(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 December 2005 (01.12.2005)

PCT

(10) International Publication Number
WO 2005/114409 A2

(51) International Patent Classification[7]: G06F 9/46

(21) International Application Number:
PCT/US2005/017603

(22) International Filing Date: 20 May 2005 (20.05.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/573,553 21 May 2004 (21.05.2004) US

(71) Applicant *(for all designated States except US)*: **COMPUTER ASSOCIATES THINK, INC.** [US/US]; One Computer Associates Plaza, Islandia, NY 11749 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **RASMUSSEN, Eric** [US/US]; 275 S. Petit Avenue, Ventura, CA 93004 (US).

(74) Agent: **CARTER, DELUCA, FARRELL & SCHMIDT, LLP**; 445 Broad Hollow Road, Suite 225, Melville, NY 11747 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) **Title:** SYSTEM AND METHOD FOR MANAGING A PATH ENVIRONMENT VARIABLE

(57) **Abstract:** A method and/or system for managing path environment variables in a data processing or computer system includes the steps of reading at least one path environment variable string; parsing at least one path variable from the at least one path environment variable string; and determining if a file directory associated with the at least one path variable exists; if the file directory does not exist, deleting the at least one path variable and reassembling the path environment variable string. Furthermore, the method and system will allow a user to rearrange the order of the path variables so operating system program searching could be optimized to allow known heavily used directories to be searched first thereby reducing processing time.

WO 2005/114409 A2

SYSTEM AND METHOD FOR MANAGING A PATH ENVIRONMENT VARIABLE

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to an application entitled "SYSTEM AND METHOD FOR MANAGING A PATH ENVIRONMENT VARIABLE" filed in the United States Patent and Trademark Office on May 21, 2004 and assigned Serial No. 60/573,553, the contents of which are hereby incorporated by reference.

BACKGROUND

1. Field

The present disclosure relates generally to data processing and desktop maintenance systems, and more particularly, to a system and method for managing a path environment variable in an operating system of a data processing system.

2. Description of the Related Art

The Windows™ operating system PATH is an environment string that contains a long list of file system directory paths. When launching any program in a conventional data processing or computer system, the Windows™ operating system searches all directories in the PATH environment variable string to find the requested program. There is a USER and SYSTEM PATH each of which can be, depending on the Windows™ version, up to 1024 chars in length. When a program search is done, the two PATHs are concatenated and searched as one. Long PATHs will slow down program execution. Additionally, there is the chance of

running out of PATH space during installation of a program. This will prevent the program from installing.

Older or poorly written Windows™ application uninstall programs, even after they have removed all of their files, tend to leave behind entries in the Windows™ PATH environment variable string. The directories left behind no longer exist on the file system so they not only slow down program loading but can contribute to the problem of running out of PATH space.

During install program testing, developers and testers install and uninstall programs many times changing the install locations as they test. The install program typically leaves behind many directories, resulting in clogged PATHs. Eventually, the install program would fail due to lack of space in the PATH environment variable string.

Path maintenance is currently done in the Windows™ Environment Variable dialog. The directory strings are concatenated and separated by a semi-colon making it unreadable and difficult to maintain. To illustrate the awkward interface used in Windows™ to maintain program PATH information, FIG. 1 is a screenshot of the Windows XP System Properties dialog 100 further expanded to show the Environment Variable lists 102 for User and System variables and the Edit System Variable dialog 104, which contains the System PATH, edit field. Due to its layout, the PATH string 106 is hard to edit and extremely error prone due to the hand editing required. The user must horizontally scroll through a standard single line edit field 106, where the directories are delimited by semi-colons. Directories are manually removed by deleting character by character until the delimiting semi-

colon is found. This can lead to mistakes that can cause programs to no longer function.

Therefore, a need exists for techniques to allow computer users, e.g., testers/developers, to view PATHs in a more user-friendly way and to determine which PATHs actually exist on a file system of a data processing or computer system. Furthermore, techniques are needed to facilitate computer system maintenance and for deleting obsolete directories.

SUMMARY

A method and system for managing environment variables in a data processing or computer system are provided. The methods and systems of the present disclosure provide a software utility and user interface for facilitating file directory maintenance.

In one embodiment, the user interface includes a standard Windows™ operating system dialog containing two list boxes, one for the SYSTEM PATH and one for the USER PATH. A Remove button is placed next to each list. When the software utility is launched, it reads and parses each path variable and loads each list box with a vertical list of file directories associated with each path variable. Each directory is examined on the file system to see if it actually exists. The user interface further includes another column to display Yes or No based on the existence of the directory on the file system. A user may then delete the path variable that has no corresponding directory on the file system.

In a further embodiment of the present disclosure, the method and system will allow the user to rearrange the order of the directory strings in the list so Windows™ program searching could be optimized to allow known heavily used directories to be searched first thereby reducing processing time.

In conventional methods, system directories are shown in their raw format (i.e. %systemroot%). The method and system of the present disclosure will expand these directory designators to their actual values (e.g., %systemroot% = C:\Windows). The System and User PATH listboxes will translate these directory names to the actual names.

Furthermore, directories that contain spaces are often represented by their DOS short path name. For example, C:\Program Files would be shown as C:\PROGRA~. Some install programs install their program paths as DOS short names. The method and system of the present disclosure will expand DOS short names into the corresponding Windows™ long path names when listing the paths making it easier for the user to appreciate the paths directory structure.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other aspects, features, and advantages of the present disclosure will become more apparent in light of the following detailed description when taken in conjunction with the accompanying drawings in which:

FIG. 1 is a screen shot of a conventional data processing system for managing a path environment variable;

FIG. 2 is a block diagram of an exemplary system for managing a path environment variable in accordance with an embodiment of the present disclosure;

FIG. 3 is a flow chart illustrating a method for managing a path environment variable in accordance with an embodiment of the present disclosure; and

FIG. 4 is an exemplary screen shot of an interface for employing a method for managing a path environment variable in accordance with an embodiment of the present disclosure.

DETAILED DESCRIPTION

Preferred embodiments of the present disclosure will be described hereinbelow with reference to the accompanying drawings. In the following description, well-known functions or constructions are not described in detail to avoid obscuring the present disclosure in unnecessary detail. In the following description, PATH (capitalized) is used to distinguish the Windows™ operating system environment variable called PATH, vs. the more general usage of the word path that indicates a file system directory structure (e.g., c:\Program Files\Pathclean).

A method and system for managing environment variables in a data processing or computer system are provided. The methods and systems of the present disclosure provide a software utility and user interface for facilitating file directory maintenance. In the following description, the method and system of the

present disclosure will be described in conjunction with the PATH environment variable of Microsoft's™ Windows™ operating system.

It is to be understood that the present disclosure may be implemented in various forms of hardware, software, firmware, special purpose processors, or a combination thereof. In one embodiment, the present disclosure may be implemented in software as an application program tangibly embodied on a program storage device. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture such as a data processing machine or a personal computer. With reference to FIG. 2, preferably, the machine 200 is implemented on a computer platform having hardware such as one or more central processing units (CPU) 202, a random access memory (RAM) 204, a read only memory (ROM) 206 and input/output (I/O) interface(s) such as a keyboard 208, cursor control device 210 (e.g., a mouse or joystick) and display device 212. The computer platform also includes an operating system and micro instruction code. The various processes and functions described herein may either be part of the micro instruction code or part of the application program (or a combination thereof) which is executed via the operating system. In addition, various other peripheral devices may be connected to the computer platform such as an additional data storage device, a printing device and a scanning device 216.

It is to be further understood that, because some of the constituent system components and method steps depicted in the accompanying figures may be implemented in software, the actual connections between the system components (or the process steps) may differ depending upon the manner in which

the present disclosure is programmed. Given the teachings of the present disclosure provided herein, one of ordinary skill in the related art will be able to contemplate these and similar implementations or configurations of the present disclosure.

Referring now to FIGS. 3 and 4, a method for managing an environment variable in a computer system will now be described. FIG. 3 is a flow chart illustrating a method for managing a path environment variable in accordance with an embodiment of the present disclosure. FIG. 4 is an exemplary screen shot of an interface for employing a method for managing a path environment variable in accordance with an embodiment of the present disclosure.

Initially, in step 302, the utility software application for managing the PATH environment variable is launched. The PATH environment variable string is read (step 304) and each path variable, e.g., a file directory, is parsed (step 306). The parsing of the path variables may be accomplished by finding successive delimiting semi-colons in the PATH environment variable string 106 as shown in FIG. 1. For example, the user and system paths are read from the Windows registry using the Windows API function RegOpenKeyEx(). The system PATH string is read from the registry key LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\Environment\PATH and the user PATH string is read from CURRENT_USER\Environment\PATH. An internal function ParsePathIntoListBox() iterates through each PATH string looking for semi-colons and separating each individual directory name when a semi-colon is found. As shown in the example of FIG. 1, an exemplary path variable may be C:\CA_APPSW 108.

Once all the path variables are parsed, the path variables are complied and loaded in a user interface dialog 400 (step 308), as shown in FIG. 4. The user interface 400 includes a standard Windows™ dialog containing two list boxes, one for the SYSTEM PATH 402 and one for the USER PATH 404. A Remove button 406, 408 is placed next to each list 402, 404, function of which will be described below.

Once the lists are compiled, the utility application determines if a file directory associated with each path variable exists (step 310). This may be accomplished by checking the root directory, e.g., C:\, of the file system of the computer to determine if the file directory exists. Alternatively, if a directory does exist but has no files in it, it may be deemed not to exist. For example, the C runtime library function _chdir() is used to check for directory existence. It attempts to change the Windows default directory to the directory in question. If _chdir() succeeds, the path exists. To check the directory to see if it is empty a Microsoft MFC CFileFind object is created and CfileFind::FindFile("<dir>*.*") is called.

If the file directory does not exist (step 312), a NO will be placed in a EXISTS column 410 of the user interface 400. Otherwise, if the file directory does exists, a YES will be placed in the EXISTS column 410.

If the directory does not exist as indicated by a NO in the EXISTS column 410, the user may easily be prompted to delete the corresponding path variable from the PATH environment variable string (step 314). As illustrated in FIG. 4, the SYSTEM PATH variable C:\Program Files\CA\DCS\CAWIN\ 412 does not exist. The user may simply highlight the non-existent directory and click the Remove

button 406 to delete the path variable. This action will subsequently remove the path variable from the PATH environment variable string 106 and the PATH environment variable string will be reassembled without the path variable. By removing path variables corresponding to non-existent directories, a computer system employing the method of the present disclosure will launch applications faster since it will not waste time search for directories no longer present on the computer's hard drive.

It is to be appreciated that the deletion of path variables associated with non-existent file directories may be performed automatically by the utility application without user intervention. The automatic deletion of path variables may be useful in performing periodic scheduled maintenance of the operating system.

Once the user has deleted all non-existent directories, the user may reorder the remaining path variables of the directory strings in the PATH variable lists 402, 404 so Windows™ program searching could be optimized to allow known heavily used directories to be searched first thereby reducing processing time (step 316). To reorder the path variables, the user will highlight a desired path variable and move it to a desired position by utilizing the UP button 414 or DOWN button 416 in the user-interface 400. Once the preferred order is set, the utility program will reassemble the PATH environment variable string using the parsed path variables, delimiting each with a semi-colon. For example, a C++ for() loop is used to concatenate each directory from the Pathclean listbox into a Cstring object. The Cstring += (append operator) is used. The resulting Cstring object is then written back to the registry using RegSetValueExt() .

A method and system for managing environment variable in a computer system has been described. The method and system in accordance with the principles of the present disclosure will be beneficial for maintaining test/development machines where software is installed and uninstalled continually. Furthermore, the method and system of the present disclosure may be utilized for general periodic cleanup for all Windows™ desktop machines.

While the disclosure has been shown and described with reference to certain preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the disclosure as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A method for managing a path environment variable in a data processing system, the method comprising the steps of:

reading at least one path environment variable string;

parsing at least one path variable from the at least one path environment variable string; and

determining if a file directory associated with the at least one path variable exists.

2. The method according to claim 1, further comprising the step of, if the file directory does not exist, deleting the at least one path variable.

3. The method according to claim 2, further comprising the step of reassembling the path environment variable string.

4. The method according to claim 1, wherein the at least one path environment variable string is a system path or a user path.

5. The method according to claim 1, further comprising the steps of:

compiling a list of parsed path variables;

presenting the list to a user; and

presenting a prompt to the user requesting deletion of a path variable associated with a non-existent file directory.

6. The method according to claim 5, further comprising the step of presenting a prompt to the user requesting reordering of the list of parsed path variables.

7. The method according to claim 6, further comprising the step of reassembling the path environment variable string with the reordered list of parsed path variables.

8. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for managing a path environment variable, the method steps comprising:

reading at least one path environment variable string;

parsing at least one path variable from the at least one path environment variable string; and

determining if a file directory associated with the at least one path variable exists.

9. The program storage device according to claim 8, further comprising the step of, if the file directory does not exist, deleting the at least one path variable.

10. The program storage device according to claim 9, further comprising the step of reassembling the path environment variable string.

11. The program storage device according to claim 8, wherein the at least one path environment variable string is a system path or a user path.

12. The program storage device according to claim 8, further comprising the steps of:

compiling a list of parsed path variables;

presenting the list to a user; and

presenting a prompt to the user requesting deletion of a path variable associated with a non-existent file directory.

13. The program storage device according to claim 12, further comprising the step of presenting a prompt to the user requesting reordering of the list of parsed path variables.

14. The program storage device according to claim 13, further comprising the step of reassembling the path environment variable string with the reordered list of parsed path variables.

15. A computer system for managing a path environment variable, the system comprising:

means for reading at least one path environment variable string;

means for parsing at least one path variable from the at least one path environment variable string;

means for determining if a file directory associated with the at least one path variable exists;

means for deleting the at least one path variable if the file directory does not exist; and

means for reassembling the path environment variable string.

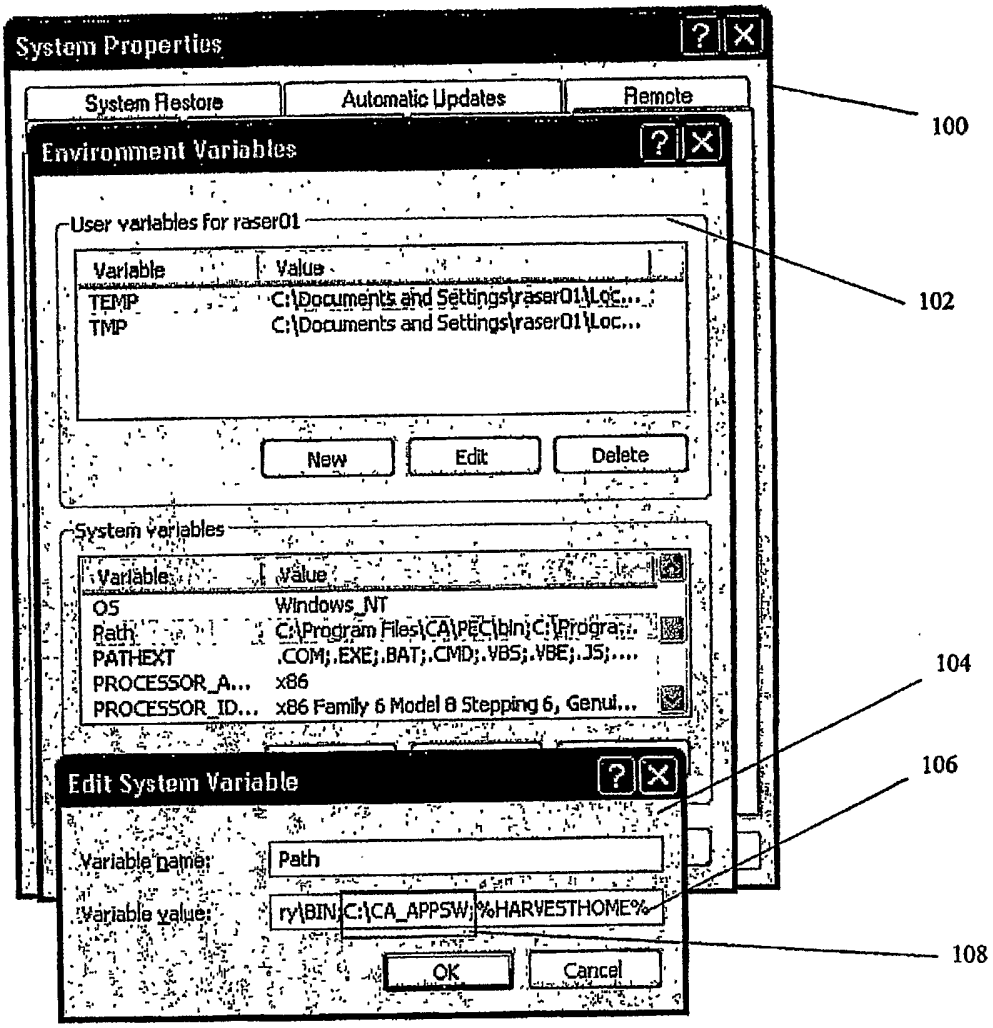

System Properties
System Restore
Automatic Updates
Remote
Environment Variables
User variables for raser01
Variable
Value
TEMP
C:\Documents and Settings\raser01\Loc...
TMP
C:\Documents and Settings\raser01\Loc...
New
Edit
Delete
System variables
Variable
Value
OS
Windows_NT
Path
C:\Program Files\CA\PEC\bin;C:\Progra...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...
x86
PROCESSOR_ID...
x86 Family 6 Model 8 Stepping 6, Genui...
Edit System Variable
Variable name:
Path
Variable value:
ry\BIN;C:\CA_APPSW;%HARVESTHOME%
OK
Cancel
100
102
104
106
108

FIG. 1

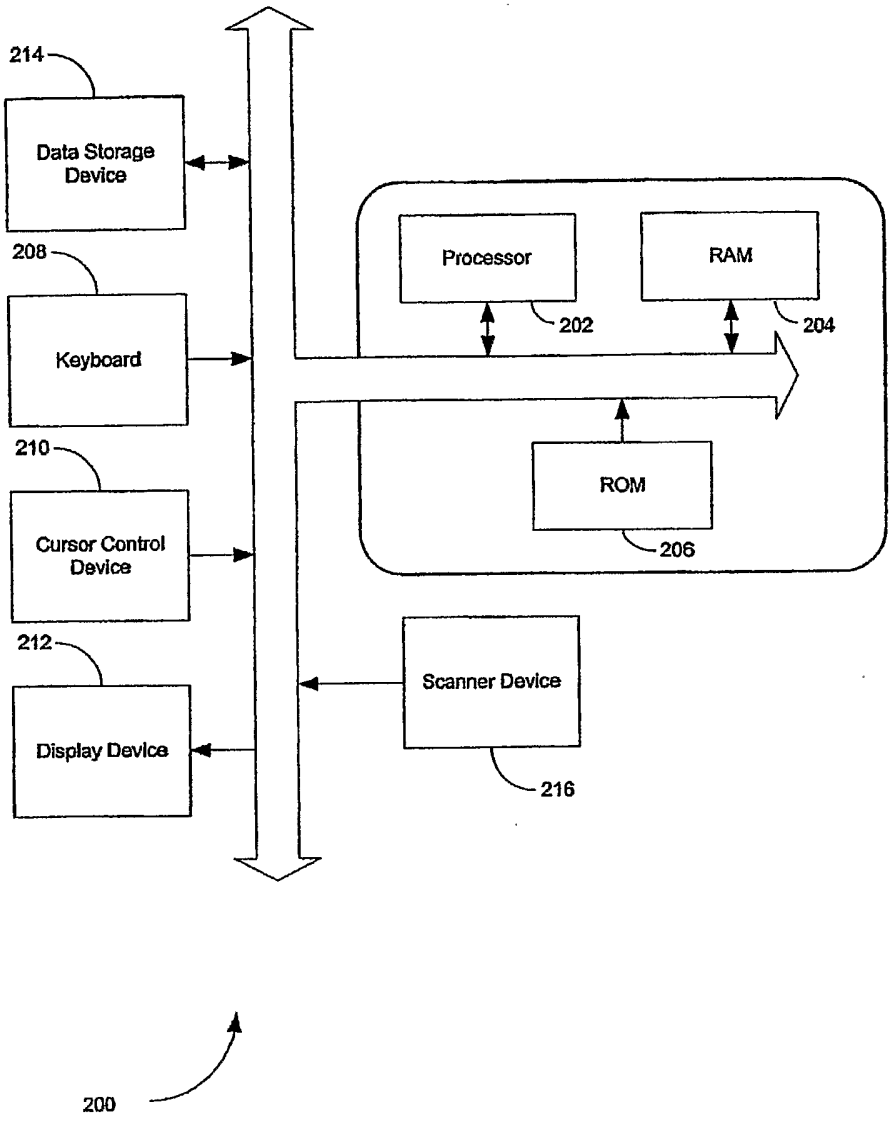

214
Data Storage Device
208
Keyboard
210
Cursor Control Device
212
Display Device
Processor
202
RAM
204
ROM
206
Scanner Device
216
200

FIG. 2

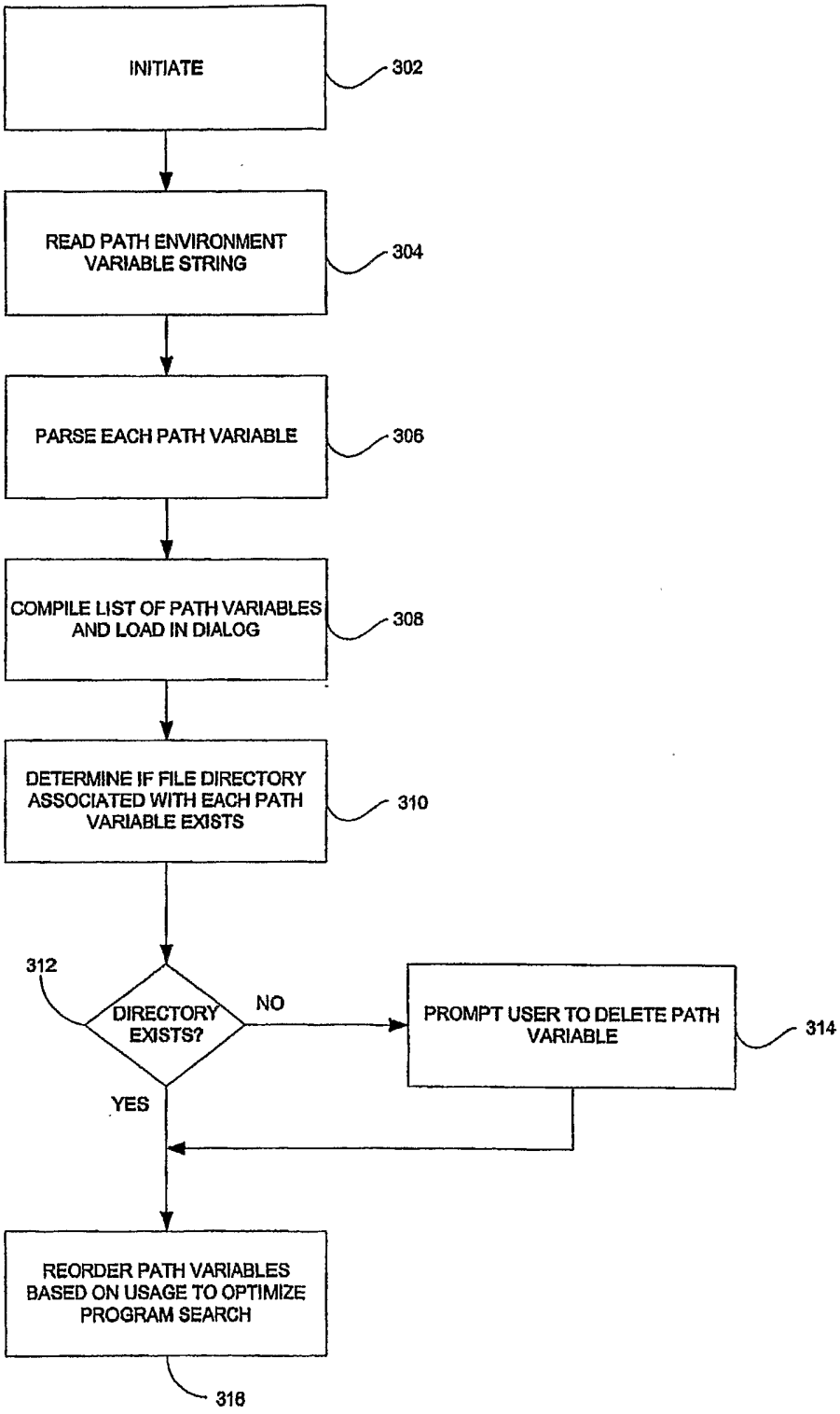

INITIATE
302
READ PATH ENVIRONMENT VARIABLE STRING
304
PARSE EACH PATH VARIABLE
306
COMPILE LIST OF PATH VARIABLES AND LOAD IN DIALOG
308
DETERMINE IF FILE DIRECTORY ASSOCIATED WITH EACH PATH VARIABLE EXISTS
310
312
DIRECTORY EXISTS?
NO
PROMPT USER TO DELETE PATH VARIABLE
314
YES
REORDER PATH VARIABLES BASED ON USAGE TO OPTIMIZE PROGRAM SEARCH
316

FIG. 3

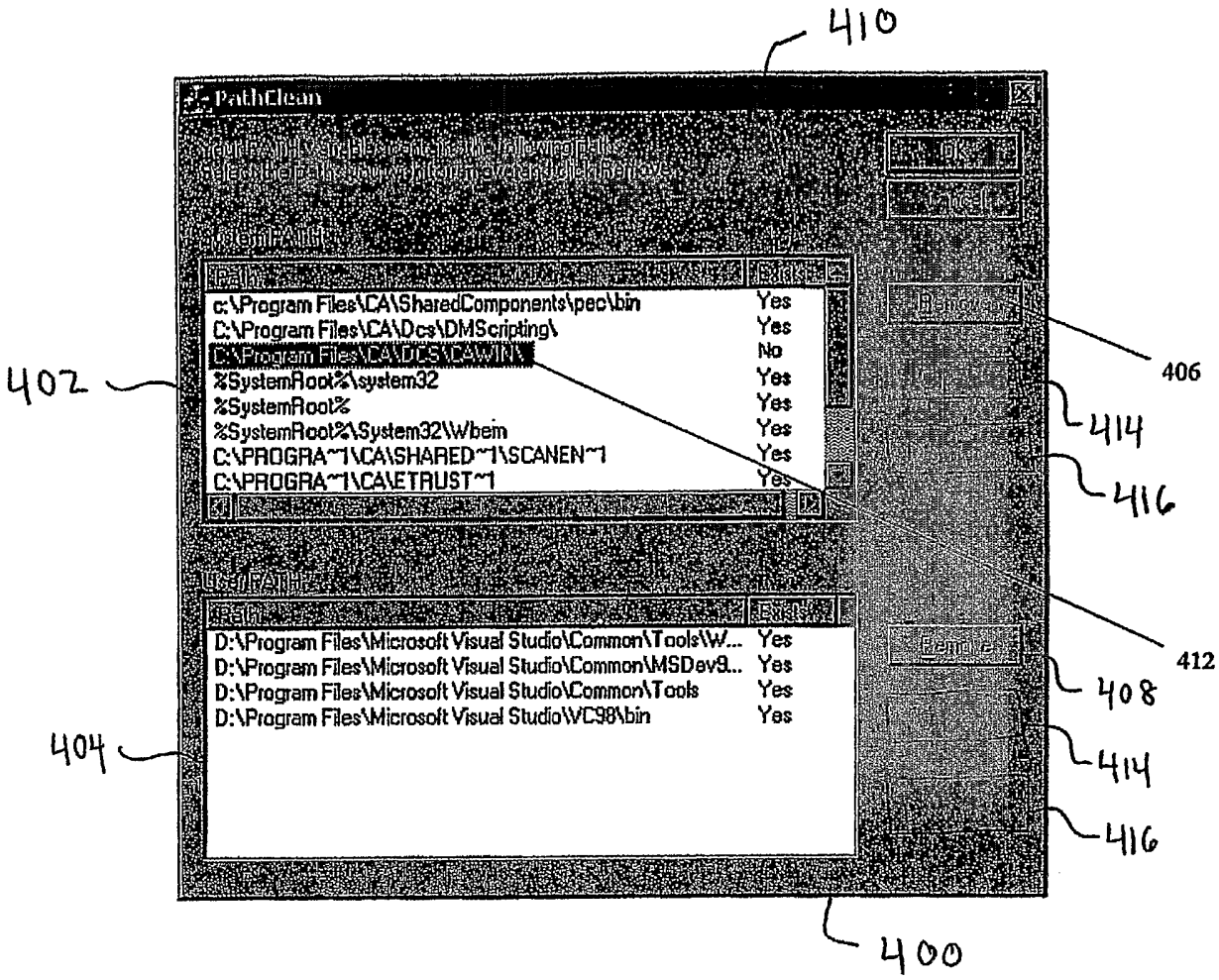

410
PathClean
c:\Program Files\CA\SharedComponents\pec\bin Yes
C:\Program Files\CA\Dcs\DMScripting\ Yes
C:\Program Files\CA\DCS\CAWIN\ No
%SystemRoot%\system32 Yes
%SystemRoot% Yes
%SystemRoot%\System32\Wbem Yes
C:\PROGRA~1\CA\SHARED~1\SCANEN~1 Yes
C:\PROGRA~1\CA\ETRUST~1 Yes
D:\Program Files\Microsoft Visual Studio\Common\Tools\W... Yes
D:\Program Files\Microsoft Visual Studio\Common\MSDev9... Yes
D:\Program Files\Microsoft Visual Studio\Common\Tools Yes
D:\Program Files\Microsoft Visual Studio\VC98\bin Yes
402
404
406
408
412
414
416
414
416
400

FIG. 4