them. Procedures for self-assignment of owner prefixes by standards bodies and other organizations are also specified. ISO/IEC 9070:1991(E) is incorporated herein by reference.

[0014] UNIX and DOS are well-known operating systems for computers. Both UNIX and DOS support a file naming scheme which involve a path from a root directory, through descendant directories, to leaf nodes which are non-directory file names.

[0015] Processing systems are known in which a data processor converts a document encoded in a markup language automatically to another format. For example, Balise software from Computing Art, Inc. processes documents encoded in SGML to convert them to a formatted output for user viewing. However, this software does not allow the user to interactively define the mapping of SGML tags to another format.

SUMMARY OF THE INVENTION

[0016] Accordingly, one object of this invention is to provide a novel object-oriented system and computer program product which can process information encoded in a structured information format to transform the information into another structured information format, and which allows a user to interactively define the mapping for the transformation. Exemplary structured information formats include markup language formats, database information formats, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, and a DOS file name scheme. Exemplary users include human users, software methods, and software objects.

[0017] It is a further object of this invention to provide a novel object-oriented system and computer program product for conversion of Standard Generalized Markup Language ("SGML") documents into HyperText Markup Language ("HTML") documents, allowing a user to interactively define the mapping for the transformation.

[0018] It is a further object of this invention to provide a novel object-oriented system and computer program product for conversion of information in a database format into information in a different database format, which allows a user to interactively define the mapping for the transformation.

[0019] It is a further object of this invention to provide a novel object-oriented system and computer program product

for conversion of information in an ISO/IEC 9070 naming scheme into a UNIX file name scheme, which allows a user to interactively define the mapping for the transformation.

[0020] It is a further object of this invention to provide a novel object-oriented system and computer program product for conversion of information from an ISO/IEC 9070 naming scheme into a DOS file name scheme, which allows a user to interactively define the mapping for the transformation.

[0021] These and other objects are accomplished by an object-oriented system and computer program product for processing information encoded in a structured information format, to transform the information into another structured information format, which allows a user to interactively define the mapping for the transformation.

[0022] An exemplary transformation for the present invention is conversion of SGML documents into HTML documents. For explanation of this example, the present invention has been developed as an object-oriented tool to allow a user to define the transformation of an SGML document into an HTML document or other structured format, for example, a database information format. The user tool for this example is currently implemented in the format of a Graphical User Interface ("GUI") using Object Oriented Programming ("OOP") technology. For this example, the current invention is designed to provide a user with an object-oriented graphic tool to transform documents written in a cryptic SGML format into another structured format for greater viewing ease and for greater portability of documents and information. A user interface object and a map creator object allow the user to select an option of performing a default or conditional mapping. The user is allowed to select an input SGML Document Type Definition ("DTD") object, or a currently existing map object. If the user selects an input SGML DTD object, the user interface object requests that a ParserService object process elements of the SGML DTD into component parts to produce an SGML symbol table object. The user interface object displays individual source component objects of the input for the user to input a selection. A user input object is utilized for accepting the selection. The user interface object and the map creator object provide the user with options for transformation of the individual source component objects such as a mapping of a source component object to a target null value, a mapping of a source component object to itself, a mapping of a source component object to a single target component object, or a mapping of a single source component object to plural target component objects. If the user selects a conditional mapping, then the map creator object checks for special cases, such as a history of an element being referenced previously, and processes special cases using further interactive input from the user by the user interface object.

[0023] The user interface object and map creator object also provide the user with options for assigning attribute values for the target components. Exemplary options are attribute values obtained from the source components, system attribute values, no value, and attribute values input interactively by the user using the user input object.

[0024] The user interface object and map creator object allow the user to interactively select options for transformation, and options for assigning attribute values for the target components, and the selected options are input to objects for properties and objects for attributes. These objects are processed by the map creator object to create a transformation rule object for the source component object.

[0025] The invention accepts and processes interactive user input, using the user input object, for making plural changes to any of the component mapping values the user desires until the user inputs a command to cease the interactive input and create a transformation map. The map creator object initiates processing of the transformation rules to create a transformation map object.

[0026] The user interface object accepts user input into a user input object for selecting an input source file for transformation to a target output file using an already existing map object specified interactively by the user. The user input is then processed, and the requested input file and map are then processed to transform the input file into the requested output file format. The created output file is then sent to the user specified destination.

[0027] Another object of this invention is to provide a novel method, apparatus, and computer program product which provides a graphical user interface for processing information encoded in a structured information format to transform the information into another structured information format, and which allows a user to interactively define the mapping for the transformation. Exemplary structured information formats include markup language formats, database information formats, an ISO/IEC 9070 naming scheme, a UNIX file name scheme, and a DOS file name scheme.

[0028] It is a further object of this invention to provide a novel method, apparatus, and computer program product which provides a graphical user interface for defining conversion of Standard Generalized Markup Language ("SGML") documents into HyperText Markup Language ("HTML") documents, which allows a user to interactively define the mapping for the transformation.

[0029] It is a further object of this invention to provide a novel method, apparatus, and computer program product which provides a graphical user interface for defining conversion of information in a database format into information in a different database format, which allows a user to interactively define the mapping for the transformation.

[0030] It is a further object of this invention to provide a novel method, apparatus, and computer program product which provides a graphical user interface for defining conversion of information from an ISO/IEC 9070 naming scheme into a UNIX file name scheme, which allows a user to interactively define the mapping for the transformation.

[0031] It is a further object of this invention to provide a novel method, apparatus, and computer program product

which provides a graphical user interface for defining conversion or information from an ISO/IEC 9070 naming scheme into a DOS file name scheme, which allows a user to interactively define the mapping for the transformation.

[0032] These and other objects are accomplished by a method, apparatus, and computer program product which provides a graphical user interface for processing information encoded in a structured information format, such as a markup language format, or such as a database information format, to transform the information into another structured information format, such as a markup language format, or such as another database information format, which allows a user to interactively define the mapping for the transformation.

[0033] An exemplary transformation for the present invention is conversion of SGML documents into HTML documents. For explanation of this example, the present invention has been developed as a tool to allow a user to define the transformation of an SGML document into an HTML document or other structured format, for example, a database information format. The user tool for this example is currently implemented in the format of a Graphical User Interface ("GUI") using Object Oriented Programming ("OOP") technology.

[0034] For this example, the current invention is designed to provide a user with a graphic tool to transform documents written in a cryptic SGML format into another structured format for greater viewing ease and far greater portability of documents and information. The user interface provides the user with selectable options of performing a default or conditional mapping. The user interface provides the user with selectable options of selecting an input SGML Document Type Definition ("DTD") or a currently existing map. The user interface displays the input for the user to select individual source components of the input. The user interface provides the user with selectable options for transformation of the individual source components such as a mapping of a source component to a target null value, a mapping of a source component to itself, a mapping of a source component to a single target component, or a mapping of a single source component to plural target components. If the user selects a conditional mapping, then special cases, such as a history of an element being referenced previously, are checked and processed using further interactive input from the user using the user interface.

[0035] The user interface also provides selectable options to the user for assigning attribute values for the target components. Exemplary options are attribute values obtained from the source components, system attribute values, no value, and attribute values input interactively by the user using the user interface.

[0036] The user interface allows the user to interactively select options for transformation, and options for assigning attribute values for the target components, and the selected options are processed to create a transformation rule for the source component.

[0037] The invention accepts interactive user input, to be processed by a map creator, for making plural changes to any of the component mapping values the user desires until the user inputs a command to cease the interactive input and create a transformation map. The transformation rules are processed by a map creator to create the transformation map.

[0038] The invention accepts user input for selecting an input source file for transformation to a target output file using an already existing map specified interactively by the user. The user input is then processed, and the requested input file and map are then processed to transform the input file into the requested output file format. The created output file is then sent to the user specified destination.

BRIEF DESCRIPTION OF THE DRAWINGS

[0039] A more complete appreciation of the invention and many of the attendant advantages thereof will be readily obtained as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, wherein:

Fig. 1A illustrates an exemplary Standard Generalized Markup Language ("SGML") Document Type Definition ("DTD");

Fig. 1B illustrates an exemplary mapping of SGML to HyperText Markup Language ("HTML");

Fig. 1C illustrates an exemplary SGML document;

Fig. 1D illustrates an exemplary HTML document output from a transformation of the SGML document;

Fig. 2 illustrates an exemplary browser output generated using the HTML document shown in Fig. 1D;

Fig. 3A illustrates, in tree format, the hierarchical nature of an SGML document and Fig. 3B illustrates the more "flat" structure of an HTML document;

Fig. 4 illustrates a design of the major components for the SGML to HTML mapping and transformation;

Fig. 5 illustrates, in a data flow diagram format, the flow of data through the SGML to HTML mapping and transformation;

Fig. 6A illustrates the flow of data and interaction of files through the mapping and transformation of information in one structured format to information in another structured format;

Fig. 6B illustrates the flow of data and interaction of files through the SGML to HTML mapping and transformation;

Fig. 7 illustrates a file organization of a map class object for the SGML to HTML mapping and transformation;
Fig. 8A illustrates a map class structure for the map module of the SGML to HTML mapping and transformation;
Fig. 8B illustrates the major classes within the map module of Fig. 8A;
Fig. 8C(1) illustrates a map class structure for a source SGML tag attribute class of the SGML to HTML mapping and transformation;
Fig. 8C(2) illustrates a map class structure for a source SGML content class of the SGML to HTML mapping and transformation;
Fig. 8C(3) illustrates a map class structure far a map service class of the SGML to HTML mapping and transformation;
Fig. 8C(4) illustrates a map class structure for a map create and edit service class of the SGML to HTML mapping and transformation;
Fig. 9 illustrates the hierarchical interaction among major modules of the SGML to HTML mapping and transformation;
Fig. 10 illustrates an exemplary main application window for the SGML to HTML mapping and transformation;
Fig. 11 illustrates exemplary dialog boxes for opening and saving a file;
Fig. 12A illustrates an exemplary window for the Map Processing Option of the SGML to HTML mapping and transformation;
Fig. 12B illustrates an exemplary window for the SGML to HTML Map Editor;
Fig. 12C illustrates an exemplary window for the SGML to HTML Map Editor with sample data displayed in exemplary dialog windows;
Fig. 13 illustrates a class diagram for the Menu Manager for the SGML to HTML mapping and transformation;
Fig. 14 illustrates an object message diagram for startup of the system of the SGML to HTML mapping and transformation;
Fig. 15 illustrates an object message diagram for opening an SGML document for the first time;
Fig. 16 illustrates an object message diagram for opening a new SGML document;
Fig. 17 illustrates the design of the GUI (Graphical User Interface) for the SGML to HTML mapping and transformation;
Figs. 18A(1)-18A(3) illustrate, in object message diagram format, the behavior among the objects of the classes for editing a map for the SGML to HTML mapping and transformation;
Figs. 18B(1)-18C(3) illustrate, i object message diagram format, the behavior of the objects of the classes for assigning values to HTML attributes;
Fig. 19 illustrates a hardware configuration for implementation of the SGML to HTML mapping and transformation;
Fig. 20A illustrates an exemplary public identifier in ISO/IEC 9070 format;
Fig. 20B illustrates an exemplary mapping of ISO/IEC 9070 to a UNIX file name format;
Fig. 20C illustrates an exemplary UNIX file name resulting from mapping the public identifier of Fig. 20A using the map of Fig. 20B;
Fig. 20D illustrates an exemplary user interface display for mapping a public identifier in ISO/IEC 9070 format to a UNIX file name format;
Fig. 20E illustrates an exemplary user interface display for mapping a registered owner field in ISO/IEC 9070 format to a UNIX file name format;
Fig. 20F illustrates an exemplary user interface for selections for a character mapping of a prefix, owner-name component separator in ISO/IEC 9070 format to the UNIX file name format;
Fig. 20G illustrates an exemplary user interface for mapping an owner name character in ISO/IEC 9070 format to valid characters of the UNIX file name format; and
Fig. 20H illustrates an exemplary user interface for a user to map a registered owner component in ISO/IEC 9070 format to a UNIX file name format.

BRIEF DESCRIPTION OF THE APPENDICES

[0040]

Appendix A is an exemplary Standard Generalized Markup Language ("SGML") Document Type Definition ("DTD") corresponding to the tree structure or Fig. 3A;
Appendix B is an exemplary map of SGML elements from the SGML DTD of Appendix A to HTML elements to produce documents which correspond to the tree structure of Fig. 3B;
Appendix C is an exemplary SGML document which conforms to the SGML DTD of Appendix A;
Appendix D is an HTML document which is generated by using the map of Appendix B to transform the SGML document of Appendix C into HTML elements.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0041] Referring now to the drawings, wherein like reference numerals designate identical or corresponding parts throughout the several views, and more particularly to Fig. 1A thereof, there is illustrated an exemplary Standard Generalized Markup Language ("SGML") Document Type Definition ("DTD"). Figs. 1A-1D are presented to illustrate sample inputs and outputs of the SGML to HTML mapping and transformation. The functions performed during the transformation which generate the outputs are described in detail below, and with respect to Figs. 4-6B and the object message diagrams of Figs. 18A(1)-18C(3).

[0042] Figs. 1A-1D and Appendices A-D show exemplary SGML DTDs, SGML documents, maps, and HTML documents produced from transforming the SGML documents using the maps. Figs. 3A-3B show exemplary tree structures for the SGML DTD of Appendix A and the HTML structure resulting from transforming the SGML DTD using the map of Appendix B. Figs. 1A-1D, Appendices A-D, and Figs. 3A-3B illustrate mapping SGML DTDs to HTML DTDs. The problem of DTD to DTD mapping is the default mapping from an SGML instance to an HTML instance based upon the DTDs. SGML tags are either mapped to zero or more HTML tags, and the sources of HTML attributes must be specified in the mapping. A more concise mathematical expression of the problem is given below.

[0043] Let SS be the space generated by the SGML DTD where SS $=\{S_i \mid i=0,...,m\}$, $S_i=\langle$Tag Name$_i$, Attribute Set$_i\rangle$, and Attribute Set$_i$=(Attribute$_j$ of Tag Name$_i$ $\mid$ j=0,...,n_j}+ϕ. Similarly, let HH be the space generated by the HTML DTD specified by W3C (world Wide Web Consortium), that is, HH=$\{H_i \mid i=0,...,k\}$ where H_i corresponds to S_i above. Further, let HG be the space generated by HH consisting of the set of ordered members of HH. Then, HG={null, $\langle H_0\rangle$, $\langle H_1\rangle$,..., $\langle H_0, H_1\rangle$, $\langle H_0, H_2\rangle$,..., $\langle H_1, H_0\rangle$,...}. The sequence of legal HTML tags to be mapped are likely to be found in HG. Then the SGML tag to HTML tag mapping is equivalent to the function F : SS $\rightarrow$ HG + {not-assigned}.

[0044] For purposes of this discussion, {} denotes a set, $\langle ... \rangle$ denotes an ordered set, and + denotes union.

[0045] Let HGG be a set generated from SS, F(SS), and HH. HGG consists of the ordered set of triplets or null. A triplet consists of S_i, F(S_i) and $\langle$HTMLTagName, an Attribute$\rangle$ where HTMLTagName belongs to one of H_j in F(S_i). Assume H_0 has $Attr_0$ and $Attr_1$, H_6 has $Attr_0$, and S_0 and S_1 are mapped to $\langle H_0\rangle$ and $\langle H_3, H_6\rangle$, respectively. Then HGG={null, $\langle S_0, \langle H_0\rangle, \langle H_0$Tag Name, $Attr_0\rangle\rangle$, $\langle S_0, \langle H_0\rangle, \langle H_0$Tag Name, $Attr_1\rangle\rangle$,...,$\langle S_1, \langle H_3, H_6\rangle, \langle H_6$ Tag Name, $Attr_0\rangle\rangle$,...}. Also, let SAtr be the source of the HTML Attribute value. Then Satr = AS+AC+{user inputs}+Null, where AS=the set of ordered pairs of tag name and one attribute name, and AC=the set of tag names with character data content. Then the identification of the attribute source is a function G mapping from HGG to Satr, denoted G: HGG$\rightarrow$SAtr.

[0046] A complete description of SGML is provided in Goldfarb, C. F., *The SGML Handbook*, Oxford University Press, Oxford, 1990, and McGrath, S., *Parseme.1st: SGML for Software Developers*, Prentice Hall PTR, New Jersey, 1998, which are incorporated herein by reference.

[0047] The exemplary SGML document of Fig. 1C, together with the exemplary SGML DTD of Fig. 1A, and the exemplary mapping of Fig. 1B are utilized in a transformation process to generate the HTML document of Fig. 1D. The SGML DTD of Fig. 1A and the SGML document of Fig. 1C are together parsed to produce the structural components of the SGML document of Fig. 1C. These components are then utilized in conjunction with the map of Fig. 1B to transform the SGML document of Fig. 1C into the HTML document of Fig. 1D. Further details of the parsing and transformation are explained below, and with respect to Figs. 4-6B and Figs. 18A(1)-18C(3).

[0048] In Fig. 1A, line 22 is a comment line containing the name of the SGML DTD file. Line 24 is a declaration of an element t containing a model group including elements t1, t2, t3, and t4. The '?' of line 24 indicates that an element is optional. The '*' of line 24 indicates that the model group may occur any number of times in a valid element t, and may also be absent. For this example, the SGML document of Fig. 1C illustrates a valid element t containing elements t1, t2, t3, and t4 in a group in lines 64-70.

[0049] Line 26 of Fig. 1A is a declaration of an element t1 having content type CDATA. The type CDATA means that the element may have a value that consists of general characters. For this example, the SGML document of Fig. 1C includes an element t1 on line 64 having as content the string of general characters '1. Hi Larry'. Usually, content of an element is delimited by a start tag for the element before the content, and an end tag for the element after the content. A start tag typically includes the character '<' followed by the name of the element, followed by optional element information such as attribute information, followed by '>'. An end tag then includes the characters '</' followed by the name of the element, followed by '>'. In SGML, the delimiters '<' and '>' can, by definition, be replaced by other characters.

[0050] Line 28 of Fig. 1A is a declaration for an attribute list for the element t1. An attribute is a property of an element that takes on different values for different instances of elements. For example, an element 'person' typically has an attribute list of attributes 'name', 'age', and 'haircolor'. A particular first person has name="Joe Smith", age ="27", and haircolor="brown", while a second person has name="Sally Jones", age="45", and haircolor="red". If the second person desires, her haircolor attribute is easily changed to haircolor="blond" by an assignment of a different value. For the example of Fig. 1A, on line 28, the attribute list includes an attribute 'name', of type CDATA. The character string '#REQUIRED' is an attribute value indicating that the attribute must be specified. For this example, in the SGML document of Fig. 1C the element t1 on line 64 has a general character content value of "hilarry" assigned to the name

attribute of this element t1.

[0051] Line 30 of Fig. 1A is a declaration for an element t2, of type CDATA. Line 32 is a declaration of an element t3, of type CDATA. Line 34 is a declaration of an element t4, of type CDATA.

[0052] In the SGML to HTML mapping of Fig. 1B, line 42 illustrates a mapping rule of the element t of line 24 to a string of HTML tags and text including '⟨html⟩⟨title⟩Title⟨/title⟩'. Line 44 illustrates a mapping rule of the element t1 of line 26 to a string of HTML tags '⟨H3⟩⟨A NAME="the source is t1's name"⟩'. The sentence between the double quotes of line 44 is a rule rather than an attribute value. Line 46 illustrates a mapping rule of the element t2 of line 30 to an HTML tag '⟨P⟩'. Line 48 illustrates a mapping rule of the element t3 of line 32 to an HTML tag '⟨P⟩'. Line 50 illustrates a mapping rule of the element t4 of line 34 to a string of HTML tags '⟨P⟩⟨A HREF="# the source is t1's name"⟩'. The sentence between the double quotes of line 50 is a rule rather than an attribute value.

[0053] Referring to the exemplary SGML document of Fig. 1C, line 60 shows the document type of the SGML document to be 't', with the DTD corresponding to the SGML document found in the system file "sample1.dtd". Line 62 contains the document start tag '⟨t⟩', which indicates that the lines following line 62 are assumed to follow the format defined in the DTD of Fig. 1A for the element t of line 24. Lines 64, 66, 68, and 70 are exemplary constituent parts of the element t shown on line 24 of Fig. 1A, defined for the exemplary SGML document of Fig. 1C. Line 72 contains the document end tag '⟨/t⟩', signifying the end of the document.

[0054] The HTML document of Fig. 1D is the output of the transformation process utilizing the SGML DTD file of Fig. 1A, the mapping of Fig. 1B, and the SGML document of Fig. 1C. Lines 82, 84, 86, 88, 90, 92, and 94 of Fig. 1D are generated from the information contained in each of Figs. 1A-1C.

[0055] The processing of the exemplary input files illustrated in Figs. 1A-1C to produce the output document illustrated in Fig. 1D will now be described. First, the SGML document line 60 of Fig. 1C is analyzed to determine the document type of the input SGML document and the name of the system file where the SGML documents DTD is stored. This causes the transformer to output the DOCTYPE HTML tag illustrated in line 80 of Fig. 1D, and to open the referenced system file for accessing the DTD for the current SGML document. A check is performed to determine that the DTD does correspond with the current SGML document. Next, the SGML tag of line 62 is parsed so that the current SGML tag becomes '⟨t⟩'. The map of Fig. 1B is referenced to determine that the current SGML tag is the start tag for the current SGML document, and maps to the HTML tag string '⟨html⟩⟨title⟩Title⟨/title⟩'. An HTML tag '⟨html⟩' is saved for the current SGML tag '⟨t⟩' and is output to line 82 of Fig. 1D. The HTML tag substring '⟨title⟩Title⟨/title⟩' is output to line 84 of Fig. 1D.

[0056] The first SGML tag in the string f line 64 of Fig. 1C is now obtained. The current SGML tag becomes '⟨t1⟩'. The transformer obtains the current attribute name and attribute value for the current SGML tag, obtaining an attribute called 'name' with a value "hilarry". The DTD of Fig. 1A is examined to determine that the SGML tag corresponds to the SGML element defined in line 26 of Fig. 1A, with its corresponding attribute list established in line 28 of Fig. 1A. The map of Fig. 1B is analyzed to determine that the current SGML tag's rule is line 44. The mapped HTML string '⟨H3⟩⟨A NAME=' followed by the current value of the name attribute for the SGML element t1, which is currently "hilarry", is then output to the HTML document on line 86 of Fig. 1D. An '⟩' is then output to terminate the tag. The HTML tags '⟨H3⟩' and '⟨A⟩' are saved for the current SGML tag. Next, the parser recognizes text that will be output to the HTML file on line 86 of Fig. 1D. The parser then recognizes the SGML end tag '⟨/t1⟩', at which point end tags for all the HTML tags currently saved for '⟨t1⟩' are output to the HTML document on line 86 of Fig. 1D in reverse order from which the tags were saved.

[0057] Next, the parser recognizes SGML tag '⟨t2⟩' from line 66 of Fig. 1C. The transformer utilizes the map rule line 46 of Fig. 1B to output HTML tag '⟨P⟩', shown on line 88 of Fig. 1D, and to save the HTML tag for the current SGML tag '⟨t2⟩'. The parser then recognizes text which is output to the HTML file, as shown on line 88 of Fig. 1D. The parser now recognizes SGML end tag '⟨/t2⟩' as terminating the text, at which point an end tag '⟨/P⟩' for the HTML tag currently saved for '⟨t2⟩' is output to the HTML document on line 88 of Fig. 1D.

[0058] The parser now recognizes SGML tag '⟨t3⟩' from line 68 of Fig. 1C. The transformer utilizes the map rule of line 48 of Fig. 1B to output HTML tag '⟨P⟩', shown on line 90 of Fig. 1D, and to save the HTML tag for the current SGML tag '⟨t3⟩'. The parser now recognizes text which is output to the HTML file, as shown on line 90 of Fig. 1D. Next, the parser recognizes SGML end tag '⟨/t3⟩' as terminating the text, at which point the end tag '⟨/P⟩' for the HTML tag currently saved for SGML tag '⟨t3⟩' is output to the HTML document on line 90 of Fig. 1D.

[0059] Next, the first SGML tag in the string of line 70 of Fig. 1C is obtained. The current SGML tag is now '⟨t4⟩'. The transformer obtains the current attribute name and attribute value for an SGML tag, obtaining an attribute called 'name' with a value "hilarry". The map of Fig. 1B is analyzed to determine that the current SGML tag's rule is line 50. The mapped HTML string '⟨P⟩⟨A HREF="#' followed by the current value of the name attribute, which is currently "hilarry", is then output to the HTML document on line 92 of Fig. 1D. An '"⟩' is then output to terminate the tag. The HTML tags '⟨P⟩' and '⟨A⟩' are saved for the current SGML tag '⟨t4⟩'. Next, the parser recognizes text that will be output to the HTML file on line 92 of Fig. 1D. The parser now recognizes the SGML end tag '⟨/t4⟩', terminating the text, at which point end tags for all the HTML tags currently saved for '⟨t4⟩' are output to the HTML document on line 92 of Fig. 1D in reverse order from which the tags were saved for '⟨t4⟩'.

[0060] Next, the parser recognizes the end of the SGML document by recognizing a '⟨/t⟩' tag of line 72 of Fig. 1C. This is interpreted to indicate an end tag for the SGML tag '⟨t⟩'. The HTML tags saved for the SGML tag '⟨t⟩' are then obtained and an end tag for the HTML tag '⟨html⟩', the only tag saved for '⟨t⟩', is output to the HTML document as shown on line 94 of Fig. 1D. This terminates the current processing of the documents.

[0061] Fig. 2 shows an output 100 resulting from opening the HTML output document of Fig. 1C as an exemplary What You See Is What You Get ("WYSIWYG") output of a Web browser on a user's computer screen. This output is in a format typically preferred by users of the World Wide Web on the Internet. Users employ Web browser programs to request HTML files, and other file types, from servers on the Internet. The browser downloads a requested HTML file and opens it to display formatted text and images on the user's computer screen. A browser does not have to download a file but is also capable of displaying an HTML file stored local to the computer running the browser or a local area network connected thereto.

[0062] Referring to Fig. 2, the 'Title' line is generated by a browser utilizing line 84 of Fig. 1D. Line 84 includes a start tag '⟨title⟩' and an end tag '⟨/title⟩' to delimit the text of the title to be displayed by the browser, usually in a title bar at the top of the user's computer screen. The '⟨H3⟩' of line 86 of Fig. 1D instructs a Web browser to output non-tag text in a larger, more bold format than normal text output. As the only text appearing after the HTML start tag '⟨H3⟩' is ' 1. Hi Larry', and this is the only text appearing before the HTML end tag '⟨/H3⟩', the text appears on a computer screen enlarged and bold in comparison with the surrounding text. The '⟨P⟩' start tag of line 88 instructs a Web browser to display non-tag text delimited by the start tag and its corresponding end tag in a new paragraph. New paragraphs start on a now line in the output. The end tag '⟨/P⟩' of line 88 instructs a Web browser that this is the end of the current paragraph, and that any non-tag text following this tag will start on a new line on screen.

[0063] On line 86 of Fig. 1D, the tag containing '⟨A name=' is an anchor tag. Anchor tags are used to set place markers in text, and to establish highlighted text that can be clicked on with a mouse to cause a jump to the text containing the place marker. For this tag, a place marker is established for the browser in the non-tag text following the next '⟩' until the '⟨/A⟩' end tag is encountered. Line 92 contains another type of anchor tag containing '⟨A HREF ='. The text appearing on line 92 between the anchor tag's '⟩' and its corresponding end tag '⟨/A⟩' appears on the screen of Fig. 2 as underlined text '(Back to the Hi Larry greeting.)'. This text is typically displayed in a different color from the surrounding text on the screen. When a user clicks a mouse on this underlined text, the text marked by the reference anchor tag of line 86 is pulled in for the user's viewing, surrounded by its neighboring text.

[0064] Fig. 3A and Fig. 3B illustrate the transformation of a hierarchical SGML document tree structure to the more "flat" tree structure of an HTML document. Fig. 3A illustrates a hierarchical SGML document tree structure, whereas Fig. 3B illustrates the corresponding "more flat" tree structure of the HTML document corresponding to the SGML document graphically displayed in Fig. 3A.

[0065] The trees of Fig 3A and Fig 3B are derived from documents illustrated in Appendices A-D. Appendix A shows an exemplary SGML DTD. The tree structure of Fig 3A is derived from the SGML DTD of Appendix A. The tree of Fig 3A has a root node test 110 which has children nodes front 112 and section 114. The node front 112 has children nodes title 116, author 118, and keywords 120. The node section 114 has children nodes number 122, title 124, para 126, and subsec 1 128. The node author 118 has children nodes fname 130, surname 132, and title 134. The node subsec 1 128 has children nodes number 136, title 138, para 140, and subsec 2 142. The node subsec 2 142 has children nodes number 144, title 146, and para 148. The tree structure of Fig. 3A which corresponds to the SGML DTD of Appendix A has five levels and twenty nodes.

[0066] The tree structure of Fig. 3B corresponds to a generalized HTML document that results from utilizing the SGML DTD of Appendix A and a mapping exemplified in Appendix B. Appendix C shows an exemplary SGML document to be processed through the mapping shown in Appendix B to give an HTML document exemplified in Appendix D. The tree structure of Fig. 3B has a root node html 150 having two children nodes, head 152 and body 154. The head node 152 has a child node title 156. The body node 154 has children h3 158 and p 160. The node p 160 has a child strong 162. In contrast to the tree structure of Fig 3A which has five levels and twenty nodes, the tree structure corresponding to the resulting HTML document has only four levels and seven nodes.

[0067] Fig. 4 illustrates an overview of major modules of the SGML to HTML mapping and transformation. A Map Module 184 interacts with a Parser 182 and a GUI 180 to create the actual mapping from an SGML document to an HTML document. A Transformer 186 interacts with the Map Module 184, the Parser 182, and the GUI 180 to transform the SGML document into the HTML document. A Service module 188 contains utility objects which can be utilized by all the modules for utility processing such as file handling. The GUI 180 handles interaction between a user and the system. The Parser 182 analyzes and breaks down input documents into recognizable component parts to be passed to other modules of the system. For example, in processing the exemplary SGML document of Fig. 1C, Parser 182 analyzes the input SGML document recognizing a DTD to generate a symbol table which can be passed to other modules of the system for processing documents. The Parser 182 recognizes line 60 of Fig. 1C as a 'DOCTYPE' tag and processes a DTD specified by a system file "sample.dtd" shown in Fig. 1A to generate the symbol table. The Parser 182 would then recognize line 62 of Fig. 1C contents as a start tag for the element t, and would transmit the tag and other

tag information to Transformer 186. Transformer 186 controls the processing of the SGML to HTML mapping and transformation, requesting information and data from the Map Module 184, the Parser 182, and the Service 188 modules when needed.

[0068] Fig. 5 illustrates a data flow diagram showing the flow of data through the SGML to HTML mapping and transformation. An SGML Application DTD 200 and HTML 3.2 DTD 204 are input to an SGML Parser 202. This SGML Parser 202 corresponds to the Parser 182 of Fig. 4. An SGML Application Symbol Table 222 and an HTML 3.2 Symbol Table 206 are output from the SGML Parser 202 to be utilized as input to a Map Editor 208, along with an interactive User 210 input. The Map Editor 208 is contained within the GUI 180 of Fig. 4. The Map Editor 208 outputs a Map 212. The SGML Application Symbol Table 222 and an SGML Application Document Instance 218 are together input to the SGML Parser 202 to give output to be used as input, along with the Map 212, to the Transformer 186. Transformer 186 corresponds to the Transformer 186 of Fig. 4. Transformer 186 then outputs an HTML Document Instance 216. The SGML Application DTD 200 and SGML Application Document Instance 218 are exemplified in Fig. 1A and Fig. 1C, respectively. The HTML Document Instance 216 is exemplified in Fig. 1D. The Map 212 is exemplified in Fig. 1B.

[0069] Fig. 6A is a more generalized data flow diagram showing exemplary paths taken by data flowing through the generalized mapping and transformation of information in one structured format to information in another structured format. A Structural Description of System A 230, together with a Structural Description of System B 232 and interactive User 210 input, are input to a Map Editor 208 to output the Map 212. The Map 212 and an Instance of System A 238 are then utilized by the Transformer 186 to output an Instance of System B 244.

[0070] Fig. 6B is a more generalized data flow diagram showing exemplary paths taken by data flowing through the SGML to HTML mapping and transformation. An SGML DTD 200, together with an SGML Document 218 and an HTML DTD 260, are input to the Mapping Editor 208 to output the Map 212. The Map 212 and the SGML Document 218 are then utilized by the Transformer 186 to output an HTML Document 216. The HTML Document 216 corresponds to the HTML Document Instance 216 of Fig. 5. The HTML Document 216 is then input to a Browser 262 for user viewing.

[0071] The SGML DTD 200, input to an SGML Editor 256, yields output to the SGML Document 218. A Database Design 250, input to the Mapping Editor 208, along with the SGML DTD 200 and the SGML Document 218, yield output to the Map 212 and a Data Base 254. The Map 212 and the SGML Document 218 are input to the Transformer 186 to yield output, which, together with the output from the Mapping Editor 208, are input to the Data Base 254. The Map 212 corresponds to the Map 212 of Fig. 5. The SGML Document 218 corresponds to the SGML Application Document Instance 218 of Fig. 5. The SGML DTD 200 corresponds to the SGML Application DTD 200 of Fig. 5. The Transformer 186 corresponds to the Transformer 186 of Fig. 5. The Mapping Editor 208 corresponds to the Map Editor 208 of Fig. 5. Arrows illustrate different paths the documents and data files take for different requests of a user.

[0072] Fig. 7 shows a hierarchical view of the DTD Map class object that can be stored in a file. The invention has been implemented using object oriented techniques, although any programming technique and/or hardware may be used to implement the invention. For purposes of this description, a class is a description of the structure and behavior of an object, while an object is an instance of the item described by a class. Objects typically communicate by passing objects and messages to each other. In structure, objects contain other objects or structures as components, as well as variables and methods.

[0073] Interpreting the horizontal lines of Fig. 7 from left to right, begin and end delimiters delimit each object in the file. List Begin and List End delimiters delimit lists from left to right. When a DTD Map Object exists, one or more SGML tag objects are placed between the SGML Tag List Begin 363 and SGML Tag List End 365.

[0074] The file begins with a Header 360 followed by a DTD Map 361. The DTD Map 361 includes, first, a DTD Map Begin 362 followed by an SGML Tag List Begin 363, followed by at least one SGML Tag 364-1 through an SGML Tag 364-n. The sequence of one or more SGML tags is followed by an SGML Tag List End 365, followed by a DTD Map End 366. Each SGML Tag 364-1 through 364-n includes an SGML Tag Begin 367, an SGML Tag Name 368, followed by an SGML Tag Empty State 369, followed by an SGML Tag Assignment Type 370, followed by an HTML Tag List 371, followed by an SGML Tag End 372.

[0075] Each HTML Tag List 371 is delimited by an HTML Tag List Begin 373 at the beginning and an HTML Tag List End 375 at the end with the list including at least one HTML Tag 374-1 through HTML Tag 374-m following the HTML Tag List Begin 373.

[0076] Each HTML Tag 374-1 through 374-m is delimited by an HTML Tag Begin 376 at the beginning and an HTML Tag End 380 at the end. Following the HTML Tag Begin 376 is an HTML Tag Name 377, followed by an HTML Tag Empty State 378, followed by an HTML Attribute List 379, followed by a delimiter HTML Tag End 380.

[0077] Each HTML Attribute List 379 is delimited by an HTML Attribute List Begin 381 at the beginning and an HTML Attribute List End 383 at the end. Following the delimiter HTML Attribute List Begin 381 is at least one HTML Attribute 382-1 through an HTML Attribute 382-P, followed by an ending delimiter HTML Attribute List End 383.

[0078] Each HTML Attribute 382-1 through 382-p is delimited by an HTML Attribute Begin 384 at the beginning and an HTML Attribute End 389 at the end. Following the HTML Attribute Begin 384 is an HTML Attribute Name 385, followed by an HTML Attribute Source Type 386, followed by an HTML Attribute Source 1 387, followed by an HTML

Attribute Source 2 388, if one exists. The delimiter HTML Attribute End 389 terminates the listing of contents.

[0079] Fig. 8A shows major class dependencies 424 for the DTD Map object. For purposes of explanation of the figures that follow, arrows show class dependencies, meaning that an object having an arrow pointing to it is contained within the object originating the arrow. A Map object 400 includes a pointer to an object DTDMap 402. A pointer is a value that represents an absolute address of an item in computer memory. A pointer to an object is used to access the information stored for the implementation of a particular object by, minimally, referencing the pointer name and the field or function name within the object. Viewing Fig. 7 and Fig. 8A together, the DTDMap 402 of Fig. 8A, corresponding to the DTD Map 361 of Fig. 7, includes, via pointers, an SGMLTagList 404 corresponding to the SGML Tag 364-1 through 364-n of Fig. 7. The SGMLTagList 404 of Fig. 8A includes, via pointers, a class SGMLTag 406 corresponding to each of the SGML Tags 364-1 through 364-n of Fig. 7. The SGMLTag class 406 of Fig. 8A includes, via pointers, HTMLTagList 408, which corresponds to the HTML Tag List 371 of Fig. 7. The HTMLTagList 408 of Fig. 8A includes, via pointers, an HTMLTag 410 which corresponds to each of the HTML Tag 374-1 through HTML Tag 374-m of Fig. 7. The HTMLTag 410 of Fig. 8A includes, via pointers, an HTMLAttrList 412 class which corresponds to the HTML Attribute List 379 of Fig. 7. The HTMLAttrList 412 of Fig. 8A includes, via pointers, an HTMLAttr 414 class which corresponds to the HTML Attributes 382-1 through 382-p of Fig. 7. The HTMLAttr 414 class of Fig. 8A includes, via pointers, a class derived from an abstract class, denoted by an 'A' inside a triangle, HTMLAttrSource 416 which corresponds to the HTML Attribute Source Type 386 of Fig. 7. An abstract class is typically defined as a model to be used for defining other closely related classes which may, for example, need to exhibit similar behavior in a system. By defining an abstract class, the other classes are defined as inheriting the structure and methods of the parent abstract class. The hollow arrows of Fig. 8A denote inheritance of classes. HTML attributes may be obtained from different sources. Therefore, a UserInput 418 class is shown to inherit from the abstract class HTMLAttrSource 416. Also, an SGMLContent 422 class inherits from HTMLAttrSource 416, as does an SGMLTagAttr 420 class.

[0080] Fig. 8B shows major classes within the Map Module 276, illustrating the major dependencies among the classes. The classes 424 illustrated inside the dashed line rectangle are the classes 424 of Fig. 8A. A class MapService 452 includes a class Transformer 450, a class DTDMapTransformerService 456, a class SrcSGMLTagAttr 462, a class SrcSGMLContent 464, a class Map 400, a class MapEdit 460, and a class MapCreateEditService 454. The class MapEdit 460 includes the class DTDMapEdit 466. The class DTDMapEdit 466 has dependencies with the class DTDMap 402, the class SGMLTagList 404, SGMLTag 405, HTMLTagList 408, HTMLAttrList 412, HTMLTag 410, HTMLAttr 414, SGMLTagAttr 420, SGMLContent 422, and UserInput 418. The class HTMLAttrSource 416 has dependencies with the class SrcSGMLTagAttr 462 and the class SrcSGMLContent 464. The class DTDMapTransformerService 456 has dependencies with the class SrcSGMLTagAttr 462 and the class SrcSGMLContent 464. The class SrcSGMLTagAttr 462 has a dependency with the class SGMLTagAttr 420 and the class SrcSGMLContent 464 has a dependency with SGMLContent 422. The class MapCreateEditService 454 has dependencies with the class DTDMapEdit 466 and the class MapEdit 460. The class Map 400 contains DTDMap 402, DTDMapTransformerService 456, SrcSGMLTagAttr 462, and the class SrcSGMLContent 464. The functionalities of these classes and their objects are explained with regard to Figs. 18A(1)-18C(3).

[0081] Data items and objects in software generally involve dynamic allocation of computer storage resources at some stage in a request for execution of program code. Pointer variables, containing addresses of data items, methods, or objects, are available to be passed among objects during execution of code. As objects and data items are constructed and destructed dynamically, an object using a pointer or reference to a data item, for example, may reference the item after it has been destructed, possibly causing a system failure. A facility for registering objects and data items as they are created and requested gives objects a means to verify the current existence and usage of objects and data items before reference. A destructor verifies the current usage of an object or data item before destruction so that other objects using the object or data item may successfully complete their usage before destruction. An exemplary use of registering objects and data items is assignment of attribute values to HTML attributes, as discussed below with regard to Figs. 8C(1)-8C(4) and Figs. 18A(1)-18C(3).

[0082] Fig. 8C(1) illustrates a class structure for a SrcSGMLTagAttr 462 class of the SGML to HTML mapping and transformation. Fig. 8C(2) illustrates a class structure for a SrcSGMLContent 464 class of the SGML to HTML mapping and transformation. Fig. 8C(3) illustrates a class structure for the MapService 452 class of the SGML to HTML mapping and transformation. As described previously with regard to Fig. 8B, the class HTMLAttrSource 416 includes references to the class SrcSGMLTagAttr 462 and the class SrcSGMLContent 464. The HTMLAttrSource 416 contains an AtrSrcSGMLTagAttr, which is a reference to a SrcSGMLTagAttr 462, and an AtrSrcSGMLContent, which is a reference to a SrcSGMLContent 464. SrcSGMLTagAttr 462 contains a method registerSGMLTagNameAndAttributeName, used for registering SGML tag attributes in the MapService 452, so that attributes that have already been registered are available to be unregistered from the HTMLAttrSource 416 by using a method unregisterTagAttrKeyAndMapEntry in SrcSGMLTagAttr 462 through a virtual function unregisterFromSourceMap. SrcSGMLTagAttr 462 also contains a method setValueForAttributeOfTag to be used at document instance processing time to transform an SGML document to an SGML document.

[0083] SrcSGMLContent 464 contains a method registerSGMLTagName, used for registering SGML tag content in the SrcSGMLContent 464, so that attributes that have already been registered are available to be unregistered from the HTMLAttrSource 416 by using a method unregisterSGMLTagName in SrcSGMLContent 464. SrcSGMLContent 464 also contains a method setValueForTag to be used at document instance processing time to transform an SGML document to an HTML document.

[0084] Fig. 8C(4) illustrates a class structure for the MapCreateEditService 454 class of the SGML to HTML mapping and transformation. MapCreateEditService 454 was discussed previously with regard to Fig. 8B, and functionalities of the class and object are explained with regard to Figs. 18A(1)-18C(3). A method setSelectedSGMLTagToBeNullAssigned in MapCreateEditService 454 sets a selected SGML tag to be mapped to a null value. A method setSelectedSGMLTagToBeNotAssigned in MapCreateEditService 454 sets a selected SGML tag to be kept in the mapping. A method getAttributeAssignmentInformationForHTMLAttribute in MapCreateEditService 454 gets assignment information for assigned values to HTML attributes. Methods assignHTMLAttributeWithSGMLAttribute, assignHTMLAttributeWithSGMLContent, assignHTMLAttributeWithSystem, assignHTMLAttributeWithNoValue, assignHTMLAttributeWithUserInput in MapCreateEditService 454 assign values to the HTML attributes.

[0085] Fig. 9 illustrates a hierarchical view of major modules for implementation of the GUI for the SGML to HTML mapping and transformation. An Application Window 470 initiates execution of a Menu Manager 472. The Menu Manager 472 initiates execution of a File Service 474, Editor for Map 476, View 478, Map 480 and Message Dialog Service 482. The Map 480 module corresponds to the Map class 400 of Fig. 8A. The View 478 module is included in the GUI 270 of Fig. 4. The functionalities of these modules are explained with regard to Figs. 18A(1)-18C(3).

[0086] Fig. 10 shows an exemplary computer screen output of a Main Application Window 510. The window includes a title bar 512, a menu bar 514, a tool bar 516, and a viewing window work space 518. In order for a user to perform any menu operation, the user must select one of the items in the menu bar 514. When the user makes a selection, a sub-menu (or pull-down menu) displays all operations available for selection from the main menu. For example, if the user selected the File option, a sub-menu appears to display the options Open SGML, Open DTD, Open Map, Save Map, Map Save As, Save HTML, HTML Save As, Close SGML, and Exit. The Open SGML option allows the user to select an SGML document to open. The Open DTD option allows the user to select a DTD to open. The Open Map option allows the user to select a map to open. The Save Map Option allows the user to save a map onto disk. The Map Save As option allows the user to save the map onto the disk with the option of selecting a new file name for the saved map. The Save HTML option allows the user to save an HTML document onto disk. The HTML Save As option allows the user to save an HTML document onto disk with the option of selecting a new file name for the HTML file. The Close SGML option allows the user to close an SGML document and Exit option allows the user to exit the Application Window processing of the conversion.

[0087] If the user selects an Edit option from the menu bar 514, a sub-menu appears to display options Create Map and Edit Map. The Create Map option allows the user to create a map that can be used, to transform an SGML document to an HTML document. The Edit Map option allows the user to modify an existing map. A sub-menu for a View option displays the options to View SGML and to View HTML. These options are designed to display an SGML document or an HTML document when selected by a user. A View SGML option allows the user to display an SGML document in the work space of the main application window. A View HTML option allows the user to display an HTML document in the work space of the main application window. If the user selects the Map option, a sub-menu appears to display an option Run Map. Selecting Run Map initiates the transformation to transform an input SGML document to an HTML document. If the user selects the Option button, a sub-menu appears to display options of Incremental and Reference. An Incremental option allows the user to perform an incremental mapping of an SGML document to an HTML document. After an SGML tag is mapped into HTML tags the SGML document is then transformed into its corresponding HTML document. This occurs after each SGML tag is mapped. A Reference option allows the user to display the reference in transforming an SGML document to an HTML document.

[0088] Referring to Fig. 11, dialog boxes for opening an SGML file and saving an HTML file are shown. Fig. 11 shows an exemplary file open dialog box 600 which would be displayed responding to an Open SGML option from the sub-menu displayed after the user selected the File option in Fig. 10. A Filter text edit box 602 is displayed allowing the user to choose a type of file to be opened. Candidate directories for files are displayed in a Directories list box 604. Candidate files for opening will be displayed in a Files list box 606. A Selection text edit box 608 displays the file name that the user selects for opening. An OK button 610 allows the user to approve the selection shown in the Selection text edit box 608. A Filter button 612 allows the user to request a display of all files of a given type, in the Files list box 606, as described in the Filter text edit box 602. A Cancel button 614 allows the user to choose termination and cancellation of the current request to open a file.

[0089] Fig. 11 also shows an exemplary Save HTML file dialog box 616 that is displayed as a result of the user selecting the Save HTML option from the sub-menu displayed after the user selected the File button in Fig. 10. A Filter text edit box 618 allows the user to select a type of file for saving the current HTML file. Candidate directories for files are displayed in a Directories list box 620. Candidate files of a given type are displayed in a Files list box 622. A Selection

text edit box 624 allows the user to select a file name for saving the file. An OK button 626 allows the user to approve an operation and complete the operation of saving a file. A Filter button 628 allows the user to request a display of all files of a given type, in the Files list box 622, as described in the Filter text edit box 618. A Cancel button 630 allows the user to terminate and cancel the current request to save the current HTML file.

[0090] Fig. 12A and Fig. 12B display exemplary Map Edit Option and Map Edit Dialog boxes utilized for creation and editing of a map. The user is allowed to create a new map or edit an already existing map. If the user selects the Edit button from the Main Application Window 510 of Fig. 10, and either the Edit Map or Create Map option is selected, the Map Edit Option 690 dialog box of Fig. 12A is displayed to allow the user to select whether the Default mapping of the SGML document or a Conditional mapping of the SGML document should be used to create or edit the map. The Default mapping, selected by clicking on a Default button 692, is the user defined tag mapping set up by the user interaction with the SGML to HTML Map Edit dialog box 700 of Fig. 12B. The Conditional mapping, selected by clicking on a Conditional button 694, involves defining the conditional or special mappings. After the user selects one of the options from the Map Edit Option 690, then the Map Edit dialog box 700 of Fig. 12B is displayed to allow the user to interact with the system in defining a map. If the Create Map option is selected, the user is allowed to create a new map. Both the Map Edit Option dialog box 690 and the Map Edit dialog box 700 are used for creating a map and for editing an existing map.

[0091] Referring to Fig. 12B, a display of the Map Edit dialog box 700 shows a display of a list of SGML Tags 702, the current HTML Tag list 704 that an SGML tag selected from the SGML Tag list 702 maps to, and a list of Legal HTML Tags 706 that can be added into the current HTML Tag list 704. For a given SGML Tag 702, the user selects the Legal HTML Tag 706 by double clicking a mouse on an HTML tag from the Legal HTML Tag list 706. This will add the HTML Tag to the Current HTML Tag list 704. If the Current HTML Tag list 704 contains a list of HTML Tags that the SGML Tag 702 maps into, a new HTML Tag will be added below the HTML Tag that is selected in the Current HTML Tag 704 list. If an HTML Tag is inserted into the current HTML Tag 704 list, the HTML Tag(s) following the inserted Tag must be deleted, as they may no longer be legal. A Clear HTML Tag 708 button will clear the current HTML Tag 704 list. A Delete HTML Tag 710 button will delete the HTML Tag selected in the Current HTML Tag 704 list. The HTML Tags following the deleted HTML Tag in the Current HTML Tag 704 list must be deleted since they may no longer be legal. An Undo 712 button will undo the last clear, delete or insert operation. These buttons are easily modified to a menu operation format by one skilled in the art of computing. A Map SGML Tag 714 button allows the user to map the SGML Tag 702 to the HTML Tag list in the Current HTML Tag 704 list and then allows the user to select the next SGML tag to map. If a Done 716 button is selected, the remaining SGML Tags 702 will not be mapped. If a Cancel 718 button is selected, all previous SGML to HTML map information will be disregarded. Two possible selections in the Legal HTML Tag list 706 are Null Assigned and Not Assigned. Null Assigned deletes the SGML Tag 702 so that the SGML tag 702 will not be mapped and will not be displayed after transformation. Not Assigned leaves the SGML Tag 702 as is, so that the SGML Tag 702 will not be mapped to HTML but will be displayed as is after transformation.

[0092] An explanation of tag attribute assignment is provided with regard to Figs. 18A(1)-18C(3).

[0093] Fig. 12C shows exemplary data in the tag list boxes of the Map Edit dialog box 700 previously discussed for Fig. 12B. An explanation of the processing of the data is provided with regard to Figs. 18A(1)-18C(3).

[0094] Fig. 13 illustrates an exemplary class diagram displaying relationships among the classes of the SGML to HTML mapping and transformation for the GUI. An Application Window 772 manages the handling of the display of the application window of the GUI. A Menu Manager 778 handles all the tasks and objects associated with menu operations. A File Service 782 handles open and save operations associated with files. An ntEntity 790 is a general system representation of an SGML document, an SGML DTD, an HTML document, or an HTML DTD. A Symbol Table 770 is the system representation of an input document after it has been processed by a Parser Service 774. A MessageDialogService 776 handles the output of messages to the system used. A View Document 786 class handles the display of SGML or HTML documents upon user request. A Map Service 780 handles the creation and editing, through a MapCreateEditService 792, of a Map 788, which is the system representation of the rules to be utilized in the transformation of an SGML document to an HTML document. A GUIEditorForMap 784 handles the GUI interface for the user to dynamically create or edit the Map 788.

[0095] Fig. 14 shows an object message diagram displaying the behavior of the system among objects of the classes for the startup of the system. The object diagram illustrates major software objects enclosed in cloud shapes. Object method calls are illustrated with an Arabic numeral preceding a colon, followed by the name of the object method. The numeric ordering illustrated by the Arabic numerals followed by colons illustrate a stepwise logical flow of execution, and a flow of object data and messages, through the diagram. For a more detailed description of object diagrams and design, see Booch, G., *Object-Oriented Analysis and Design with Applications*, *Second Edition*, Addison Wesley Longman, California, 1994, which is incorporated herein by reference.

[0096] An Application Window 772 is the object which generates the main application window of Fig. 10. It is the first object created when the system starts execution. It contains all the necessary information and functions to display the main application window of Fig. 10. An HTMLSymbolTable 800 is an object, which is the system representation of the

HTML DTD, created by the Application Window 772, in a call Create (new) 802 through ParserService 774. The HTML Symbol Table 800 exists throughout the lifetime of the system. A MenuManager 778 is an object created by the Application Window 772, in a call Create (new) 804, to handle all menu operations associated with the menu of the main Application Window 772. The Application Window 772 passes the HTMLSymbolTable 800 object it created to the MenuManager 778 so that the MenuManager 778 can pass it to any other objects which require it. The MenuManager 778 creates and manages all objects necessary carry to out menu operations.

[0097] A MessageDialogService 776 is an object created by the MenuManager 778, in a call Create (new) 806, to allow message dialog boxes to display any kind of message to the user from anywhere in the system. Exemplary messages are error messages, warnings or instructions to the user, or any other type of message required. The MenuManager 778 passes the MessageDialogService 776 to other objects which may need to display messages to the user.

[0098] A MapService 780 is an object created by the MenuManager 778, in a call Create (new) 808, to handle map related objects. The MenuManager 778 initiates a call MapServiceInit 810 to initialize the state of the newly created MapService 780. For this example, a map is an object which describes how the SGML document will be transformed into an HTML document. The MenuManager 778 passes the HTMLSymbolTable 800 and the MessageDialogService 776 to the MapService 780 so that it has information about the HTML DTD and can send messages to the user.

[0099] A MapCreateEditService 792 is an object created by the MapService 780, in a call Create (new) 812, to handle the creation of a map or the modification of an existing map. The MapService 780 passes the HTMLSymbolTable 800 to the MapCreateEditService 792 so that it has information about an HTML DTD. The MenuManager 778 receives the MapCreateEditService object 792 from Map Service 780, in a call getMapCreateEditServiceObject 814, so that it may create or edit a map at any time.

[0100] A FileService 782 is an object created by the MenuManager 778, in a call Create (new) 816, to handle the tasks of opening and saving a file. The file corresponds to an SGML document, an SGML DTD, a map, or an HTML document. The user requests actions for files by selecting the File button exemplified in the menu bar 514 in Fig. 10. The File Service 782 creates an Open or Save dialog box to allow the user to choose the file the user wants to open or save, as exemplified in Fig. 11. MenuManager 778 passes MessageDialogService 776 to File Service 782 so that it may display messages to the user. The MenuManager 778 also passes the MapCreateEditService 792 to the File Service 782.

[0101] A GUIEditorForMap 784 is an object created by the MenuManager 778, in a call Create (new) 818, to handle the task of allowing the user to create a map or modify an existing map through a dialog box, as exemplified in Figs. 12B-12C. The MenuManager 778 passes the MessageDialogService 776 to the GUIEditorForMap 784 for displaying messages to a user. The MenuManager 778 passes the MapCreateEditService 792 to GUIEditorForMap 784 to create or modify a map.

[0102] A View Document 786 is an object created by the MenuManager 778, in a call Create (new) 820, to handle the task of displaying an SGML or HTML document through a display window. The user requests document viewing by selecting the View button from the menu bar 514 exemplified in Fig. 10. The Menu Manager 778 passes the MessageDialogService 776 to the ViewDocument 786 so that it may display messages to the user.

[0103] Fig. 15 shows an object message diagram to display the dynamic relationships that exist among the objects of the invention when an SGML document is being opened for the first time. A User 830 requests, from an Application Window 772, opening an SGML document file using a call OpenSGML 840. This is accomplished by the user's selection of the File button in the menu bar 514 of Fig. 10, followed by selection of the Open SGML option in the resulting sub-menu. The Application Window 772 sends a call OpenSGML 842 to a MenuManager 778 to process the user request. The MenuManager 778 then sends a call OpenSGML 844 to a FileService 782. The FileService 782 initiates a call getFileName 846 to request a file name for the SGML document from the User 830. A User 830 response, a FileName 846, is returned to the FileService 782. The FileService 782 sends a request isFound 848, accompanied by a FileName 848, to an IOService 832 to determine, by a response YES 848, that the FileName 846 returned by the User 830 exists. The FileService 782 then sends a request isReadable 850, accompanied by a FileName 850, to the IOService 832 to determine, by a response YES 850, that the file is also readable by the system. The FileService 782 then sends a request getEntityObject 852 to IOService 832, accompanied by the FileName 852, so that IOService 832 may obtain and return an ntEntity 852, which is associated with an external entity such as an SGML document, and its corresponding DTD, requested by the User 830.

[0104] The MenuManager 778 sends a request getSGMLNtEntity 854 to the FileService 782 to receive the ntEntity 854 from the FileService 782. The MenuManager 778 then sends the ntEntity 856 to a Parser Service 774 with a call getSymbolTable 856. The Parser Service 774 then generates a SymbolTable 856 for the SGML document, checks that it is a valid symbol table for the SGML document, and returns the SymbolTable 856 to the MenuManager 778. The MenuManager 778 then sends the ntEntity 858 to a MapCreateEditService 792 using a call useThisSGMLDoc 858, and sends the SymbolTable 860 to the MapCreateEditService 792 using a call UseThisSymbolTable 860. The MapCreateEditService 792 then handles further processing of the requested document.

[0105] Fig. 16 shows an object message diagram to display the dynamic relationships that exist among the objects

of the invention as an SGML document as being opened, with an existing SGML document already opened. A User 830 requests, from an Application Window 772, opening an SGML document file by a call OpenSGML 890. This is accomplished by the user's selection of the File button in the menu bar 514 of Fig. 10, followed by selection of the Open SGML option in the resulting sub-menu. The Application Window 772 sends a call OpenSGML 892 to a MenuManager 778 to process the user request. The MenuManager 778 then sends a call closeHTMLWindow 894 to a ViewDocument 786 so that if there is an open window displaying an HTML document, it will be closed. If there is an HTML document that has not been saved, the MenuManager 778 sends a request SaveHTML 896 to a MessageDialogService 776. The MessageDialogService 776 then sends a request SaveHTML 898 to the User 830 as a message asking if the user wishes to save the currently displayed HTML file. A User 830 response of NO 898 is returned to the MessageDialogService 776, which then returns a NO 896 to the MenuManager 778. The MenuManager 778 then sends a call Destroy(delete) 900 to an HTMLntEntity 880, representing the HTML file. The MenuManager 778 then sends a call closeSGMLWindow 902 to the ViewDocument 786 for the ViewDocument 786 to process closing of the display window of an opened SGML document. The MenuManager 778 then sends a call Destroy(delete) 904 to an SGML ntEntity 882, representing an opened SGML file. The MenuManager then sends a call OpenSGML 906 to a FileService 782. The FileService 782 initiates a call getFileName 908 to request a file name for the SGML document from the User 830. The User 830 response, a FileName 908, is returned to the FileService 782. The FileService 782 sends a request isFound 910, accompanied by a FileName 910, to an IOService 832 to determine, via a response YES 910, that the FileName 908 returned by the User 830 exists. The FileService 782 then sends a request isReadable 912, accompanied by a FileName 912, to the IOService 832 to determine, by a response YES 912, that the file is also readable by the system. The FileService 782 then sends a request getEntityObject 914, accompanied by a FileName 914, to the IOService 832, so that the IOService 832 may obtain and return an ntEntity 914. The ntEntity 906 is then returned to the MenuManager 778 in response to the request OpenSGML 906.

[0106] The MenuManager 778 then sends an ntEntity 916 to a Parser Service 774 with a call getSymbolTable 916. The Parser Service 774 then generates a SymbolTable 916 for the SGML document and DTD and returns the SymbolTable 916 to the MenuManager 778. The MenuManager 778 then sends an ntEntity 918 and a Map 918 to a MapService 780 using a call areThisMapAndTheSGMLDocConsistent 918, to determine if the existing Map 918 can be used with the new SGML document, ntEntity 918. A response NO 918 is returned to the MenuManager 778 if the Map 918 cannot be used. The MenuManager 778 then sends a request isMapSaved 920 to the MapService 780, to receive a response of YES 920, if the Map 918 is saved. The MenuManager 778 then sends a call destroy(erase & delete) 922 to an SGMLSymbolTable 884, to destroy the system's current SGML file represented in the format of a symbol table. The MenuManager 778 then sends a call Destroy (delete) 924 to a Map 788 to destroy the system's current map for a previous SGML document. The MenuManager 778 then sends a call resetMap 926 to the MapService 780 to initialize a new map for the new SGML document to be processed. The MenuManager 778 then sends a call useThisSGMLDoc 928 to a MapCreateEditService 792, sending an ntEntity 928 obtained from IOService 832. The MenuManager 778 then sends a call UseThisSymbolTable 930, along with a SymbolTable 930 obtained from ParserService 774, to the MapCreateEditService 792. The MapCreateEditService 792 then handles further processing of the requested SGML document.

[0107] Fig. 17 illustrates, in a class diagram format, the design of the Map Editor GUI. The Application Window 772, the MenuManager 778, the GUIEditorForMap 784, the SymbolTable 770, and the MapCreateEditService 792 have been described previously with regard to Figs. 13-15. In Fig. 17, a MapEditDialog 950 is contained by the GUIEditorForMap 784. The MapEditDialog 950 manages the handling of the display of the Map Edit dialog box. This includes determining what type of operation or requests the user can perform at a given point in the mapping of an SGML element. A MapTag 952 interacts with MapCreateEditService 792 and SymbolTable 770 to handle tasks and objects associated with mapping an SGML element to an HTML element. An AssignAttribute 954 interacts with MapCreateEditService 792 and SymbolTable 770 to handle tasks and objects associated with assigning values to the attributes of HTML elements.

[0108] Figs. 18A(1)-18C(3) are object message diagrams showing the behavior of the SGML to HTML mapping for assigning values to HTML attributes and for mapping SGML tags to HTML tags. As the object diagram of Figs. 18A(1)-18C(3) was large, it was divided over a plurality of drawing sheets. However, these drawing sheets, when taken together, constitute one object diagram. The MapEditDialog 950, the MapTag 952, the AssignAttribute 954, and the MapCreateEditService 792 have been described previously with regard to Fig. 17. The GUIEditorForMap 784 has been described previously with regard to Fig. 13. The HTMLSymbolTable 800 has been described previously with regard to Fig. 14. The SGMLSymbolTable 884 has been described previously with regard to Fig. 16.

[0109] As an exemplary manner of operating, the GUIEditorForMap 784 of Fig. 18A(1) creates the Map Edit dialog box of Fig. 12B by a function call EditMap 960 to call a constructor of the MapEditDialog 950.

[0110] To fill the SGML Tag list box 702 of Fig. 12B, MapEditDialog 950 gets a first SGMLTag 962, with a YES message 962, from MapTag 952 through a function getNextSGMLTag 962, and displays the SGML tag in the SGML Tag list box 702 of Fig. 12B. MapTag 952 gets a first SGMLTag 964 of Fig. 18A(2), with a YES message 964, from the SGML-

SymbolTable 884 through a function call getFirstElement 964. An alternative GUI design is to keep a list of all SGML tags already processed in the display.

[0111] The user selects an SGML tag to map by double clicking the mouse on the SGML tag in the SGML Tag list box 702 of Fig. 12B and the SGMLTag 966 of Fig. 18A(1) is sent to MapTag 952 through a function call selectedSGMLTag 966 of Fig. 18A(1). MapTag 952 of Fig. 18A(2) sends the SGMLTag 967 to MapCreateEditService 792 of Fig. 18A(2) through a function call selectedSGMLTag 967 of Fig. 18A(2).

[0112] To fill the Current HTML Tag list box 704 of Fig. 12B, the MapEditDialog 950 box of Fig. 18A(1) then gets an HTMLTag 968, with a YES message 968, to which the selected SGML tag maps, from MapTag 952 through a function call getNextCurrentHTMLTag 968. The function will get one HTMLTag 968 at a time and display it in the Current HTML Tag list box 704 of Fig. 12B. MapTag 952 of Fig. 18A(2) gets a next HTMLTag 970, along with a YES message 970, from the MapCreateEditService 792 using a function call getExistingHTMLTagsinMap 970. If an SGML tag is being assigned for the first time, NOT ASSIGNED will be displayed in the Current HTML Tag list box 704 of Fig. 12B.

[0113] To fill the Legal HTML Tag list box 706 of Fig. 12B, the MapEditDialog 950 box of Fig. 18A(1) sends a last HTML tag 972 in the Current HTML Tag list box 704 of Fig. 12B to MapTag 952 through a function call setLastCurrentHTMLTagInList 972. Then the MapEditDialog 950 box gets legal HTMLTags 974, along with a YES message 974, through a function call getLegalHTMLTag 974 from MapTag 952. The function will get one HTMLTag 974 at a time and display it in the Legal HTML Tag list box 706 of Fig. 12B. The legal HTML tags are those which can follow the last HTML tag in the current HTML Tag list box 704 of Fig. 12B. In Fig. 18A(2), the MapTag 952 gets legal HTMLTags 976, along with a YES message 976, one at a time, from HTMLSymbolTable 800, through a function call getFirstLegalElement 976, or MapTag 952 will get legal HTMLTags 978, along with a YES message 978, from HTMLSymbolTable 800 through a function call GetNextLegalElement 978.

[0114] To add an HTML tag from the Legal HTML Tag list box 706 of Fig. 12B to the Current HTML Tag list box 704 of Fig. 12B, the user selects an HTML tag from the Legal HTML Tag list box 706 of Fig. 12B by double clicking the mouse on the HTML tag. The HTML tag is added to the Current HTML Tag list box 704. The selected HTMLTag 980 of Fig. 18A(1) is sent to MapTag 952 through a function call addSelectedHTMLTag 980. In Fig. 18A(2), MapTag 952 sends an HTMLTag 982 to MapCreateEditService 792 through a function call selectedHTMLTag 982. MapTag 952 informs MapCreateEditService 792 that the HTMLTag 982 must be added to the current map through a function call addSelectedHTMLTagToCurrentMappingList 984. MapEditDialog 950 of Fig. 18A(1) sets an HTML tag 986 as the last tag in the list by a function call setLastCurrentHTMLTagInList 986 to MapTag 952.

[0115] In order to update the list of Legal HTML Tags 706 of Fig 12B, MapEditDialog 950 of Fig. 18A(1) obtains an HTMLTag 988, with a YES message 988 if there exists a legal tag, from MapTag 952 using a function call getLegalHTMLTag 988. MapTag 952 then obtains an HTMLTag 990, with a YES message 990, from HTMLSymbolTable 800 using a function call getFirstLegalElement 990, if it is the first legal element requested, or MapTag 952 obtains an HTMLTag 992, with a YES message 992, from HTMLSymbolTable 800 using a function call getNextLegalElement 992, if it is not the first legal element requested.

[0116] To fill the HTML Tag Attribute list box 720 of Fig. 12B, MapEditDialog 950 of Fig. 18A(1) sends an HTMLTag 994 in the Current HTML Tag list box 704 of Fig. 12B to AssignAttribute 954 through a function call selectedHTMLTag 994. Then the MapEditDialog 950 gets attributes of the HTML tag, HTMLAttribute 996, along with a YES message 996, from AssignAttribute 954 through a function call getNextHTMLAttribute 996. The function getNextHTMLAttribute 996 gets one HTMLAttribute 996 at a time and displays them in the HTML Tag Attribute list box 720 of Fig. 12B. AssignAttribute 954 of Fig. 18A(3) gets, one at a time, an HTMLAttribute 998, along with a YES message 998, of the HTMLTag 998 from HTMLSymbolTable 800 through a function call getFirstAttributeOfElement 998. If it is not the first attribute, AssignAttribute 954 gets an HTMLAttribute 1002 of the HTMLTag 998, along with a YES message 1002, from HTMLSymbolTable 800 through a function call getNextAttribute 1002. For each attribute AssignAttribute 954 gets from HTMLSymbolTable 800, and returns to MapEditDialog 950 of Fig. 18A(1), MapEditDialog 950 checks with AssignAttribute 954 to see if the attribute is of the required type through a function call IsCurrentAttributeOfRequiredType 1000, with a NO message 1000 indicating it is not, to be obtained from HTMLSymbolTable 800. AssignAttribute 954 of Fig. 18A(3) then checks with HTMLSymbolTable 800 to see if the attribute is of the required type through a function call IsCurrentAttributeOfRequiredType 1001, with a NO message 1001 indicating it is not.

[0117] To select an attribute to assign it a value, the user selects an attribute in the HTML Tag Attribute list box 720 of Fig. 12B by double clicking the mouse on the attribute. The MapEditDialog box 950 of Fig. 18A(1) sends a selected HTMLAttribute 1004 to AssignAttribute 954 through a function call selectedHTMLAttribute 1004. Depending upon the attribute type to which the HTMLAttribute 1006 is assigned, the MapEditDialog box 950 of Fig. 18A(1) will take different actions. For example, AssignAttribute 954 of Fig. 18A(3) gets an SGMLAttribute type 1006, an SGMLTag 1006, and an SGMLAttribute 1006 by sending an HTMLAttribute 1006 to MapCreateEditService 792 using a function call getAttributeAssignmentInformationForHTMLAttribute 1006. The MapEditDialog 950 box of Fig. 18A(1) gets an SGMLAttribute type 1008 for the selected HTML Attribute 1004 from AssignAttribute 954 through a function call getAttributeType 1008. The SGML Attribute radio button 726 of Fig. 12B is displayed depressed. The MapEditDialog 950 of Fig. 18A(1)gets a

source SGMLTag 1010 and a source SGMLAttribute 1010 assigned to the HTML Attribute 1004 through a function call getSourceSGMLTagAndAttribute 1010 and displays them in the Source SGML Tag list box 722 of Fig. 12B, and the SGML Tag Attribute list box 724 of Fig. 12B. Next, the MapEditDialog 950 gets a source SGMLTag 1012, one SGMLTag 1012 at a time, which can be assigned to the HTML attribute 1004, along with a YES message 1012, from AssignAttribute 954 through a function call getNextSourceSGMLTag 1012, and displays them in the Source SGML Tag list box 722 of Fig. 12B. AssignAttribute 954 of Fig. 18A(3) gets source SGML Tags 1014, with a YES message 1014 if getting the first element, from SGMLSymbolTable 884 through a function call GetFirstElement 1014, or, one by one, gets a source SGML Tag 1018, with a YES message 1018, through a call GetNextElement 1018 if it is not the first element. AssignAttribute 954 verifies that the source SGML Tag 1014 has an attribute by checking with the SGMLSymbolTable 884 through a function call elementHasAttribute 1016 to obtain a YES 1016 response, for the first element if it has attributes. If it is not the first element, AssignAttribute 954 verifies that the source SGML Tags 1018 have attributes by checking with the SGMLSymbolTable 884 through a function call elementHasAttribute 1020 to obtain a YES 1020 response. The MapEditDialog 950 of Fig. 18A(1) gets all attributes SGMLAttribute 1022, one SGMLAttribute 1022 at a time, with a YES message 1022, of the previously assigned source SGML tag from AssignAttribute 954 through a function call getNextSourceSGMLAttribute 1022 and displays the attributes in the source SGML Tag Attribute list box 724 in Fig. 12B. AssignAttribute 954 of Fig. 18A(3) gets SGML attributes 1024, and a YES message 1024, from SGMLSymbolTable 884 by sending an SGML tag 1024 through a function call getFirstAttributeOfElemont 1024 if it is the first attribute of the element, or it gets SGMLAttributes 1026, one SGMLAttribute 1026 at a time, along with a YES message 1026, from SGMLSymbolTable 884 through a function call getNextAttribute 1026. The user selects an SGML tag from the Source SGML Tag list box 722 of Fig. 12B by double clicking the mouse on the SGML tag. The current implementation supports selection by double clicking the mouse on the selection. However, any alternative selection technique can be used, such as highlighting the selection and pressing the Enter or Return key. MapEditDialog 950 of Fig. 18C(1) sends a selected source SGMLTag 1110 to AssignAttribute 954 through a function call selectedSourceSGMLTag 1110. Then MapEditDialog 950 of Fig. 18A(1) gets all SGMLAttributes 1022 of the selected source SGMLTag 1110, one SGMLAttribute 1022 at a time, along with a YES message 1022, from AssignAttribute 954 through a function call getNextSourceSGMLAttribute 1022 and displays them in the SGML Tag Attribute list box 724 of Fig. 12B. AssignAttribute 954 of Fig. 18A(3) gets the SGMLAttributes 1024, along with a YES message 1024, from SGMLSymbolTable 884 through a function call getFirstAttributeOfElement 1024 if it is requesting the first attribute, or AssignAttribute 954 gets the SGMLAttributes 1026, one SGMLAttribute 1026 at a time, along with a YES message 1026, from SGMLSymbolTable 884 through a function call getNextAttribute 1026 if it is not the first attribute.

[0118] The user selects an SGML attribute from the source SGML Attribute list box 724 of Fig. 12B by double clicking the mouse on the SGML attribute. MapEditDialog 950 of Fig. 18C(1) sends a selected source SGMLAttribute 1112 to AssignAttribute 954 through a function call selectedSourceSGMLAttribute 1112. The user selects the Assign button 736 of Fig. 12B. MapEditDialog 950 lets AssignAttribute 954 know that an SGML attribute was assigned to the HTML attribute through a function call assignedSGMLAttribute 1114 of Fig. 18C(1). AssignAttribute 954 of Fig. 18C(3) sends, for the HTML attribute that is being assigned, an HTMLAttribute 1116, a source SGMLTag 1116, and the source SGMLAttribute 1116 to MapCreateEditService 792 through a function call assignHTMLAttributeWithSGMLAttribute 1116.

[0119] If SGML content was assigned to the HTML attribute, then the SGML Content radio button 728 of Fig. 12B is depressed. The MapEditDialog box 950 of Fig. 18B(1) gets a content SGMLTag 1048 assigned to the HTML attribute from AssignAttribute 954 through a function call getContentSGMLTag 1048, along with a YES message 1048, and displays it in the Source SGML Tag list box 722 of Fig. 12B. The MapEditDialog box 950 gets all the content SGMLTags 1052, one SGMLTag 1052 at a time, which can be assigned to the HTML attribute, along with a YES message 1052, from AssignAttribute 954 through a function call getNextContentSGMLTag 1052 and displays them in the Source SGML Tag list box 722 of Fig. 12B. AssignAttribute 954 of Fig. 18B(3) gets content SGMLTags 1054, along with a YES message 1054, from SGMLSymbolTable 884 through a function call getFirstPrimitiveElement 1054 if it is the first SGML tag requested, or AssignAttribute 954 gets content SGMLTags 1056, one SGMLTag 1056 at a time, along with a YES message 1056, from SGMLSymbolTable 884 through a function call getNextPrimitiveElement 1056.

[0120] The user selects the content SGML tag from the Source SGML Tag list box 722 of Fig. 12B by double clicking the mouse on the SGML tag. MapEditDialog 950 of Fig. 18B(1) sends the selected content SGMLTag 1058 to AssignAttribute 954 through a function call selectedContentSGMLTag 1058.

[0121] The user selects the Assign button 736 of Fig. 12B. MapEditDialog 950 of Fig. 18B(1) lets AssignAttribute 954 know that an SGML content was assigned to the HTML attribute through a function call assignSGMLContent 1060. AssignAttribute 954 of Fig. 18B(3) sends an HTMLAttribute 1062 that is being assigned the SGML content and a content SGMLTag 1062 to MapCreateEditService 792 through a function call assignHTMLAttributeWithSGMLContent 1062.

[0122] If System was assigned to the HTML attribute, the System radio button 730 of Fig. 12B is depressed. MapEditDialog 950 takes no further action. The user selects the Assign button 736 of Fig. 12B. MapEditDialog 950 of Fig. 18B(1) lets AssignAttribute 954 know that a system value was assigned to the HTMLAttribute through a function call

assignSystem 1049. AssignAttribute 954 of Fig. 18B(3) sends the HTMLAttribute 1051 that is being assigned a system value to MapCreateEditService 792 through a function call assignHTMLAttributeWithSystem 1051.

[0123] If No Value was assigned to the HTML attribute, the No Value radio button 732 is depressed. MapEditDialog 950 will take no further action. The user selects the Assign button 736 of Fig. 12B. MapEditDialog 950 of Fig. 18C(1) lets AssignAttribute 954 know that no value is assigned to the HTML attribute through a function call assignNoValue 1102. AssignAttribute 954 of Fig. 18C(3) sends the HTMLAttributes 1104 to be assigned no value to MapCreateEditService 792 through a function call assignHTMLAttributeWithNoValue 1104.

[0124] If User Input was assigned to the HTML attribute, the User Input radio button 734 is depressed. MapEditDialog 950 of Fig. 18C(1) gets a UserInput 1100 from AssignAttribute 954 through a function call getUserInputData 1100 and then displays the information in the User Input text edit box 731 of Fig. 12B. The user enters data into the text edit box 731 of Fig. 12B. MapEditDialog 950 of Fig. 18C(1) sends a UserInputData 1106 to AssignAttribute 954 through a function call selectedUserInputData 1106. The user then selects the Assign button 736 of Fig. 12B. MapEditDialog 950 of Fig. 18C(1) lets AssignAttribute 954 know that a user input is to be assigned to the HTML attribute through the function assignUserInput 1108. AssignAttribute 954 of Fig. 18C(3) sends an HTMLAttribute 1109 that is being assigned a user input value, and the UserInputData 1109, to MapCreateEditService 792 through a function call assignHTMLAttributeWithUserInput 1109.

[0125] For any of the HTML attribute source types assigned to the HTML attribute, the user has options to change the source type of the HTML attribute. For example, if the No Value radio button 732 of Fig. 12B is depressed for a selected HTML attribute, the user has an option to change the source type of the HTML attribute by pressing any of the other radio buttons such as the User Input radio button 734 of Fig. 12B. Depending upon which radio button the user selects, the user will need to enter more information before the HTML, attribute is assigned a value.

[0126] For the radio buttons SGML Attribute 726, SGML Content 728, and User Input 734, the user can change the input selection more than one time. For example, the user can change the user input in the text edit box 731 more than one time. As another example, the user can select one content SGML tag in the Source SGML Tag list box 722, and later decide to select another content SGML tag. The most recent value given by the user is the value assigned to the HTML attribute when the Assign button 736 is selected.

[0127] The user repeats selection of an attribute to assign it a value, selecting a radio button for an HTML attribute source type, assigning a value to the HTML attribute, and selecting the Assign button 736 of Fig. 12B until the user has assigned all the HTML attributes desired. The user then selects the Assign Done button 740 of Fig. 12B. MapEditDialog 950 of Fig. 18B(1) lets MapCreateEditService 792 know that it is done assigning values to the attributes through a function call assignDoneSelected 1064 to AssignAttribute 954, which sends a call assignDoneSelected 1066 to MapCreateEditService 792 of Fig. 18B(3).

[0128] The user repeats adding more HTML tags to the Current HTML Tag list box 704 of Fig.12B at will. At any time the user can clear the Current HTML Tag list box 704 or delete HTML tags from the Current HTML Tag list box 704 by selecting the Clear HTML button 708 or the Delete HTML button 710 of Fig. 12B. Once the user has completed mapping one SGML tag, the user selects the Map SGML Tag button 714. MapEditDialog 950 of Fig. 18B(1) informs MapTag 952 that the user has completed mapping the selected SGML tag through a function call doneMappingSGMLTag 1068. MapTag 952 of Fig. 18B(2) informs MapCreateEditService 792 that the user has completed mapping the selected SGML tag through the function finishOneMapping 1070. MapEditDialog 950 of Fig. 18B(1) requests a next SGMLTag 1072 for mapping from MapTag 952, along with a YES message 1072, using a function call getNextSGMLTag 1072 and displays the SGMLTag 1072 in the SGML Tag list box 702 of Fig. 12B. MapTag 952 of Fig. 18B(2) then obtains a next SGMLTag 1074, along with a YES message 1074, from SGMLSymbolTable 884 using a function call getNextElement 1074. MapTag 952 then returns the SGMLTag 1072 to MapEditDialog 950 of Fig. 18B(1) in response to the function call getNextSGMLTag 1072.

[0129] The user repeats processing the map for all the SGML tags the user wants to map. When the user is finished with the map editor, the user selects the Done button 716 of Fig. 12B. MapEditDialog 950 of Fig. 18B(1) informs MapTag 952 that it is done mapping through a function call doneMapping 1076. MapTag 952 of Fig. 18B(2) informs MapCreateEditService 792 that the mapping of the SGML tags is completed through a function call finishCreatingMap 1078.

[0130] Fig. 19 illustrates an exemplary hardware configuration upon which the invention may be implemented. A Workstation 1200 has component parts a Display Controller 1202, a Central Processing Unit ("CPU") 1204, a Random Access Memory ("RAM") 1206, a Read Only Memory ("ROM") 1208, an Input Controller 1210, connected to a Keyboard 1212 and a Mouse 1214, a System Bus 1220, a Hard Disk 1222 and a Floppy Drive 1224 connected to a Disk Controller 1226, a Comm Controller 1228 connected to a Network 1230, and an Input/Output ("I/O") Controller 1232 connected to a Hard Disk 1236 and a Printer 1234, and a Cathode Ray Tube ("CRT") 1238 connected to the Display Controller 1202. The System Bus 1220 connects the CPU 1204, the RAM 1206, the ROM 1208, the Input Controller 1210, the Disk Controller 1226, the Comm Controller 1228, the I/O Controller 1232, and the Display Controller 1202 for transmitting data over the connection line.

[0131] For example, the computer code generated for execution is loaded into the RAM 1206 for execution by the CPU 1204, using the System Bus 1220, with input files stored on the Hard Disk 1236, with other input coming from the Keyboard 1212 and the Mouse 1214 through the Input Controller 1210, and from the Hard Disk 1222 and the Floppy Drive 1224, through the Disk Controller 1226, onto the System Bus 1220. The System Bus 1220 interacts with the ROM 1208, the Network 1230, and the Comm Controller 1228. The GUI of the system can be displayed on the CRT 1238 through the Display Controller 1202, and on output to the Printer 1234 or to the Hard Disk 1236 through the I/O Controller 1232.

[0132] Other implementations of the map creator and editor for transforming a first structured information format to a second structured information format are possible using the procedures described previously for Figs. 1A-19. For example, variable names in a first database format may be mapped to variable names in a second database format. Another exemplary implementation is a mapping of public identifiers in an ISO/IEC 9070 format to file names in a UNIX file system format.

[0133] For other implementations, the same techniques described with regard to the SGML to HTML mapping and transformation are utilized, with structure of information defined differently from an SGML DTD. In general terms, a parser breaks down an input source file into source components and their structure, based upon a structure format specified for the input source file, for map creating and editing. The source components and their structure are presented to the user for interactive selection of components of the first structure, with candidate target components of the second structure presented to the user for selection of target components for the mapping of the source components for creation of rules for a transformation map. An exemplary implementation of a mapping of public identifiers in an ISO/IEC 9070 format to file names in a UNIX file system format is discussed below with regard to Figs. 20A-20H.

[0134] Fig. 20A illustrates a public identifier 1400 in ISO/IEC 9070 standard format. An owner name is made up of a registered owner name and an unregistered owner name. For this example, the owner name is 'XYZ'. The public identifier further has an object name, separated from the owner name by '//'. For this example, the object name is 'font::metric::x-offset::622'.

[0135] Mapping one system structure to another system structure involves transformation of strings in one allowable character set to strings of another allowable character set. For example, computer programming languages are defined as having an alphabet of acceptable characters for forming valid variable names. A first programming language may be defined as allowing the '-' (hyphen) character to be embedded in a valid variable name, but not the '_' (underscore) character. A second programming language may be defined as allowing the '_' character, but not the '-' character. A mapping of valid variable names of the first programming language to valid variable names of the second programming language would involve mapping occurrences of '-' to a different character, such as '_', which is acceptable in the second programming language environment.

[0136] In the example of mapping the ISO/IEC 9070 naming scheme to the UNIX file name scheme, the separator '//' is allowed in ISO/IEC 9070, but is not a valid character sequence in the UNIX file system naming conventions. A user wishing to map valid ISO/IEC 9070 names to valid UNIX file names needs to transform every occurrence of the separator '//' into a valid UNIX file name character string.

[0137] Fig. 20B illustrates an exemplary mapping of an ISO/IEC 9070 public identifier to a UNIX file name format. The exemplary mapping maps the structured public identifier to a flat UNIX file name. Lines 1420, 1422, 1424, and 1426 illustrate rules to map component name strings to identical strings in the UNIX format. Line 1428 illustrates a rule to map an ISO/IEC 9070 owner name component separator '::', which is not widely used, to the character '_', which is a valid UNIX character. Line 1430 illustrates a rule to map an ISO/IEC 9070 owner name, object name component separator '//', which is not a valid string in the UNIX file format, to '__' (two underscore characters).

[0138] Fig. 20C illustrates an exemplary UNIX file name 1440 resulting from mapping the exemplary ISO/IEC 9070 name of Fig. 20A through the mapping of Fig. 20B. Ownername 'XYZ' of line 1400 of Fig. 20A is mapped to 'XYZ' using the rule of line 1420 of Fig. 200. The '//' is mapped to '__' using the rule of line 1430 of Fig. 20B. The three substrings '::' in the object name of public identifier 1400 of Fig. 20A are each mapped to a character '_' using the rule 1428 of Fig. 20B.

[0139] Fig. 20D illustrates an exemplary user interface for mapping a public identifier 1502 of ISO/IEC 9070 to a UNIX file system format 1504. A map editor and creator 1500 maps names in the ISO/IEC 9070 convention to names in the UNIX file system convention. An exemplary mapping starts with system display 1502 of public identifier components and a display of valid UNIX file name components 1504. The public identifier components 1502 are registered owner name, unregistered owner name, and object name. A user selects one of these options. If owner name is selected, then the user is asked to select from prefix and owner-name components 1522 in Fig. 20E. User options presented are a create directory 1506, a map 1508, a merge all 1510, a next 1512, and a previous 1514. The create directory 1506 option allows the user to create a new directory name in the UNIX file system window 1504. The map option 1508 allows the user to request that a map be created at the time of request. The merge all 1510 option allows the user to create a UNIX file name 1504 by merging all the components of the public identifier name 1502 into a flat file name 1504. The next 1512 option allows the user to step to a next screen. The previous 1514 option allows the user to back up to the previous screen.

[0140] Fig. 20E illustrates an exemplary user interface 1520 for a registered owner 1522 component of the ISO/IEC 9070 public identifier 1502 of Fig. 20D. User interface 1520 options presented are a window 1522 showing a prefix and an owner-name component. User options are a map individually 1524, a merge both 1526, a next 1527, and a previous 1528. The map individually 1524 option allows the user to map individual components of the ISO/IEC 9070 name 1522 to individual components of the UNIX file system scheme 1504 of Fig. 20D. The merge both 1526 option allows the user to merge components of the registered owner name 1522 into one flat UNIX file name or directory name. The next 1527 option allows the user to step to a next screen. The previous 1528 option allows the user to back up to the previous screen.

[0141] Fig. 20F illustrates an exemplary user interface 1530 for mapping the prefix, owner name component separator to the UNIX legal character set format 1534. The registered owner component has a prefix and an owner-name component separator "::" 1532 which is not a widely used character string in the UNIX environment. The user is allowed to map the '::' separator 1532 to any of the valid characters in the UNIX file system character set 1534, with a mapping to '_' as a default mapping. User options are a map 1536, a next 1537, and a previous 1538. The map 1536 option allows the user to select creating a map, with the assumption that the user has finished selecting options for creation of the map. The next 1537 option allows the user to step to a next screen. The previous 1538 option allows the user to back up to the previous screen.

[0142] Fig. 20G illustrates an exemplary user interface 1540 for mapping an owner name character 1542 to valid characters 1544 of the UNIX file system format. The user is given the options of mapping special characters 1542 which are valid in the ISO/IEC 9070 scheme to characters which are valid in the UNIX file system scheme 1544. A mapping of a character in the ISO/IEC 9070 scheme 1542 is set to '_' as a default mapping. The user is given options of a map 1546, a next 1548, and a previous 1550. The map 1546 option allows the user to select creating a map, with the assumption that the user has finished selecting options for creation of the map. The next 1548 option allows the user to step to a next screen. The previous 1550 option allows the user to back up to the previous screen.

[0143] Fig. 20H illustrates an exemplary user interface 1560 for the user to map a registered owner component 1562 to a UNIX file scheme format 1564. The user is allowed to select a prefix component of the registered owner name or an owner name component other than prefix 1562. The user is allowed to select a directory option in the UNIX scheme 1564. The user is also allowed to select a file with object name option in the UNIX file scheme 1564. The user is given an option of a map 1566 for creating the map with the options currently selected. The user is also given an option of a previous 1568 to back up to the previous screen.

[0144] The present invention has been described using an exemplary implementation of a mapping creator and editor for an SGML to HTML transformer with user interaction for creation and editing of the map, and an exemplary mapping creator and editor for an ISO/IEC 9070 to a UNIX file format transformer. The example shown in this disclosure uses OOP and Windows GUI techniques to implement the user interface, map processing, and transformation. However, the user interface can be implemented using text line queries or menus. Programming methodologies other than OOP can be used for implementing the processing. References to storage areas can be made by techniques other than using pointers.

[0145] This invention may be conveniently implemented using a conventional general purpose digital computer or microprocessor programmed according to the teachings of the present specification, as will be apparent to those skilled in the computer art. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art. The invention may also be implemented by the preparation of application specific integrated circuits or by interconnecting an appropriate network of conventional component circuits, as will be readily apparent to those skilled in the art.

[0146] The present invention includes a computer program product which is a storage medium including instructions which can be used to program a computer to perform a process of the invention. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, CD-ROMs, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions.

[0147] Obviously, numerous modifications and variations of the present invention are possible in light of the above teachings. It is therefore to be understood that within the scope of the appended claims, the invention may be practiced otherwise than as specifically described herein.

Appendix A

```
<!doctype test [
<!element test - o (front, section+)
<!element front - o (title, author, keywords?)>
<!element title - o (#PCDATA)>
<!element author - o (fname, surname, title?)>
<!element (fname, surname) - o (#PCDATA)>
<!element keywords - o (#PCDATA)>
<!element section - o (number?, title?, para*, subsec1*)>
<!element subsec1 - o (number?, title?, para*, subsec2*)>
<!element subsec2 - o (number?, title?, para+)>
<!element number - o (#PCDATA)>
<!element para - o (#PCDATA)>
]>
```

Appendix B

30 Mapping		
32 Front	→	Null
34 Title	→	H3
36 Author	→	Null
38 Frame	→	P
40 Surname	→	P
42 Keywords	→	P
44 Section	→	Null
46 Number	→	P strong
48 Para	→	P
50 Subsec 1	→	Null
52 Subsec 2	→	Null

Appendix C

```
<test>
<front>
<title>
Test mapping
</title>
<author>
<fname>
Tester
</fname>
<surname>
Giver
</surname>
</author>
<keywords>
Mapping
</keywords>
</front>
<section>
<number>
1
</number>
<title>
First Major Section
</title>
<para>
The first major section para.
</para>
<subsec1>
<number>
1.1
</number>
<title>
Subsection 1.1
</title>
<para>
This is a para in the subsec1.1
</para>
</subsec1>
<subsec1>
  <number>
  1.2
  </number>
  <title>
  Subsection 1.2
  </title>
  <para>
  This is a subsection 1.2
  </para>
  </subsec1>
  </section>
  <section>
  <number>
  2
  </number>
```

```
<title>
Second Major Section
</title>
<para>
The second major section para.
</para>
<subsec1>
<number>
2.1
</number>
<title>
Subsection 2.1
</title>
<para>
This is a para in the subsec 2.1
</para>
</subsec1>
<subsec1>
<number>
2.2
</number>
<title>
Subsection 2.2
</title>
<para>
This is a subsection 2.2
</para>
</subsec1>
</section>
</test>
```

Appendix D

```
250  <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
252  <html>

254  <head>
256  <meta http-equiv="Content-Type"
258  content="text/html; charset=iso-8859-1">
260  <meta name="GENERATOR" content="Microsoft FrontPage 2.0">
262  <title>test</title>
264  </head>

266  <body bgcolor="#FFFFFF">

268  <h3>Test mapping</h3>

270  <p>Tester</p>

272  <p>Giver</p>

274  <p>Mapping</p>

276  <p><strong>1</strong></p>

278  <h3>First Major Section</h3>

280  <p>The first major section para.</p>

282  <p><strong>1.1</strong></p>

284  <h3>Subsection 1.1</h3>

286  <p>This is a para in the subsec1.1</p>

288  <p><strong>1.2</strong></p>

290  <h3>Subsection 1.2</h3>

292  <p>This is a subsection 1.2</p>

294  <p><strong>2</strong></p>

296  <h3>Second Major Section</h3>

298  <p>The second major section para.</p>

300  <p><strong>2.1</strong></p>

302  <h3>Subsection 2.1</h3>

304  <p>This is a para in the subsec 2.1</p>

306  <p><strong>2.2</strong></p>

308  <h3>Subsection 2.2</h3>

310  <p>This is a subsection 2.2</p>
312  </body>
314  </html>
```

Claims

1. An object-oriented system for processing structured information for implementation by a computer in an object-oriented framework, comprising:

   a storage means;
   a first obtaining means for obtaining an interactive input from a user;
   a second obtaining means for obtaining a first structural description of a first structured information format;
   a third obtaining means for obtaining a second structural description of a second structured information format;
   means for creating a rule to transform an element of the first structured information format into an element of the second structured information format utilizing the interactive input from the user, the first structural description, and the second structural description; and
   means for outputting the rule,
   wherein at least one of the first obtaining means, the second obtaining means, the third obtaining means, the means for creating, and the means for outputting includes a software object.

2. A system according to Claim 1, wherein the first structured information format includes an ISO/IEC 9070 public identifier naming format, the second structured information format includes an operating system file name format, and the means for creating comprises:

   means for creating a rule to transform an ISO/IEC 9070 public identifier naming format element into an operating system file name format element utilizing the interactive input from the user, a structural description of the ISO/IEC 9070 public identifier naming format, and a structural description of the operating system file name format.

3. A system according to Claim 1, wherein the first structured information format includes a first database variable name format, the second structured information format includes a second database variable name format, and the means for creating comprises:

   means for creating a rule to transform a first database variable name format element into a second database variable name format element utilizing the interactive input from the user, a structural description of the first database variable name format, and a structural description of the second database variable name format.

4. A system according to Claim 1, wherein the structured information includes a markup language format, the first structured information format includes a first markup language format, the second structured information format includes a second markup language format, and the means for creating comprises:

   means for creating a rule to transform a first markup language format element into a second markup language format element utilizing the interactive input from the user, a structural description of the first markup language format, and a structural description of the second markup language format.

5. A system according to Claim 4, wherein the first markup language format includes a Standard Generalized Markup Language ("SGML"), the second markup language format includes a HyperText Markup Language ("HTML"), and the means for creating comprises:

   means for creating a rule to transform an SGML element of the first markup language format into an HTML element of the second markup language format utilizing the interactive input from the user, the first structural description which includes an SGML Document Type Definition ("DTD"), and the second structural description which includes an HTML DTD.

6. A system according to Claim 1, wherein the means for creating comprises:

   a map creator object.

7. A system according to Claim 6, wherein the means for outputting the rule comprises:

   an object method for outputting the rule to a map object.

8. A system according to Claim 6, wherein the map creator object comprises:

a reference to a software object for an element for transformation of the first structured information format;
a reference to a software object for an element of the second structured information format, for transformation of the element of the first structured information format;
a reference to a software object for a property of the element of the second structured information format, for transformation of the element of the first structured information format;
a reference to a software object for an attribute value of the element of the second structured information format, for transformation of the element of the first structured information format;
an object method for obtaining the element for transformation of the first structured information format, which has been interactively selected by the user, using the software object for the element for transformation of the first structured information format;
an object method for obtaining the element of the second structured information format which corresponds to the element of the first structured information format, which has been interactively selected by the user, using the software object for the element of the second structured information format;
an object method for determining a property of the element of the second structured information format which has been selected by the user, using the software object for a property of the element of the second structured information format;
an object method for obtaining a second structured information format attribute value which has been interactively input by a user, using the software object for the attribute value of the element of the second structured information format; and
an object method for assigning the attribute value which has been interactively input by a user to the second structured information format attribute value.

9. A system according to Claim 8, wherein the map creator object further comprises:

a reference to a software object for registering an instance of an element for transformation of the first structured information format; and
a reference to a software object for unregistering the instance of an element for transformation of the first structured information format when the element is no longer needed by the map creator

10. A system according to Claim 1, further comprising:

means for processing an element of the first structured information format into a plurality of first structured information format components.

11. A system according to Claim 10, wherein the means for processing an element of the first structured information format into a plurality of first structured information format components comprises:

a parser object.

12. A system according to Claim 11, wherein the parser object comprises:

a reference to the element of the first structured information format;
a reference to a storage area for the plurality of first structured information format components; and
an object method for processing the element of the first structured information format into the plurality of first structured information format components for storage in the storage area for the plurality of first structured information format components.

13. A system according to Claim 12, further comprising:

means for utilizing the rule to transform the element of the first structured information format into the element of the second structured information format.

14. A system according to Claim 13, wherein the means for utilizing the rule to transform the element of the first structured information format into the element of the second structured information format comprises:

a transformer object.

15. A system according to Claim 1, wherein the means for obtaining an interactive input from a user comprises:

a user interface object.

16. A system according to Claim 15, wherein the user interface object comprises:

a reference to a software object for user input;
an object method for obtaining interactive input from the user of a selection of an element for transformation of a first structured information format using the software object for user input; and
an object method for obtaining interactive input from the user of a selection of an element of a second structured information format which corresponds to the element of the first structured information format using the software object for user input.

17. A system according to Claim 15, wherein the user interface object further comprises:

a reference to a software object for user input of a selection of a transformation to be performed on the element of the first structured information format; and
an object method for obtaining interactive input from the user of the selection of a transformation to be performed on the element of the first structured information format using the software object for user input of the selection of the transformation.

18. A system according to Claim 17, wherein the object method for creating a rule to transform an element of a first structured information format into an element of a second structured information format utilizing the interactive input from the user further comprises:

a reference to a software object for a rule to be created; and
an object method for creating a rule to map the second element of the first structured information format to a null string using the software object for the rule to be created, when the selection of a transformation which has been input by the user indicates a null transformation is to be performed.

19. A system according to Claim 17, wherein the object method for creating a rule to transform an element of a first structured information format into an element of a second structured information format utilizing the interactive input from the user further comprises:

a reference to a software object for a rule to be created; and
an object method for creating a rule to map the second element of the first structured information format to a copy of the second element of the first structured information format using the software object for the rule to be created, when the selection of a transformation which has been input by the user indicates a transformation of the second element of the first structured information format to a copy of the second element of the first structured information format is to be performed.

20. A system according to Claim 15, wherein the user interface object further comprises:

a reference to a software object for an interactive user input of a source for inputting the second structured information format attribute value;
a reference to a software object for an interactive user input of the second structured information format attribute value;
an object method for obtaining interactive input from the user of the source for inputting the second structured information format attribute value using the software object for the interactive user input of the source for inputting the second structured information format attribute value; and
an object method for obtaining interactive input from the user of the second structured information format attribute value using the software object for the interactive user input of the second structured information format attribute value.

21. A system according to Claim 20, further comprising:

a reference to a software object for a rule to be created;
an object method for examining the source which has been input by the user; and