

Method And Apparatus For Performing A Search

5 Field of Invention

The present invention relates broadly to a method and apparatus for performing a search on a spreadsheet program.

10 Background

Spreadsheet programs typically include embedded programs within the spreadsheet programs. One example of a spreadsheet programs is EXCEL from Microsoft of Redmond, Wash. EXCEL includes features for developing an "application",
15 allowing a user to write programs within EXCEL to perform a desired task. These programs have codes that are embedded within and specific to the spreadsheet.

Typically, when a user wishes to search through the code of the embedded program, the user will have to export and convert the code of the embedded program,
20 search through the code manually line by line, and if necessary amend the codes. The need to search and amend the embedded code may, for example, arise due to a run-time-error when the application within the spreadsheet program is executed and a search needs to be performed to identify a bug in the program.

25 When there are codes in multiple programs, a user would have to manually select each program in turn to export and convert the code, and then do a visual debugging check.

30 As described above, the conventional method of performing such a search is often tedious, time consuming and prone to errors due to visual lethargy.

Therefore, there is a need for a method and apparatus for performing a search that overcomes at least some of the above problems.

35 Summary of Invention

According to an aspect of the present invention, there is provided a method for performing a search on a spreadsheet program, said spreadsheet program having a first layer and a second layer, the method including: receiving a search input, said search input in the form of an input string, accessing a code at the second layer, identifying a portion of the code for searching, and performing a search for the input string within the identified portion of the code, wherein the search input is provided at the first layer.

The second layer may be embedded in the first layer.

The search input may be pre-determined.

The search input may be provided by a user.

The method may further include obtaining code line position information of the portion of the code, wherein the code line position information determines the portion of the code for searching.

The position information may include a first line position and a second line position of the portion of the code.

The first line position may be a start line position and the second line position may be an end line position.

The portion of the code may include a module portion having a procedure portion within the module portion.

The method may further include counting a number of module portions and/or procedure portions in the code.

The method may further include counting a number of module portions and/or procedure portions of the code containing the input string.

The method may further include counting a number of module portions and/or procedure portions without the input string.

The method may further include inserting the input string into a pre-defined location within the portion of the code.

5 The method may further include counting a number of module portions and/or procedure portions where the input string is inserted.

The spreadsheet program may include an integrated programming language integrated to read data from spreadsheets.

10

The code at the second layer may be written in the integrated programming language.

The integrated programming language may be Visual Basic.

15

The method may further include logging any changes made to the code.

According to another aspect of the present invention, there is provided an apparatus for performing a search on a spreadsheet program, said spreadsheet program
20 having a first layer and a second layer, the apparatus including means for receiving a search input, said search input in the form of an input string, means for accessing a code at the second layer, means for identifying a portion of the code for searching, and means for performing a search for the input string within the identified portion of the code, wherein the search input is provided in the first layer.

25

The second layer may be embedded in the first layer.

The search input may be pre-determined.

30

The search input may be provided by a user.

The apparatus may further include means for obtaining code line position information of the portion of the code, wherein the code line position information determines the portion of the code for searching.

The position information may include a first line position and a second line position of the portion of the code.

5 The first line position may be a start line position and the second line position may be an end line position.

The portion of the code may include a module portion having a procedure portion within the module portion

10

The apparatus may further include means for counting a number of module portions and/or procedure portions in the code.

15 The apparatus may further include means for counting a number of module portions and/or procedure portions of the code containing the input string.

The apparatus may further include means for counting a number of module portions and/or procedure portions without the input string.

20 The apparatus may further include means for inserting the input string into a pre-defined location within the portion of the code.

The apparatus may further include means for counting a number of module portions and/or procedure portions where the input string is inserted.

25

The spreadsheet program may include an integrated programming language integrated to read data from spreadsheets.

30 The code at the second layer may be written in the integrated programming language.

The integrated programming language may be Visual Basic.

The apparatus may further include a log file for logging any changes made to the code.

5 In accordance with another aspect of the present invention, there is provided a method for performing a search on a spreadsheet program, said spreadsheet program having a first layer and a second layer, the method including receiving a search input, said search input in the form of an input string, accessing a code at the second layer, identifying a portion of the code for searching, and performing a search for the input string within the identified portion of the code, wherein the search input is provided at the
10 first layer and the code is associated with a test program for automatic test equipment.

The second layer may be embedded in the first layer.

15

Brief Description of the Drawings

Embodiments of the invention will be better understood and readily apparent to one of ordinary skill in the art from the following written description, by way of example
20 only, and in conjunction with the drawings, in which:

Figure 1 shows a flowchart 100 illustrating a method of performing a search on a spreadsheet program according to an example embodiment;

25 Figure 2 shows a flowchart 200 of a searching method employed by an example embodiment of the present invention;

Figure 3a and Figure 3b are screen shots of a Graphical User Interface (GUI) implemented in one example embodiment.

30

Detailed Description

35 In an example embodiment of the present invention, there is provided a search tool for performing a search on a spreadsheet program. In this example, the spreadsheet program is EXCEL from Microsoft of Redmond, Wash. However, it will be appreciated

that the method and apparatus of the present invention can also be applied to other spreadsheet programs, for example, Lotus 1-2-3.

EXCEL is a spreadsheet program that allows a user to create a "workbook". A
5 workbook is a collection of data that is organized into different spreadsheets (i.e. worksheets). A worksheet typically includes a grid of cells, each cell holding a piece of data or an equation defining how the data value should be calculated from the values in other cells. EXCEL also includes features for defining the spreadsheets and developing an "application". EXCEL has been used for applications such as developing accounting
10 systems. EXCEL can also be used to make an application through the use of Visual Basic - a programming language that is built into EXCEL. For example, EXCEL can be customized to create a test program for automatic test equipment, or to create macros within EXCEL, which are short run applications on EXCEL.

15 Figure 1 shows a flowchart 100 illustrating a method of performing a search on a spreadsheet program according to an example embodiment.

The spreadsheet program in the example embodiment is EXCEL and includes a first layer 102 and a second layer 104. The first layer 102 may include an EXCEL
20 worksheet/workbook and the second layer 104 may include applications e.g. macros, customized programs for automatic test equipment, etc. and their associated codes. The second layer may be embedded in the first layer of the spreadsheet program such that the macros, customized programs etc., and their associated codes are embedded within the spreadsheet program.

25 Each program code in the second layer 104 may include one or more module portions and each module portion may include one or more procedure portions (i.e. subroutines and/or functions).

30 Step 106 involves receiving a search input, where the search input is provided in the first layer of the spreadsheet program. The search input may be in the form of a string. Step 108 involves accessing the code at the second layer 104. Step 110 involves identifying a portion of the code for searching and Step 112 involves performing a search for the input string within the portion of the code.

The search input may be a pre-determined search input incorporated into the search tool or provided by a user.

5 In Step 108, accessing the code at the second layer 104 does not require additional operations such as extracting and converting the code into another form prior to performing the search.

10 Subsequently, to identify a portion of code for searching (i.e. Step 110), code line position information of the portion of the code may be obtained by identifying the program codes defining a start line and an end line. The code line position information determines the portion of the code for searching. The code line position information may include a start line position and an end line position of the portion of the code. Other code line positions e.g. body line position may also be obtained depending on the type of
15 search required.

 For example, where the portion of the code is a module portion, the start line position and end line position of the module portion is obtained (similar to the code line position information) to identify the module portion for searching. Similarly, where the
20 portion of the code is a procedure portion within the module portion, the start line position and the end line position of the procedure is obtained to identify the procedure portion for searching.

 Further, where there may be several types of module portions in the code, the
25 method allows for searching on a specific type of module portions. Differentiation of the types of module portions may be by a module name assigned to the specific type of module portions thereby allowing the module portion(s) of the same type to be identified. In this example, the search input at Step 106 may further include or in place of a user or pre-determined input for the module name/type (not shown in Figure 1).

30 In Step 112, the search for the input string is performed on the portion of the code that has been identified in Step 110. The input string may be included as the search input. Where the search input is provided by the user, the user may enter the input string as the search input. The search input may be, for example, an "On Error

GoTo" statement (i.e. error handling code), other common codes that may contribute to runtime error (RTE) of the program, or any other statement/string that the user wishes to search for in the code. Performing the search (Step 112) involves going through each line of the portion of the code to search for the input string. For example, if the portion of the code is a module portion, performing the search will involve going through each line of the module portion. Where a module portion includes a subroutine and a function, performing the search will involve going through each line of code in the subroutine and function to find the input string. It will be appreciated that the method may be used to search the entire code or specific module/procedure portions of the code.

Further, the method may include a step of inserting the input string into a searched location within the portion of the code (not shown in Figure 1). In an alternative, the pre-defined location may be selected by the user or may be a default location that is incorporated into the search tool. For example, in a case where the input string is not found after the search is performed on the portion of the code, the input string may be inserted into the portion of the code.

The search may be conducted on one or more module portions and/or procedure portions or the entire code at the second layer depending on the requirements of the user.

After the search is conducted, results of the search may be displayed. Search results may be in various forms depending on the user's requirement and may include an indication of whether the input string was found, the number of module portions searched, etc. Consequently, the method may also include one or more of the following steps:

- counting the number of module portions and/or procedure portions in the code
- counting the number of module portions and/or procedure portions of the code containing the input string
- counting the number of module portions and/or procedure portions without the input string
- counting the number of module portions and/or procedure portions where the input string is inserted.

Where the input string has been inserted into the portion of the code, the method may include saving a copy of the code that has been amended (due to the insertion of the input string) and an indication that an amendment has been made to the code may also be included in the search results. The search results may be presented to the user as display data through a user interface. Figure 3b shows an example of how the results of the search may be displayed.

Figure 2 shows a flowchart 200 of a searching method employed by an example embodiment of the present invention. In this example the search is performed on a code of a test program used for automatic test equipment. The test program is written in Visual Basic and embedded in EXCEL.

Test programs generally have some form of error handling (i.e. exception handling) to handle the occurrence of some condition that changes the normal flow of execution. It has been found that the absence of an "On Error GoTo" or equivalent statement may cause run-time-error (RTE) when the test program is executed. One way of debugging such errors is to determine whether the "On Error GoTo" statement is present in the code. A search is therefore performed on the code of the test program to determine whether the "On Error GoTo" statement is present and to insert the statement into the code if the "On Error GoTo" statement is absent.

The test program may include one or more types of modules and each module may include one or more procedures (i.e. subroutines and/or functions). For example, each module may include one function and one subroutine. The search may be conducted on a specific module, for example, a Visual Basic for Test (VBT) module or a Visual Basic (VB) module in the code. VBT is derived from Visual Basic, where Visual Basic is used to create an environment with testing capabilities and features for e.g. automatic test equipment. "Environment" generally refers to applying a tool or means using a high level programming language for building application software.

In an embodiment, the method starts at Step 202 and receives the "On Error GoTo" statement as a search input at Step 204. In this example embodiment, the search input is a pre-determined default search input that is incorporated into the search tool. Alternatively, the search input may be provided by the user. Step 208 includes accessing

the code of the test program in the second layer 206. Step 210 determines whether iteration of all the modules in the test program is completed. If "Yes", display data is updated with the necessary information at Step 248. Display data may include a summary of the search results or any other information required by the user. If "No", the method continues to Step 212 to obtain a total line count of the module. Step 214 determines whether module line iteration has been completed (i.e. whether a search has been performed on each line in the module). If "Yes", a feedback is provided to Step 210 and the display data is updated accordingly at Step 248. When a "Yes" feedback is provided to Step 210 and iteration of all the modules is not completed, the method will continue to obtain the total line count of the next module at Step 212 until iteration of the modules in the code is completed.

If the output at Step 214 is "No" (i.e. module line iteration is not done), the method continues to Step 216 to determine whether the name of the module is available. If "No", the method continues to search the next line of the module at Step 218 for the name of the module. For example, if a current module line count is a first line and the name of the module is absent, the method continues to search the second line (i.e. current module line count + 1) to determine whether the name of the module is available. Search for the name of the module continues line by line until iteration of all the lines in the module is completed.

If the output at Step 216 is "Yes" then the method continues to Step 220 to determine whether the module is a VBT module. If "No" (i.e. the module is not a VBT module), a search is performed on the next module at Step 222. The next module is identified by obtaining line position information, in this case, a line position that is equal to the total current module line count plus the total current subroutine and function line count. When the next module is identified, Steps 214 to 220 are repeated until iteration of all the modules is completed.

If the output at Step 220 is "Yes", the method continues to Step 224 to obtain line position information of the subroutine/function within the module. Each subroutine/function in the module is identified by obtaining one or more line positions for, example, the start line position, end line position, module line position and current line position, etc. At Step 226, the method includes determining whether a current line count

is less than the subroutine/function end line count. If "Yes", at Step 228 the method continues with determining whether the "On Error ToGo" statement is found by searching the current line for the "On Error ToGo" statement. If the "On Error ToGo" statement is not found in the current line (i.e. output at Step 228 is "No"), the search
5 continues to the next line at Step 230 by increasing the current line count by one line (i.e. current subroutine/function line count + 1). The search for the "On Error ToGo" statement continues until iteration of the entire subroutine/function is completed.

When the "On Error GoTo" statement is found at Step 228, an Error Flag at Step
10 232 will not be "True" (i.e. output is "No") indicating that the Error Flag is not necessary and the method proceeds to search the next module by going back to Step 222 and repeating the process described above.

When the current line count is greater than the current subroutine/function end
15 line count at Step 226 (i.e. output is "No"), it is implied that the "On Error GoTo" statement is not found in the subroutine/function and the method proceeds to Step 232 and the Error Flag is set to "True" (i.e. output is "Yes"), indicating that error handling is necessary. The method then continues to Step 234.

20 Step 234 determines whether an Add Flag status should be set as "True". The Add Flag provides an indication of whether the "On Error GoTo" statement should be inserted. If the output at Step 234 is "No" searching is performed on the next module by going back to Step 222.

25 If the output at Step 234 is "Yes" the method proceeds to reset the current line count at Step 236 and proceeds to a line that is one line after before the module line position (i.e. Goto module line + 1) at Step 238 and inserts the "On Error GoTo" statement at Step 240. The "module line + 1" is a pre-determined location within the subroutine/function. Other locations for inserting the statement are possible, depending
30 on the program architecture. The "On Error GoTo" statement may include a label, XX, (i.e. "On Error GoTo XX") where "XX" may be the name of the error handler e.g "Handler" or "ErrorHandler", etc.,) In this example, an "On Error GoTo Handler" statement is inserted in the module line +1 position as execution of the program will be switched to a predefined error handler when error occurs. Consequently, the label

“Handler” which includes a code that handles errors e.g. “Error Handler Code” is also inserted into the subroutine/function into a line that is one line before the end line of the subroutine/function (i.e. Goto End line – 1) as shown in Step 242 and Step 246 such that when error occurs, execution of the program is directed to the “Handler”. After insertion
5 of the relevant error handling codes, searching proceeds to the next module in the program.

The search ends at Step 250 when iteration of all modules is completed and the display data (Step 248) is updated accordingly.

10 Figures 3a and 3b are screen shots of a Graphical User Interface (GUI) implemented in one example embodiment. The GUI of the search tool includes a window 300 that is split into a plurality of panes. A browsing pane 302 allows the user to browse through folders in a data storage medium, e.g. a hard disk on a computer, to locate a specific folder. The files within the folder are displayed in a display pane 304. The user
15 may select a specific spreadsheet file in the display pane 304 to perform a search (Figure 3b).

The window 300 includes selection boxes, namely a “check” box 308 and an “add” box 310 for selection by the user. When the “check” box 308 is selected by the
20 user, the search tool will perform a search on the code of the embedded program for the input string. When the “add” box 310 is chosen, the search tool will insert the input string into the pre-defined location within the code. The user may select both or either one of the selection boxes. In Figure 3b, both the “search” box 308 and the “add” box 310 are selected.

25

The window 300 also includes an search results display pane 306 that displays a summary of the search results. In the example embodiment, the following information is displayed in the search results display pane 306:

30

- Total number of modules in the embedded program
- Number of subroutines without the input string
- Number of subroutines with the input string
- Number of functions without the input string
- Number of functions with the input string

- Indication of a log file 316 of the search (see Figure 3b)

The log file may be used to keep track of any changes (e.g. amendments) that have been made to the code during the search. An indication of the log file including the name and location of the log file may be displayed in the search results display pane 306 or the log file itself may be displayed.

Figure 3b shows one way of displaying the search results. It will be appreciated that other information e.g., number of functions/subroutines where the input string has been inserted, number of lines of code in each function/subroutine, etc. may be included in the summary pane, depending on the type of information required by the user.

In the above example, the input string is an "On Error GoTo" statement (i.e. error handling code) as the search tool is pre-programmed to search for the presence of error handling in the code. The input string may also be any other pre-defined string / statement.

The window 300 further includes an "execute" button 312 which runs the search upon selection by the user, and an "exit" button 314 to close the search tool when the search has been completed.

The split display of the window 300 may be implemented in a number of different ways, including side-by-side, concatenated etc. and is not limited to the arrangement depicted in the drawings in this description. Further, more than one window may also be implemented.

An example work flow a user may adopt when using the search tool of the example embodiment is as follows:

- a. launch the search tool
- b. select the file the user wishes to perform the search on
- c. select the "check" box and/or the "add" box
- d. click the "execute" button to start the search

In the above example, the search input is a pre-determined string (i.e. "On Error GoTo" statement) is a default search input. However, the search tool may also provide for the user to input any user-defined string such that the search tool searches for and/or adds the user-defined string into the portion of the code. In this case, an additional step to the work flow above may be for the user to input the user-defined string prior to step c. above – e.g. between step b. and step c. In addition, it may also be necessary to allow the user to specify a pre-defined location to add the string should the "add" function be selected.

An embodiment of the present invention may include an apparatus for performing a search on a spreadsheet program, the spreadsheet having a first layer and a second layer. The apparatus includes means for receiving the search input, means for accessing the code at the second layer, means for identifying a portion of the code for searching and means for performing a search for the input string within the portion of the code.

Further, the apparatus may include means for obtaining code line position information of the portion of the code where the code line position information provides the portion of the code for searching. Also included are means for counting the number of module portions and/ means for counting a number of module portions and/or procedure portions in the code, means for counting a number of module portions and/or procedure portions of the code containing the input string, and means for counting a number of module portions and/or procedure portions without the input string. Where insertion of the input string is required, the apparatus may include means for inserting the input string into a pre-defined location within the portion of the code, and means for counting a number of module portions and/or procedure portions where the input string is inserted.

Example embodiments of the present invention may have the following features and advantages.

Example embodiments of the present invention can perform a search on the embedded code efficiently as there is no need to export and convert the code prior to performing the search. Further, the user does not have to review the code manually line

by line to amend or search for input strings as the search is performed directly at the second layer and this helps to save time.

- 5 It will be appreciated by a person skilled in the art that numerous variations and/or modifications may be made to the present invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects to be illustrative and not restrictive.

Claims:

- 5 1. A method for performing a search on a spreadsheet program, said spreadsheet program having a first layer and a second layer, the method including:
receiving a search input, said search input in the form of an input string;
accessing a code at the second layer;
identifying a portion of the code for searching; and
10 performing a search for the input string within the identified portion of the code, wherein the search input is provided at the first layer.
- 15 2. The method as claimed in claim 1, wherein the second layer is embedded in the first layer.
3. The method as claimed in claim 1 or claim 2, wherein the search input is pre-determined.
- 20 4. The method as claimed in any one of the preceding claims, wherein the search input is provided by a user.
- 25 5. The method as claimed in any one of the preceding claims, further including:
obtaining code line position information of the portion of the code, wherein the code line position information determines the portion of the code for searching.
6. The method as claimed in claim 5, wherein the position information includes a first line position and a second line position of the portion of the code.
- 30 7. The method as claimed in claim 6, wherein the first line position is a start line position and the second line position is an end line position.
8. The method as claimed in any one of the preceding claims, wherein the portion of the code includes a module portion having a procedure portion within the module portion.

9. The method as claimed in claim 9, further including:
counting a number of module portions and/or procedure portions in the code.
- 5 10. The method as claimed in any one of claim 8 or claim 9, further including:
counting a number of module portions and/or procedure portions of the code
containing the input string.
- 10 11. The method as claimed in any one of claims 8 to 10, further including:
counting a number of module portions and/or procedure portions without the
input string.
12. The method as claimed in any one of claims 8 to 11, further including:
inserting the input string into a pre-defined location within the portion of the code.
- 15 13. The method as claimed in claim 12 further including:
counting a number of module portions and/or procedure portions where the input
string is inserted.
- 20 14. The method as claimed in any one of the preceding claims, wherein the
spreadsheet program includes an integrated programming language integrated to read
data from spreadsheets.
- 25 15. The method as claimed in any one of the preceding claims wherein the code at
the second layer is written in the integrated programming language.
- 30 16. The method as claimed in any one of the preceding claims wherein the
integrated programming language is Visual Basic.
17. The method as claimed in any one of the preceding claims, further including
logging any changes made to the code.
18. An apparatus for performing a search on a spreadsheet program, said
spreadsheet program having a first layer and a second layer, the apparatus including:

means for receiving a search input, said search input in the form of an input string;

means for accessing a code at the second layer;

means for identifying a portion of the code for searching; and

5 means for performing a search for the input string within the identified portion of the code, wherein the search input is provided in the first layer.

19. The apparatus as claimed in claim 18, wherein the second layer is embedded in the first layer.

10

20. The apparatus as claimed in claim 18 or claim 19, wherein the search input is pre-determined.

10

21. The apparatus as claimed in any one of claims 18 to 20, wherein the search input is provided by a user.

15

22. The apparatus as claimed in any one of claims 18 to 21, further including:
means for obtaining code line position information of the portion of the code,
wherein the code line position information determines the portion of the code for
20 searching.

20

23. The apparatus as claimed in claim 22, wherein the position information includes a first line position and a second line position of the portion of the code.

25

24. The apparatus as claimed in claim 23, wherein the first line position is a start line position and the second line position is an end line position.

25. The apparatus as claimed in any one of claims 18 to 24, wherein the portion of the code includes a module portion having a procedure portion within the module portion

30

26. The apparatus as claimed in claim 25, further including:
means for counting a number of module portions and/or procedure portions in the code.

27. The apparatus as claimed in any one of claim 25 or claim 26, further including:
means for counting a number of module portions and/or procedure portions of
the code containing the input string.
- 5 28. The apparatus as claimed in any one of claims 25 to 27, further including:
means for counting a number of module portions and/or procedure portions
without the input string.
- 10 29. The apparatus as claimed in any one of claims 25 to 28, further including:
means for inserting the input string into a pre-defined location within the portion
of the code.
- 15 30. The apparatus as claimed in claim 29, further including:
means for counting a number of module portions and/or procedure portions
where the input string is inserted.
- 20 31. The apparatus as claimed in any one of claims 18 to 30, wherein the
spreadsheet program includes an integrated programming language integrated to read
data from spreadsheets.
32. The apparatus as claimed in any one of claims 18 to 31 wherein the code at the
second layer is written in the integrated programming language.
- 25 33. The apparatus as claimed in any one of claims 18 to 32 wherein the integrated
programming language is Visual Basic.
- 30 34. The apparatus as claimed in any one of claims 18 to 33 further including a log file
for logging any changes made to the code.
35. A method for performing a search on a spreadsheet program, said spreadsheet
program having a first layer and a second layer, the method including:
receiving a search input, said search input in the form of an input string;
accessing a code at the second layer;
identifying a portion of the code for searching; and

performing a search for the input string within the identified portion of the code, wherein the search input is provided at the first layer and the code is associated with a test program for automatic test equipment.

- 5 36. The method as claimed in claim 35, wherein the second layer is embedded in the first layer.