



- (51) **International Patent Classification:**
G06F 11/263 (2006.01) *G06F 3/048* (2013.01)
- (21) **International Application Number:**
PCT/US2015/013422
- (22) **International Filing Date:**
29 January 2015 (29.01.2015)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (71) **Applicant:** HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).
- (72) **Inventor:** GRAY, David; 5400 Legacy, Suites 400/475, Plano, Texas 75024 (US).
- (74) **Agents:** HEWLETT PACKARD ENTERPRISE et al.; 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (US).
- (81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,

HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

- as to the identity of the inventor (Rule 4.17(i))
- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))

Published:

- with international search report (Art. 21(3))

(54) **Title:** TEST GENERATION FOR BROWSER-BASED USER INTERFACE

200

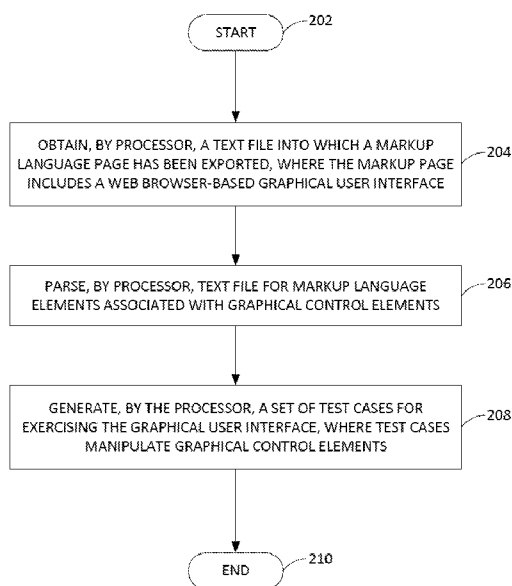


FIG. 2

(57) **Abstract:** An example method is described in which a text file is obtained. A markup language page including a web browser-based graphical user interface is exported into the text file. The text file is parsed for a markup language element associated with a graphical control element of the graphical user interface. A test case is then generated for exercising the graphical user interface. The test case includes a function that manipulates the graphical control element.

TEST GENERATION FOR BROWSER-BASED USER INTERFACE

BACKGROUND

[0001] Graphical user interfaces (GUIs) are tested to ensure that they meet written specifications. Testing typically involves subjecting a GUI to a set of test cases, i.e., a set of conditions or operations that attempts to exercise all of the features of the GUI.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] FIG. 1 is a block diagram of an example system of the present disclosure;

[0003] FIG. 2 illustrates a flowchart of an exemplary method for generating a set of test cases with which to test a web browser-based graphical user interface;

[0004] FIG. 3 illustrates a flowchart of another exemplary method for generating a set of test cases with which to test a web browser-based graphical user interface; and

[0005] FIG. 4 depicts an example high-level block diagram of a computer that can be transformed into a machine that is dedicated to perform the functions described herein.

DETAILED DESCRIPTION

[0006] The present disclosure broadly discloses a method, computer system, and non-transitory computer-readable medium for automating test generation for web browser-based user interface applications. Testing of graphical user interfaces (GUIs) has trended toward automation. However, the graphical and interactive nature of GUI software makes tests difficult to automate. Tests of web browser-based GUIs are even more difficult to automate, because the content of the browser page may change quickly and unexpectedly as design and development progresses, especially in an agile software development environment. As a result, an automated test may fail not because the GUI is defective, but because the test could not adapt to the changes in the content.

[0007] Examples of the present disclosure provide a novel method for automating generation of a set of test cases for use in testing a web browser-based GUI. For instance, examples of the present disclosure export a web page written in a markup language such as hypertext markup language (HTML) into a text file. The text file is parsed for markup language elements, e.g., tags, associated with graphical control elements of the GUI. Test cases are generated that include functions that manipulate the graphical control elements, e.g., functions named for the types of tags parsed from the text file. As a result, the set of test cases can be generated automatically without needing to program every graphical control element of the GUI. This frees the test programmer or test developer from having to find and update every HTML web element identifier, class, or other element attribute when changing or updating the GUI. It also frees the test programmer or test developer from having to add detailed additional programming when a new graphical control element is added to the GUI. Thus, updates to the set of test cases can be generated automatically as changes are made to the GUI.

[0008] Within the context of the present disclosure, the term “graphical control element” is understood to refer to a graphical component of a GUI that interacts with a user. A user directly manipulates a graphical control element in order to access or input information or to access application functions. Graphical control elements include, for example, buttons, menus, drop down lists, text input fields, and selection boxes, among other components. A “graphical control element” may also be referred to as a “GUI widget.”

[0009] Furthermore, within the context of the present disclosure, a “test case” as generated by examples of the present disclosure may not be an executable test case. For instance, the test case could be an input used by a test script or an executable test case to exercise a GUI according to a test design.

[0010] FIG. 1 is a block diagram of an example system 100 of the present disclosure. In one example, the system 100 includes an automated test generation tool 102. The automated test generation tool 102 may include a processor, and, in one case, includes a browser debugger. The automated test generation tool 102 takes as an input a text file 104 into which a markup

language page 103, such as an HTML page, has been exported. The markup language page 103 comprises a web browser-based GUI. In one example, the automated test generation tool 102 also takes as an input metadata 108 associated with graphical control elements of the GUI. The automated test generation tool 102 generates as output a set 106 of test cases for exercising the GUI.

[0011] FIG. 2 illustrates a flowchart of an exemplary method 200 for generating a set of test cases with which to test a web browser-based graphical user interface (GUI). In one example, the method 200 may be performed by the test generation tool illustrated in FIG. 1 or by a computer as illustrated in FIG. 4 and discussed below. For simplicity of explanation, the method 200 is described as being performed by a processor, which could be a component of the test generation tool or the computer.

[0012] At block 202, the method 200 begins. At block 204, the processor obtains a text file. The text file comprises a file into which a markup language page, such as an HTML page, has been exported. The markup language page may comprise code written in alternative or additional programming languages, including JavaScript. The markup page comprises a web browser-based GUI. The processor may export the markup language page into the text file, or the processor may receive the text file from another device.

[0013] At block 206, the processor parses the text file for markup language elements associated with graphical control elements. For example, the processor may parse the text file for basic HTML tags such as <href>, <select>, or <input>, which indicate the presence of known HTML elements. The processor may also parse the text file for other types of HTML elements, such as tables.

[0014] At block 208, the processor generates a set of test cases for exercising the GUI. In one example, the set of test cases involves a set of functions, or possibly sub-routines, that manipulate the graphical control elements of the GUI. In one example, the functions are uniquely named for the types of tags or other markup language elements associated with the specific graphical control elements that the functions are intended to manipulate. For

instance, a “select” function may make a selection from a drop down list. In a further example, the set of test cases may include a generic function that allows an operation on a markup language element. For instance, a “table-select” function may select contents from a particular row and column of a table, or a “find-table” function may locate a row and column of a table containing the contents of a text search string.

[0015] In one example, generating the set of test cases involves updating an existing set of test cases to reflect changes to the GUI. In another example, generating the set of test cases involves generating a new set of test cases.

[0016] The method 200 ends in block 210.

[0017] FIG. 3 illustrates a flowchart of another exemplary method 300 for generating a set of test cases with which to test a web browser-based graphical user interface (GUI). In one example, the method 300 may be performed by the test generation tool illustrated in FIG. 1 or by a computer as illustrated in FIG. 4 and discussed below. For simplicity of explanation, the method 300 is described as being performed by a processor, which could be a component of the test generation tool or the computer.

[0018] At block 302, the method 300 begins. At block 304, the processor obtains a text file. The text file comprises a file into which a markup language page, such as an HTML page, has been exported. The markup language page may comprise code written in alternative or additional programming languages, including JavaScript. The markup page comprises a web browser-based GUI. The processor may export the markup language page into the text file, or the processor may receive the text file from another device.

[0019] At block 306, the processor obtains metadata associated with graphical control elements of the GUI. In one example, the metadata is associated with custom-written graphical control elements created specifically for the GUI, e.g., graphical control elements programmed using the JavaScript programming language. This type of metadata may map specific attributes of the graphical control elements, such as tags, element identifiers, or classes, to functions associated with browser actions, such as button clicks or text inputs.

[0020] At block 308, the processor parses the text file for markup language elements associated with graphical control elements. For example, the processor may parse the text file for basic HTML tags such as <href>, <select>, or <input>, which indicate the presence of known HTML elements. The processor may also parse the text file for other types of HTML elements, such as tables. In one example, the processor uses the metadata obtained in block 306 to parse the text file for HTML elements associated with custom-written graphical control elements that might otherwise not be directly recognizable.

[0021] At block 310, the processor generates a set of test cases for exercising the GUI. In one example, the set of test cases involves a set of functions, or possibly sub-routines, that manipulate the graphical control elements of the GUI. In one example, the functions are uniquely named for the types of tags or other markup language elements associated with the specific graphical control elements that the functions are intended to manipulate. For instance, a “select” function may make a selection from a drop down list. In a further example, the set of test cases may include a generic function that allows an operation on a markup language element. For instance, a “table-select” function may select contents from a particular row and column of a table, or a “find-table” function may locate a row and column of a table containing the contents of a text search string.

[0022] In another example, the functions may include browser actions associated with custom-written graphical control elements. As discussed above, metadata may map specific attributes of a custom-written graphical control element to a function associated with a browser action. For instance, the element identifier “main-menu” may be mapped by the metadata to generate an action resulting in a “main_menu_wait_for_element_and_click” function that takes a string argument. When programmed into a test case, this function would attempt to locate and click on an element matching the string argument in the “main-menu” drop down menu box.

[0023] In one example, generating the set of test cases involves updating an existing set of test cases to reflect changes to the GUI. In another example, generating the set of test cases involves generating a new set of test cases.

[0024] At block 312, the processor outputs the set of test cases. In one example, the set of test cases is directly output to a separate tool that controls the web browser, for instance by causing the web browser to implement actions such as button clicks. For instance, the set of test cases may be imported by a test script.

[0025] The method 300 ends in block 314.

[0026] In some examples, the processor that performs the method 200 or the method 300 is part of a test generation tool that includes a browser debugger. In this case, the processor may open the markup language page itself and parse the contents, rather than rely on another device to export the markup language page to a text file. In this case, the processor may also open and parse any sub-pages associated with the markup language page (provided that any necessary mapping of page generator elements is known to the processor). The processor may be configured in this case to traverse an entire tree of linked markup language pages, e.g., by clicking and opening <href> tags and then parsing the content of the opened pages in the JavaScript document object model (DOM) programming interface.

[0027] As a result, the examples of the present disclosure improve the functioning of a computer. For example, a computer may experience better operation of web browser-based GUIs through automated testing that is more accurate and more thorough than the testing that is possible without the improvements provided by the present disclosure. In other words, the technological art of web browser-based GUIs is improved by providing a computer that is modified with the ability to automatically and dynamically test the GUIs, as disclosed by the present disclosure.

[0028] It should be noted that although not explicitly specified, one or more blocks, functions, or operations of the method 200 or the method 300 described above may include a storing, displaying and/or outputting step as required for a particular application. In other words, any data, records, fields, and/or intermediate results discussed in the methods can be stored, displayed, and/or outputted to another device as required for a particular application. Furthermore, blocks, functions, or operations in FIG. 2 and FIG. 3 that recite a

determining operation, or involve a decision, do not necessarily require that both branches of the determining operation be practiced. In other words, one of the branches of the determining operation can be deemed as an optional step.

[0029] FIG. 4 depicts an example high-level block diagram of a computer that can be transformed into a machine that is dedicated to perform the functions described herein. Notably, no computer or machine currently exists that performs the functions as described herein. As a result, the examples of the present disclosure improve the operation and functioning of the general-purpose computer to automate test generation for a web browser-based user interface application, as disclosed herein.

[0030] As depicted in FIG. 4, the computer 400 comprises a hardware processor element 402, e.g., a central processing unit (CPU), a microprocessor, or a multi-core processor, a memory 404, e.g., random access memory (RAM) and/or read only memory (ROM), a module 405 for generating automated tests of web browser-based GUIs, and various input/output devices 406, e.g., storage devices, including but not limited to, a tape drive, a floppy drive, a hard disk drive or a compact disk drive, a receiver, a transmitter, a speaker, a display, a speech synthesizer, an output port, an input port and a user input device, such as a keyboard, a keypad, a mouse, a microphone, and the like. Although only one processor element is shown, it should be noted that the general-purpose computer may employ a plurality of processor elements. Furthermore, although only one general-purpose computer is shown in the figure, if the method(s) as discussed above is implemented in a distributed or parallel manner for a particular illustrative example, i.e., the blocks of the above method(s) or the entire method(s) are implemented across multiple or parallel general-purpose computers, then the general-purpose computer of this figure is intended to represent each of those multiple general-purpose computers. Furthermore, one or more hardware processors can be utilized in supporting a virtualized or shared computing environment. The virtualized computing environment may support one or more virtual machines representing computers, servers, or other computing devices. In such virtualized virtual machines, hardware components

such as hardware processors and computer-readable storage devices may be virtualized or logically represented.

[0031] It should be noted that the present disclosure can be implemented by machine readable instructions and/or in a combination of machine readable instructions and hardware, e.g., using application specific integrated circuits (ASIC), a programmable logic array (PLA), including a field-programmable gate array (FPGA), or a state machine deployed on a hardware device, a general purpose computer or any other hardware equivalents, e.g., computer readable instructions pertaining to the method(s) discussed above can be used to configure a hardware processor to perform the blocks, functions and/or operations of the above disclosed methods. In one example, instructions and data for the present module or process 405 for generating automated tests of web browser-based GUIs, e.g., machine readable instructions can be loaded into memory 404 and executed by hardware processor element 402 to implement the blocks, functions or operations as discussed above in connection with the exemplary method 200 or method 300. Furthermore, when a hardware processor executes instructions to perform “operations”, this could include the hardware processor performing the operations directly and/or facilitating, directing, or cooperating with another hardware device or component, e.g., a co-processor and the like, to perform the operations

[0032] In one example, instructions and data for the present module or process 405 for generating tests for web browser-based user interface application, e.g., machine readable instructions can be loaded into memory 404 and executed by hardware processor element 402 to implement the blocks, functions or operations as discussed above in connection with the exemplary method 200 or method 300. For instance, the module 405 may include a plurality of programming code components, including a markup language page exporter component 408, a text parser component 410, and a case generator component 412. These programming code components may be included in a computing system that includes an automated test generation tool, such as the system 100.

[0033] The markup language page exporter component 406 may be configured to export a markup language page, such as an HTML page or a JavaScript page including a GUI, into a text file that is parsable by the text parser component 410.

[0034] The text parser component 410 may be configured to parse a text file into which a markup language page has been exported for markup language elements associated with graphical control elements, including markup language elements associated with custom-written graphical control elements.

[0035] The case generator component 412 may be configured to generate a set of test cases for exercising the GUI. In one example, the set of test cases involves a set of functions, or possibly sub-routines, that manipulate the graphical control elements of the GUI.

[0036] Furthermore, when a hardware processor executes instructions to perform "operations", this could include the hardware processor performing the operations directly and/or facilitating, directing, or cooperating with another hardware device or component, e.g., a co-processor and the like, to perform the operations.

[0037] The processor executing the machine readable instructions relating to the above described method(s) can be perceived as a programmed processor or a specialized processor. As such, the present module 405 for generating automated tests of web browser-based GUIs, including associated data structures, of the present disclosure can be stored on a tangible or physical (broadly non-transitory) computer-readable storage device or medium, e.g., volatile memory, non-volatile memory, ROM memory, RAM memory, magnetic or optical drive, device or diskette and the like. More specifically, the computer-readable storage device may comprise any physical devices that provide the ability to store information such as data and/or instructions to be accessed by a processor or a computing device such as a computer or an application server.

[0038] It will be appreciated that variants of the above-disclosed and other features and functions, or alternatives thereof, may be combined into many other different systems or applications. Various presently unforeseen or unanticipated alternatives, modifications, variations, or improvements therein

may be subsequently made by those skilled in the art which are also intended to be encompassed by the following claims.

What is claimed is:

1. A method, comprising:
 - obtaining, by a processor, a text file into which a markup language page has been exported, wherein the markup language page comprises a web browser-based graphical user interface;
 - parsing, by the processor, the text file for a markup language element associated with a graphical control element of the graphical user interface; and
 - generating, by the processor, a test case for exercising the graphical user interface, wherein the test case comprises a function that manipulates the graphical control element.
2. The method of claim 1, wherein the graphical control element is a component of the web-browser based graphical user interface that interacts with a user.
3. The method of claim 1, wherein the markup language element indicates a presence of a table.
4. The method of claim 1, wherein the markup language element is a hypertext markup language tag.
5. The method of claim 1, wherein the graphical control element is a custom-written graphical control element created specifically for the web browser-based graphical user interface.
6. The method of claim 5, further comprising:
 - obtaining, by the processor, metadata associated with the graphical control element, prior to the parsing, wherein the metadata maps an attribute of the graphical control element to a function associated with an action of a web browser,
 - wherein the parsing includes parsing the text file for the attribute of the

graphical control element that is mapped to the function associated with the action of the web browser.

7. The method of claim 1, wherein the function is named for the markup language element.

8. The method of claim 1, further comprising:
outputting, by the processor, the test case to a tool that controls a web browser.

9. The method of claim 1, wherein the parsing comprises:
parsing a sub-page associated with the markup language page.

10. The method of claim 1, wherein the test case is added to an existing set of test cases for the web browser-based graphical user interface.

11. The method of claim 10, wherein the graphical control element is part of an update to the web browser-based graphical user interface.

12. A computer system comprising:
a processor;
a storage coupled to the processor;
a user interface device to receive input from a user and present information to the user in human perceptible form; and
an instruction set to cooperate with the processor and the storage to:
obtain a text file into which a markup language page has been exported, wherein the markup language page comprises a web browser-based graphical user interface;
parse the text file for a markup language element associated with a graphical control element of the graphical user interface; and

generate a test case for exercising the graphical user interface, wherein the test case comprises a function that manipulates the graphical control element.

13. The computer system of claim 12, wherein the instruction set further cooperates with the processor and the storage to:

obtain metadata associated with the graphical control element, wherein the metadata maps an attribute of the graphical control element to a function associated with an action of a web browser; and

parse the text file for the attribute of the graphical control element that is mapped to the function associated with the action of the web browser.

14. A non-transitory machine-readable storage medium encoded with instructions executable by a processor, the machine-readable storage medium comprising:

instructions to obtain a text file into which a markup language page has been exported, wherein the markup language page comprises a web browser-based graphical user interface;

instructions to parse the text file for a markup language element associated with a graphical control element of the graphical user interface; and

instructions to generate a test case for exercising the graphical user interface, wherein the test case comprises a function that manipulates the graphical control element.

15. The non-transitory machine-readable storage medium of claim 14, further comprising:

instructions to obtain metadata associated with the graphical control element, wherein the metadata maps an attribute of the graphical control element to a function associated with an action of a web browser; and

instructions to parse the text file for the attribute of the graphical control element that is mapped to the function associated with the action of the web browser.

┌

1/4

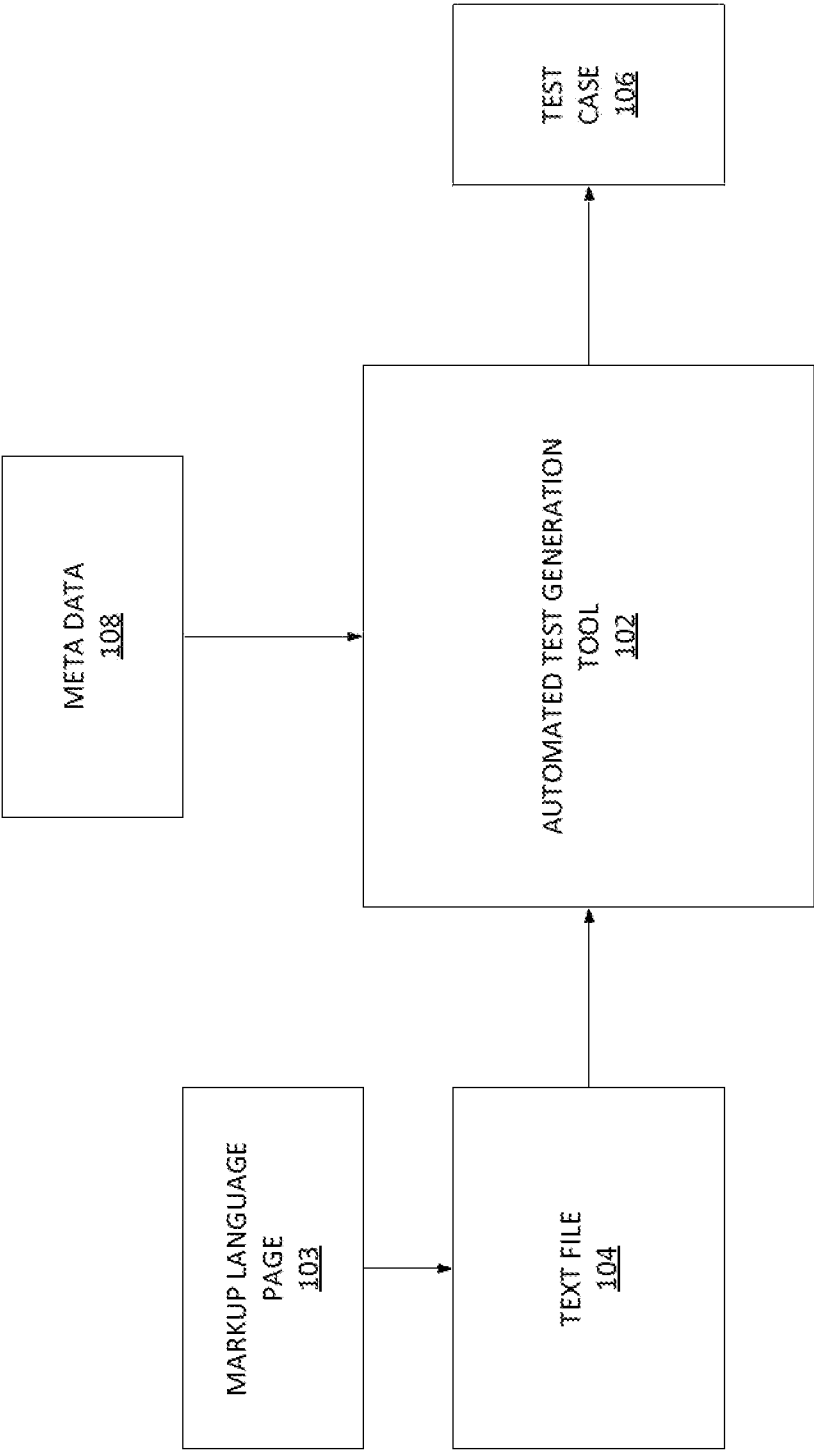


FIG. 1

└

200

2/4

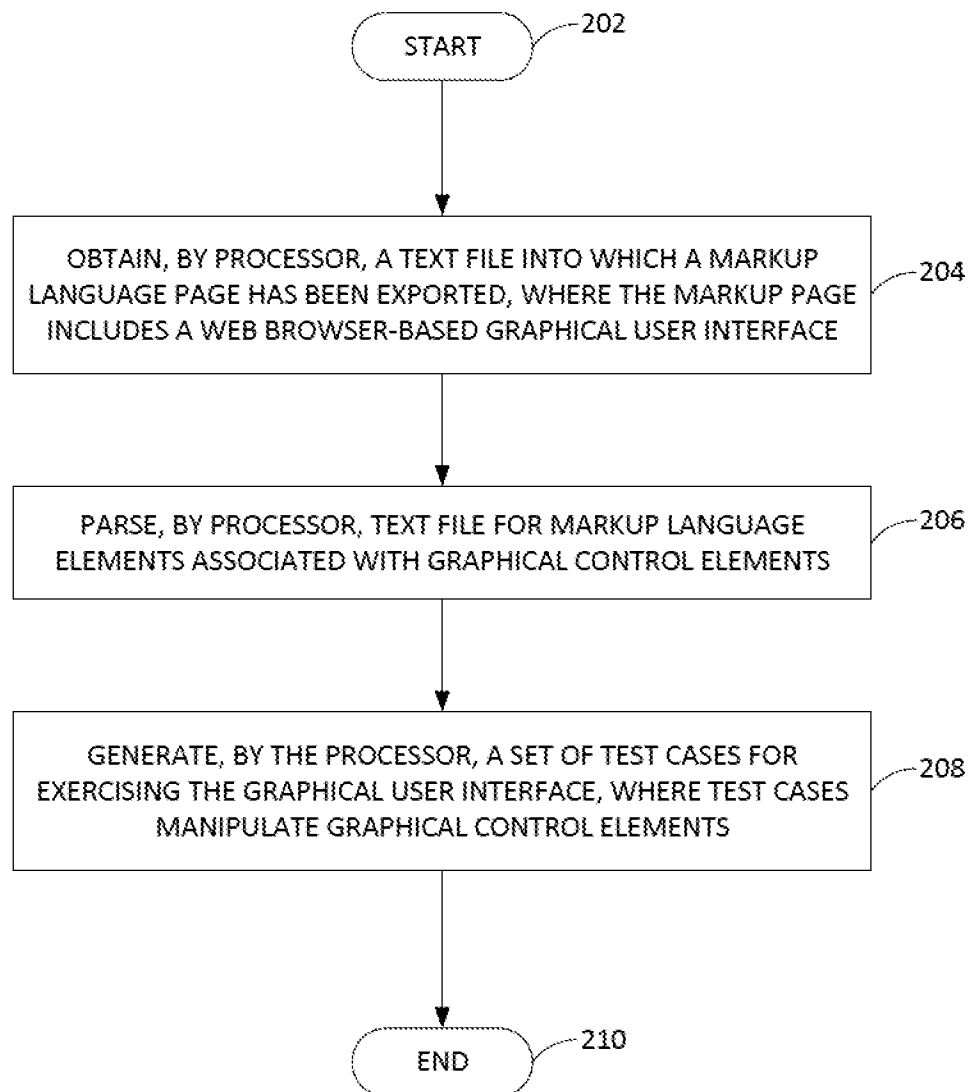


FIG. 2

300

3/4

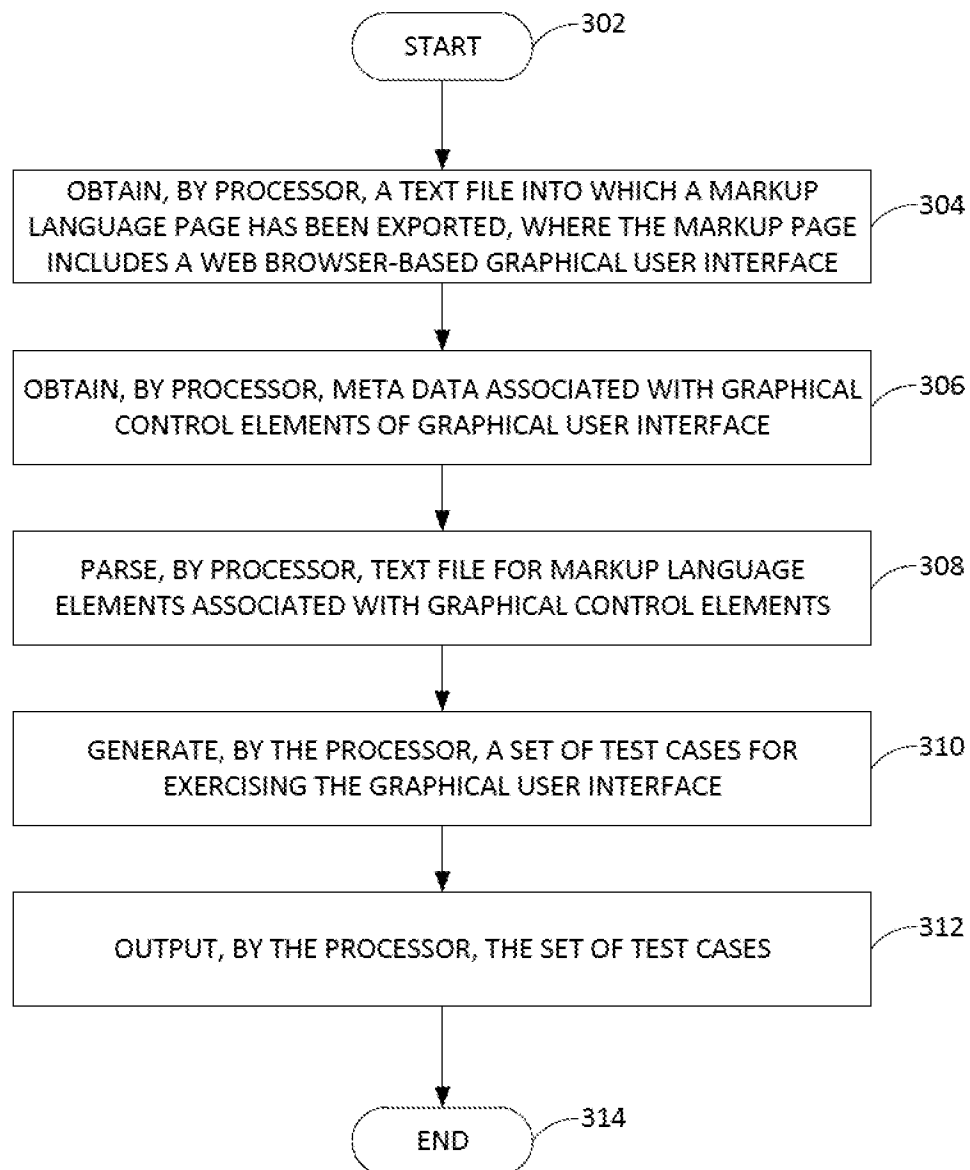


FIG. 3

4/4

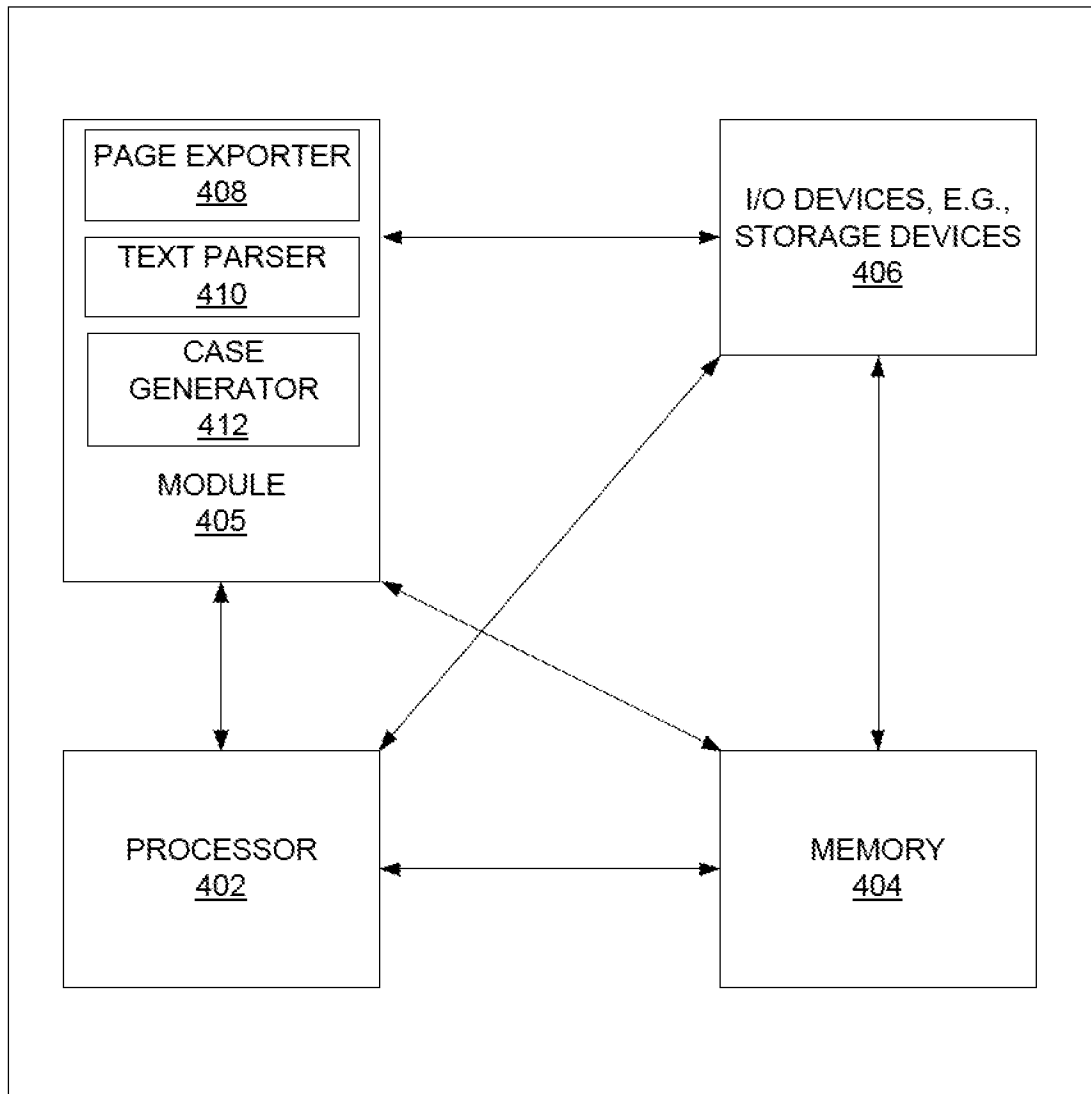
400

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/013422**A. CLASSIFICATION OF SUBJECT MATTER****G06F 11/263(2006.01)i, G06F 3/048(2006.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 11/263; H04L 1/22; G06F 9/44; G06F 11/36; G06F 11/00; G06F 3/048

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords:markup language, test case, graphical user interface, web-browser based, hypertext, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2002-0091968 A1 (DONALD MOREAUX et al.) 11 July 2002 See paragraphs [0014], [0027], [0030]-[0034], and [0053]; claims 1, 9, and 17; and figures 2-3 and 5.	1-15
Y	US 2013-0346948 A1 (YAN ZHANG) 26 December 2013 See paragraphs [0014], [0019], [0025], [0038], [0040], and [0043]; and figures 1 and 9-10.	1-15
A	US 8,060,791 B2 (WESLEY D. HAYUTIN) 15 November 2011 See column 4, lines 27-45; column 4, line 56 - column 5, line 3; column 5, lines 41-55; column 6, lines 34-43; and figures 1 and 3.	1-15
A	US 2015-0007149 A1 (CA, INC.) 01 January 2015 See paragraphs [0024]-[0026], [0028], [0032]-[0033], and [0042]-[0043]; and figures 1-3.	1-15
A	WO 02-23336 A1 (BEA SYSTEMS, INC.) 21 March 2002 See paragraphs [0112], [0115], and [0119]-[0120]; and figures 13 and 24.	1-15



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

17 September 2015 (17.09.2015)

Date of mailing of the international search report

18 September 2015 (18.09.2015)

Name and mailing address of the ISA/KR

International Application Division
Korean Intellectual Property Office
189 Cheongsu-ro, Seo-gu, Daejeon Metropolitan City, 35208,
Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

NHO, Ji Myong

Telephone No. +82-42-481-8528



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/013422

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2002-0091968 A1	11/07/2002	None	
US 2013-0346948 A1	26/12/2013	CN 103502952 A EP 2684130 A1 US 9104810 B2 WO 2012-119267 A1	08/01/2014 15/01/2014 11/08/2015 13/09/2012
US 8060791 B2	15/11/2011	US 2009-0210749 A1 US 2011-0035630 A1 US 7840851 B2	20/08/2009 10/02/2011 23/11/2010
US 2015-0007149 A1	01/01/2015	US 2013-042222 A1 US 8893087 B2	14/02/2013 18/11/2014
WO 02-23336 A1	21/03/2002	AU 2001-94555 A1 US 2002-0085020 A1 US 2006-0015847 A1 US 6990654 B2 US 7571426 B2	26/03/2002 04/07/2002 19/01/2006 24/01/2006 04/08/2009