

(51) International Patent Classification:
G06F 11/36 (2006.01)(21) International Application Number:
PCT/CN2015/098840(22) International Filing Date:
25 December 2015 (25.12.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: INTEL CORPORATION [US/US]; 2200 Mission College Blvd., Santa Clara, California 95054 (US).

(72) Inventors; and

(71) Applicants (for BZ only): DENG, Tao [CN/CN]; 880 Zixing Road, Shanghai, 200241 (CN). SHEN, Haihao [CN/CN]; No. 880, Zixing Road, Shanghai, 200241 (CN). WANG, Yun [CN/CN]; NO.880 Zixing Road, Shanghai, 200241 (CN). GONG, Jiong [CN/CN]; 880 Zixing Road, Shanghai, 200241 (CN). HUANG, Zao [CN/CN]; 880 Zixing Road, Shanghai, 200241 (CN).

(74) Agent: NTD PATENT AND TRADEMARK AGENCY LIMITED; 10th Floor, Block A, Investment Plaza, 27 Jinrongdajie, Xicheng District, Beijing 100033 (CN).

(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).

Published:

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

(54) Title: TECHNOLOGIES FOR AUTOMATED APPLICATION TESTING COVERAGE AND RESULTS EVALUATION

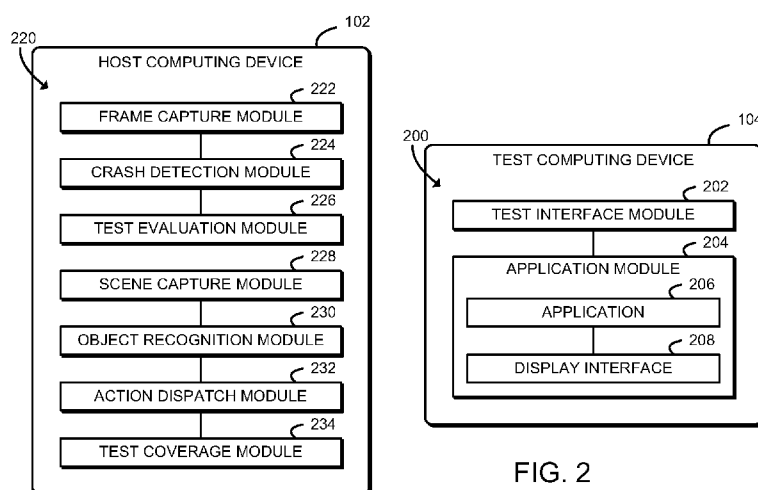


FIG. 2

(57) Abstract: Technologies for application testing include a host computing device (102) and a test computing device (104, 104a, 104b). The host computing device (102) captures a frame of a display interface generated by an application executed by the test computing device (104, 104a, 104b). The host computing device (102) may detect a crash indication as a function of the captured frame. In response to detecting the crash indication, the host computing device (102) may detect an application crash based on an error log and in response report a test failure. The host computing device (102) may capture a scene of the display interface, recognize one or more user interface objects of the scene, select a test action based on the scene, and perform the test action. The test action and a user interface object may be selected probabilistically. The host computing device (102) records test coverage data in response to performing the test action.

TECHNOLOGIES FOR AUTOMATED APPLICATION TESTING COVERAGE AND RESULTS EVALUATION

BACKGROUND

[0001] Currently, a wide number and variety of applications (“apps”) are available for many different computing platforms. However, as the number of applications and the number of computing devices that may execute those applications increase, application validation and testing becomes an increasingly difficult problem. Application validation and testing may require a tester to design a test case that mimics real-world human interaction with an application.

[0002] Typical application testing systems evaluate test results by analyzing application log files produced during a test run. However, many application test failures are caused by and/or result in a crash dialog box, a blanked screen, a flash quit, or an otherwise unresponsive application. Application testing systems using log-file-based results checking may not detect crashes and other dynamic errors that are not included in application log files.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] The concepts described herein are illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. Where considered appropriate, reference labels have been repeated among the figures to indicate corresponding or analogous elements.

[0004] FIG. 1 is a simplified block diagram of at least one embodiment of a system for device-independent automated application testing;

[0005] FIG. 2 is a simplified block diagram of at least one embodiment of various environments that may be established by the system of FIG. 1;

[0006] FIG. 3 is a simplified flow diagram of at least one embodiment of a method for automated test results checking that may be executed by a computing device of FIGS. 1 and 2;

[0007] FIG. 4 is a simplified flow diagram of at least one embodiment of a method for automated detection of crash dialogs that may be executed by a computing device of FIGS. 1 and 2;

[0008] FIG. 5 is a simplified flow diagram of at least one embodiment of a method for automated test coverage that may be executed by the computing device of FIGS. 1 and 2;

[0009] FIG. 6 is pseudocode illustrating at least one potential embodiment of configuration data that may be used during execution of the method of FIG. 5; and

[0010] FIG. 7 is pseudocode illustrating at least one potential embodiment of test coverage data that may be produced during execution of the method of FIG. 5.

DETAILED DESCRIPTION OF THE DRAWINGS

[0011] While the concepts of the present disclosure are susceptible to various modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and will be described herein in detail. It should be understood, however, that there is no intent to limit the concepts of the present disclosure to the particular forms disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives consistent with the present disclosure and the appended claims.

[0012] References in the specification to “one embodiment,” “an embodiment,” “an illustrative embodiment,” etc., indicate that the embodiment described may include a particular feature, structure, or characteristic, but every embodiment may or may not necessarily include that particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described. Additionally, it should be appreciated that items included in a list in the form of “at least one of A, B, and C” can mean (A); (B); (C); (A and B); (A and C); (B and C); or (A, B, and C). Similarly, items listed in the form of “at least one of A, B, or C” can mean (A); (B); (C); (A and B); (A and C); (B and C); or (A, B, and C).

[0013] The disclosed embodiments may be implemented, in some cases, in hardware, firmware, software, or any combination thereof. The disclosed embodiments may also be implemented as instructions carried by or stored on one or more transitory or non-transitory machine-readable (e.g., computer-readable) storage media, which may be read and executed by one or more processors. A machine-readable storage medium may be embodied as any storage device, mechanism, or other physical structure for storing or transmitting information in a form readable by a machine (e.g., a volatile or non-volatile memory, a media disc, or other media device).

[0014] In the drawings, some structural or method features may be shown in specific arrangements and/or orderings. However, it should be appreciated that such specific arrangements and/or orderings may not be required. Rather, in some embodiments, such features may be arranged in a different manner and/or order than shown in the illustrative figures. Additionally, the inclusion of a structural or method feature in a particular figure is not meant to imply that such feature is required in all embodiments and, in some embodiments, may not be included or may be combined with other features.

[0015] Referring now to FIG. 1, in an illustrative embodiment, a system 100 for device-independent application testing includes a host computing device 102 and one or more test computing devices 104, which may be in communication over a network 106. The illustrative system 100 includes two test computing devices 104a, 104b; however it should be understood that in some embodiments the system 100 may include a different number of test computing devices 104. In use, as described in more detail below, the host computing device 102 connects to the test computing device 104a and starts an application test session. In some embodiments, the host computing device 102 may start an application test session with a test computing device 104. During the test session, the host computing device 102 captures frames of a display interface generated by the application of the test computing device 104. The host computing device 102 processes the frames to detect crash indications such as crash dialogs, blank screens, and/or application “flash quits.” The host computing device 102 may double-check detected crash indications against traditional application logs to determine whether the test session has failed. Thus, the host computing device 102 may detect many common types of application crashes that may not be detected through log file analysis alone. Additionally, by performing checks against both the captured frame and the log file, the host computing device 102 may prevent false positive crash detections.

[0016] Additionally or alternatively, in some embodiments the host computing device 102 may capture the display interface of the application and then probabilistically select a test action type and a user interface object on which to perform the test action. The host computing device 102 may perform a depth-first probabilistic exploration of the application. The host computing device 102 records test coverage while performing the probabilistic exploration. Thus, the host computing device 102 may allow for automatic exploration and testing of many applications with high test coverage.

[0017] The host computing device 102 may be embodied as any type of computation or computer device capable of performing the functions described herein, including, without limitation, a computer, a desktop computer, a workstation, a laptop computer, a notebook computer, a tablet computer, a mobile computing device, a wearable computing device, a network appliance, a web appliance, a distributed computing system, a processor-based system, and/or a consumer electronic device. As shown in FIG. 1, the host computing device 102 illustratively includes a processor 120, an input/output subsystem 122, a memory 124, a data storage device 126, and communication circuitry 128. Of course, the host computing device 102 may include other or additional components, such as those commonly found in a desktop computer (e.g., various input/output devices), in other embodiments. Additionally, in some embodiments, one or more of the illustrative components may be incorporated in, or otherwise form a portion of, another component. For example, the memory 124, or portions thereof, may be incorporated in the processor 120 in some embodiments.

[0018] The processor 120 may be embodied as any type of processor capable of performing the functions described herein. The processor 120 may be embodied as a single or multi-core processor(s), digital signal processor, microcontroller, or other processor or processing/controlling circuit. Similarly, the memory 124 may be embodied as any type of volatile or non-volatile memory or data storage capable of performing the functions described herein. In operation, the memory 124 may store various data and software used during operation of the host computing device 102 such as operating systems, applications, programs, libraries, and drivers. The memory 124 is communicatively coupled to the processor 120 via the I/O subsystem 122, which may be embodied as circuitry and/or components to facilitate input/output operations with the processor 120, the memory 124, and other components of the host computing device 102. For example, the I/O subsystem 122 may be embodied as, or otherwise include, memory controller hubs, input/output control hubs, firmware devices, communication links (i.e., point-to-point links, bus links, wires, cables, light guides, printed circuit board traces, etc.) and/or other components and subsystems to facilitate the input/output operations. In some embodiments, the I/O subsystem 122 may form a portion of a system-on-a-chip (SoC) and be incorporated, along with the processors 120, the memory 124, and other components of the host computing device 102, on a single integrated circuit chip.

[0019] The data storage device 126 may be embodied as any type of device or devices configured for short-term or long-term storage of data such as, for example, memory devices and circuits, memory cards, hard disk drives, solid-state drives, or other data storage devices. The communication circuitry 128 of the host computing device 102 may be embodied as any communication circuit, device, or collection thereof, capable of enabling communications between the host computing device 102, the test computing devices 104, and/or other remote devices either directly or over the network 106. The communication circuitry 128 may be configured to use any one or more communication technology (e.g., wired or wireless communications) and associated protocols (e.g., direct serial communication, USB communication, Ethernet, Bluetooth®, Wi-Fi®, WiMAX, etc.) to effect such communication.

[0020] Additionally, the host computing device 102 may also include a display 130 and a camera 132. The display 130 may be embodied as any type of display capable of displaying digital information such as a liquid crystal display (LCD), a light emitting diode (LED), a plasma display, a cathode ray tube (CRT), or other type of display device. As described below, the display 130 may be used to display a graphical user interface or other information to the user of the host computing device 102. Additionally, in some embodiments, the host computing device 102 may include a touch screen coupled to the display 130. The touch screen may be used to record user input that is similar to user input of the test computing device 104, as described further below.

[0021] The camera 132 may be embodied as a digital camera or other digital imaging device integrated with the host computing device 102 or otherwise communicatively coupled thereto. The camera 132 includes an electronic image sensor, such as an active-pixel sensor (APS), e.g., a complementary metal-oxide-semiconductor (CMOS) sensor, or a charge-coupled device (CCD). The camera 132 may be used to capture images of the user interface presented by one or more of the test computing devices 104 including, in some embodiments, capturing still images or video images.

[0022] Each of the test computing devices 104 is configured to execute an application under test and, in some embodiments, provide data on user interactions and the interface of the application to the host computing device 102 and/or respond to commands initiated by the host computing device 102. Each test computing device 104 may be embodied as any type of computation or computer device capable of performing the functions described herein, including,

without limitation, a mobile computing device, a smartphone, a tablet computer, a wearable computing device, a computer, a laptop computer, a desktop computer, multiprocessor system, a server, a rack-mounted server, a blade server, a network appliance, a web appliance, a distributed computing system, a processor-based system, and/or a consumer electronic device. Each test computing device 104 may include components and devices commonly found in a smartphone or similar computing device, such as a processor 140, an I/O subsystem 142, a memory 144, a data storage device 146, communication circuitry 148, a display 150, and/or other peripheral devices. Those individual components of the test computing device 104 may be similar to the corresponding components of the host computing device 102, the description of which is applicable to the corresponding components of the test computing device 104 and is not repeated herein so as not to obscure the present disclosure.

[0023] Additionally, in some embodiments, each test computing device 104 may include a touch screen 152. The touch screen 152 may be embodied as any type of touch screen capable of generating input data in response to being touched by the user of the test computing device 104. The touch screen 152 may be embodied as, for example, a resistive touch screen, a capacitive touch screen, or a camera-based touch screen.

[0024] As discussed in more detail below, the host computing device 102 and the test computing devices 104 may be configured to transmit and receive data with each other and/or other devices of the system 100 over the network 106. The network 106 may be embodied as any number of various wired and/or wireless networks. For example, the network 106 may be embodied as, or otherwise include, a wired or wireless local area network (LAN), a wired or wireless wide area network (WAN), a cellular network, and/or a publicly-accessible, global network such as the Internet. As such, the network 106 may include any number of additional devices, such as additional computers, routers, and switches, to facilitate communications among the devices of the system 100. Additionally or alternatively, the host computing device 102 may communicate directly with one or more test computing devices 104, for example over a direct serial connection, direct USB connection, direct wireless connection, or other direct connection.

[0025] Although illustrated as including separate test computing devices 104, it should be understood that in some embodiments, the functions of one or more of the test computing device 104 may be performed by the host computing device 102. For example, the host computing

device 102 may execute a platform simulator associated with one or more test computing devices 104.

[0026] Referring now to FIG. 2, in the illustrative embodiment, a test computing device 104 establishes an environment 200 during operation. The illustrative environment 200 includes a test interface module 202 and an application module 204. The various modules of the environment 200 may be embodied as hardware, firmware, software, or a combination thereof. For example the various modules, logic, and other components of the environment 200 may form a portion of, or otherwise be established by, the processor 140 or other hardware components of the test computing device 104. As such, in some embodiments, any one or more of the modules of the environment 200 may be embodied as a circuit or collection of electrical devices (e.g., a test interface circuit or an application circuit, etc.).

[0027] The test interface module 202 is configured to communicate with the host computing device 102 during an application test session. For example, the test interface module 202 may be configured to receive commands from the host computing device 102 to start or stop an application test record session or an application test playback session, and the test interface module 202 may be configured to receive commands from the host computing device 102 corresponding to requested user interface actions. The test interface module 202 may also be configured to transmit information to the host computing device 102, such as user interface event data or display interface data.

[0028] The application module 204 is configured to execute an application 206 during an application test session. In some embodiments, the application module 204 may be configured to control the application 206, for example by issuing synthetic user interface events to the application 206. The application 206 may be embodied as computer program executed by the test computing device 104 such as a native application, a web application, a bytecode application, or any other executable application. The particular format, underlying operating system or application toolkit, or other characteristics of the application 206 may depend on the particular test computing device 104 that executes the application 206. During execution, the application 206 creates and/or manages a display interface 208, which may be displayed on the display 150 of the test computing device 104. The display interface 208 may be embodied as any graphical user interface, and may include multiple user interface objects, such as buttons, menu items, text labels, images, or other user interface controls. The size, layout, appearance, language, and other

characteristics of the display interface 208 may also depend on the particular test computing device 104 that executes the application 206.

[0029] Still referring to FIG. 2, in an illustrative embodiment, the host computing device 102 establishes an environment 220 during operation. The illustrative environment 220 includes a frame capture module 222, a crash detection module 224, a test evaluation module 226, a scene capture module 228, an object recognition module 230, an action dispatch module 232, and a test coverage module 234. The various modules of the environment 220 may be embodied as hardware, firmware, software, or a combination thereof. As such, in some embodiments, one or more of the modules of the environment 200 may be embodied as circuitry or collection of electrical devices (e.g., frame capture circuitry 222, crash detection circuitry 224, test evaluation circuitry 226, scene capture circuitry 228, object recognition circuitry 230, action dispatch circuitry 232, and/or test coverage circuitry 234). It should be appreciated that, in such embodiments, one or more of the frame capture circuitry 222, the crash detection circuitry 224, the test evaluation circuitry 226, the scene capture circuitry 228, the object recognition circuitry 230, the action dispatch circuitry 232, and/or the test coverage circuitry 234 may form a portion of one or more of the processor 120, the I/O subsystem 122, and/or other components of the host computing device 102. Additionally, in some embodiments, one or more of the illustrative modules may form a portion of another module and/or one or more of the illustrative modules may be independent of one another.

[0030] The frame capture module 222 is configured to capture a frame of the display interface 208 generated by the application 206 executed by the test computing device 104. The frame capture module 222 may be configured to capture a key frame in response to an application event generated by the application 206, or in response to expiration of a timer.

[0031] The crash detection module 224 is configured to detect a crash indication as a function of the captured frame of the display interface 208. The crash indication may be, for example, a crash dialog box, a blank frame, an application flash crash, or any other visual indication that the application 206 has crashed. To detect a crash dialog box, the crash detection module 224 may be configured to detect a rectangle in the frame of the display interface 208 and perform optical character recognition of the rectangle to generate text contents of the rectangle. The crash detection module 224 may be further configured to compare the text contents to a list of known crash words to determine a number of instances of crash words in the text contents.

The crash detection module 224 may be configured to determine whether the number of instances of crash words exceeds a predetermined threshold number of crash words, such as one crash word.

[0032] The test evaluation module 226 is configured to detect an application crash based on an error log of the application in response to detection of the crash indication and report a test failure in response to detection of the application crash. Reporting the test failure may include storing the frame of the display interface 208.

[0033] The scene capture module 228 is configured to capture a scene of the display interface 208 generated by the application 206 executed by the test computing device 104. The object recognition module 230 is configured to recognize one or more user interface objects of the scene of the display interface 208. Recognizing the one or more user interface objects may include determining a hash value corresponding to each of the one or more user interface objects, or a hash value corresponding to the scene of the display interface 208.

[0034] The action dispatch module 232 is configured to select a test action based on the scene of the display interface 208 and perform the test action with the scene of the display interface 208 in response to recognizing the one or more user interface objects. Selecting the test action may include recognizing a scene type and selecting the test action based on the scene type or probabilistically selecting a test exploration action, such as a click action or a swipe action. In some embodiments, the action dispatch module 232 may be configured to increase a probability value associated with a critical path object of the one or more user interface objects. The critical path object may be included in a predefined test script.

[0035] The test coverage module 234 is configured to record test coverage data based on the test action, the scene, and the one or more user interface objects in response to performing the test action. In some embodiments, the test coverage module 234 may be further configured to determine whether the scene of the display interface 208 changes in response to performing the test action and record test coverage data indicating a traversal between scenes if the scene of the display interface 208 changes.

[0036] Although illustrated as being established by a separate test computing device 104, it should be understood that in some embodiments part or all of the environment 200 may be established by the host computing device 102. For example, in some embodiments, the test interface module 202 and/or the application module 204 may be established by the host

computing device 102 using a platform simulator associated with one or more test computing devices 104.

[0037] Referring now to FIG. 3, in use, the host computing device 102 may execute a method 300 for automated test results checking. The method 300 begins with block 302, in which the host computing device 102 connects to a test computing device 104. As described above, the host computing device 102 may be directly connected to the test computing device 104 using, for example, a direct serial connection, direct USB connection, or direct wireless connection. In some embodiments, the host computing device 102 may connect to the test computing device 104 using the network 106.

[0038] In block 304, the host computing device 102 starts an application test session for an application 206 executed by the test computing device 104. The host computing device 102 may perform any configuration, initialization, or other operations required to cause the test computing device 104 to execute the application 206. For example, the host computing device 102 may side-load or otherwise provide the test computing device 104 with binary code corresponding with the application 206. In some embodiments, in block 306, the host computing device 102 may cause the test computing device 104 to launch the application 206. For example, the host computing device 102 may send a message or other command to the test computing device 104 to launch the application 206. In some embodiments, the test computing device 104 may launch the application 206 in a special testing mode or including special testing code (e.g., debugging mode, instrumented execution, etc.). Additionally or alternatively, in some embodiments a user may manually launch the application 206 on the test computing device 104. After starting the application test session, the test computing device 104 may execute a test script to control the application 206, allow the user to manually control the application 206, or otherwise test the operation of the application 206.

[0039] In block 308, the host computing device 102 captures a frame from a display interface 208 generated by the application 206 of the test computing device 104. The host computing device 102 may use any technique to capture the frame of the display interface 208. For example, the host computing device 102 and/or the test computing device 104 may capture the frame using screen capture software to record framebuffer data or other image data representing the display interface 208. In some embodiments, the host computing device 102 may capture the frame by recording the contents of the display 150 of the test computing device

104 using the camera 132 of the host computing device 102. In some embodiments, in block 310 the host computing device 102 may capture a key frame of the display interface 208. The key frame may be embodied as any frame of the display interface 208 generated in response to or otherwise associated with an event of the application 206. In some embodiments, in block 312 the host computing device 102 may capture a frame of the display interface 208 if a timer has expired. For example, the host computing device 102 may set a timer to expire after a predetermined time period has elapsed without the capture of a key frame.

[0040] In block 314, the computing device 102 analyzes the frame to detect a crash indication in the display interface 208. The computing device 102 may perform any computer vision, machine learning, or other algorithm to detect the crash indication. In some embodiments, the computing device 102 may analyze the frame in stages, for example performing a quick first pass analysis followed by more lengthy or processor-intensive analysis stages if necessary. The crash indication may include any visual indication that the application 206 has crashed, is frozen or unresponsive, or has otherwise failed to execute correctly. In some embodiments, in block 316 the host computing device 102 may detect the crash indication by detecting a crash dialog box in the frame. One embodiment of a method for detecting a crash dialog box is described further below in connection with FIG. 7. In some embodiments, in block 318 the host computing device 102 may detect the crash indication by detecting a frozen and/or blank screen. For example, the host computing device 102 may detect a frame that has been unchanged for a relatively long period of time, or a frame that displays only a single color (e.g., black, white, or another solid color). In some embodiments, in block 320, the host computing device 102 may detect the crash indication by detecting an application flash crash. That is, the host computing device 102 may detect that the test computing device 104 displays the display interface 208 associated with the application 206 for a short time and then quickly displays a different display interface such as a device home screen, application launcher, or other default interface. An application flash crash may indicate that the application 206 starts and then immediately or shortly thereafter exits, quits, or otherwise stops processing.

[0041] In block 322, the computing device 102 determines whether a crash indication has been detected in the frame of the display interface 208. If not—or, in some embodiments, if it is inconclusive whether a crash indication has been detected—the method 300 loops back to block

308 to continue capturing frames and detecting crash indications. If a crash indication has been detected, the method 300 advances to block 324.

[0042] In block 324, the host computing device 102 double-checks for an application crash against one or more error logs. The error logs may be generated or otherwise maintained by the application 206 or by an application testing framework of the test computing device 104 and/or the host computing device 102. The host computing device 102 may perform any appropriate analysis of the error logs to determine whether an actual application crash has occurred or whether the detected crash indication is a false positive. In block 326, the host computing device 102 determines whether a crash has been detected. If not—or, in some embodiments, if it is inconclusive whether a crash indication has been detected—the method 300 loops back to block 308 to continue capturing frames and detecting crash indications. If a crash has been detected, the method 300 advances to block 328.

[0043] In block 328, the host computing device 102 saves one or more frames of the display interface 208 that include the detected crash indication. The saved frames may be analyzed by a user to verify that a crash has occurred, to determine the nature of the crash, and/or to otherwise review the results of the application test session. In block 330, the host computing device 102 indicates a test failure. The host computing device 102 may use any technique to indicate the failure, such as notifying the user, executing a failure script command, and/or logging the error. After indicating the test failure, the method 300 may be completed. In some embodiments, after indicating the test failure, the method 300 may loop back to block 308, restart the method 300, or otherwise continue testing the application 206.

[0044] Referring now to FIG. 4, in use, the host computing device 102 may execute a method 400 for crash dialog box detection. The method 400 begins with block 402, in which the host computing device 102 quick-detects any rectangles in the captured frame of the display interface 208. The host computing device 102 may perform any computer vision, machine learning, or other algorithm that quickly detects the presence of a rectangle in the frame. For example, the quick detection algorithm may determine whether a rectangle is present in the frame in about 10 milliseconds. Quick rectangle detection may allow the host computing device 102 to efficiently process many frames of the display interface 208. In some embodiments, the host computing device 102 may use one or more heuristics to speed the rectangle detection. For example, the host computing device 102 may restrict its search to the middle of the frame where

crash dialog boxes are typically displayed. In some embodiments, in block 404 the host computing device 102 may detect rectangles including two lines of text, which may be typical for crash dialog boxes.

[0045] In block 406, the host computing device 102 determines whether a rectangle has been detected in the frame. If not, the method 400 branches ahead to block 418, in which the method 400 returns that no crash dialog box has been detected. As described above in connection with FIG. 3, after determining that no crash indication was detected, the host computing device 102 may continue to capture and process frames of the display interface 208.

[0046] Referring back to block 406, if a rectangle has been detected, the method advances to block 408, in which the host computing device 102 performs optical character recognition on the contents of the detected rectangle. The host computing device 102 may produce a text representation of any message included in the detected rectangle, such as a message produced by a crash dialog box. In block 410, the host computing device 102 compares the text contents of the rectangle to a list of words typically associated with application crashes, called crash words. In particular, the host computing device 102 may determine whether any crash words appear in the text contents, or the host computing device 102 may determine the number of crash words that appear in the text contents. In some embodiments, in block 412 the host computing device 102 may compare the number of crash words in the text contents to a predetermined number of crash words.

[0047] In block 414, the host computing device 102 determines whether sufficient crash words are found in the text contents. For example, the host computing device 102 may determine whether more than the predetermined number of crash words (e.g., two or more crash words) are included in the text results. If sufficient crash words are found, the method 400 branches to block 416, in which the method 400 returns that a crash dialog box has been detected. As described above in connection with FIG. 3, after determining that a crash indication was detected, the host computing device 102 may double-check for a crash and/or indicate a test failure. Referring back to block 414, if sufficient crash words are not found, the method 400 branches ahead to block 418, in which the method 400 returns that no crash dialog box has been detected. As described above in connection with FIG. 3, after determining that no crash indication was detected, the host computing device 102 may continue to capture and process

frames of the display interface 208. After returning that a crash dialog box has been detected or that a crash dialog box has not been detected, the method 400 is completed.

[0048] Referring now to FIG. 5, in use, the host computing device 102 may execute a method 500 for automated test coverage. The method 500 begins with block 502, in which the host computing device 102 connects to a test computing device 104. As described above, the host computing device 102 may be directly connected to the test computing device 104 using, for example, a direct serial connection, direct USB connection, or direct wireless connection. In some embodiments, the host computing device 102 may connect to the test computing device 104 using the network 106.

[0049] In block 504, the host computing device 102 starts an application test session for an application 206 executed by the test computing device 104. The host computing device 102 may perform any configuration, initialization, or other operations required to cause the test computing device 104 to execute the application 206. For example, the host computing device 102 may side-load or otherwise provide the test computing device 104 with binary code corresponding with the application 206. In some embodiments, in block 506, the host computing device 102 may cause the test computing device 104 to launch the application 206. For example, the host computing device 102 may send a message or other command to the test computing device 104 to launch the application 206. In some embodiments, the test computing device 104 may launch the application 206 in a special testing mode or including special testing code (e.g., debugging mode, instrumented execution, etc.). Additionally or alternatively, in some embodiments a user may manually launch the application 206 on the test computing device 104.

[0050] In block 508, the host computing device 102 configures one or more random seeds and probabilistic actions. The random seeds may be used to initialize one or more random number generators, pseudorandom number generators, entropy sources, or other sources probabilistic values that may be used by the host computing device 102. In particular, by initializing a pseudorandom number generator with a particular predefined seed, the host computing device 102 may allow for the recording and/or replay of particular test interactions, as described further below. As described further below, the probabilistic actions may be embodied as one or more user interface actions that may be selected and performed by the host computing device 102 based on the current scene of the display interface 208 of the application 206. The host computing device 102 may configure the probabilistic actions, for example, by assigning

one or more predefined probability values to each of the probabilistic actions. In some embodiments, the random seed and/or probability values may be supplied by a user in one or more configuration files.

[0051] Referring now to FIG. 6, pseudocode 600 illustrates one potential embodiment of a configuration file that may be used to configure random seeds and/or probabilistic actions. The illustrative pseudocode 600 is an XML file; however, it should be understood that the host computing device 102 may use configuration files in any appropriate format. The illustrative configuration file includes a Test element for a package named “a.b.c,” which may correspond to a name or other identifier of the application 206. The Test element includes a Seed attribute equal to “100,” which may be used to initialize a pseudorandom number generator. The Test element further includes three Action elements, each of which includes a Type attribute and a Value attribute. The Type attribute corresponds to the type of user interface action that may be selected by the host computing device 102, as described further below. Illustratively, the pseudocode 600 includes three action types: ACCOUNT, CLICK, and SWIPE. The Value attribute corresponds to the probability value that may be assigned to each of the action types. As shown, the CLICK action has a probability value of 90% and the SWIPE action has a probability value of 10%. As described further below, the ACCOUNT action is not selected probabilistically and thus includes Value attribute of “YES.” The ACCOUNT action further includes an Account element including a Name attribute and a Password attribute, which may be used by the host computing device 102 to log into an account while testing the application 206, as described further below. The CLICK action further includes two Click elements. Each Click element includes a Type attribute and a Value attribute. The Type of the Click element corresponds to the type of click, tap, button press, or similar user interface action that may be selected by the host computing device 102. Potential Type values may include PLAIN and HOME, as shown, as well as BACK, MENU, or other interface actions. The Value of the Click element corresponds to the probability value that may be assigned to each of the click types. As shown, the PLAIN click action has a probability value of 80% and the HOME click action has a probability value of 20%.

[0052] Referring back to FIG. 5, in block 510 the host computing device 102 captures an image of a display interface 208 generated by the application 206 of the test computing device 104. The host computing device 102 may use any technique to capture the display interface 208.

For example, the host computing device 102 and/or the test computing device 104 may capture the image using screen capture software to record framebuffer data or other image data representing the display interface 208. In some embodiments, the host computing device 102 may capture the image by recording the contents of the display 150 of the test computing device 104 using the camera 132 of the host computing device 102.

[0053] In block 512, the host computing device 102 selects a test action type based on a scene of the display interface 208. The scene includes the identity, arrangement, and other characteristics of the user interface objects of the display interface 208, and may be determined by the host computing device 102 based on the captured image of the display interface 208. The scene may be based on object recognition and thus may not include variable or transient features of the image such as transition animations, blinking cursors, or similar features. As described further below, the host computing device 102 may select an advanced action type that performs a particular function on the scene, or the host computing device 102 may probabilistically select an action type to explore the scene and provide complete test coverage of the application 206.

[0054] In block 514, the host computing device 102 selects the test action type based on scene recognition. The host computing device 102 may perform any computer vision, artificial intelligence, or other machine learning technique to identify the scene represented by the image of the display interface 208. In particular, the host computing device 102 may be trained to identify scenes that are common to many applications 206, such as account login screens, application download confirmation screens, or other common user interface screens. As described further below, the selected test action may include one or more specialized user interface commands or other data relating to the identified scene.

[0055] In block 516, the host computing device 102 selects the test action type based on probabilistic exploration of the scene. The host computing device 102 may select the test action based on probabilistic exploration, for example, if the scene of the display interface 208 is not recognized. The host computing device 102 may randomly select the test action type from one or more probabilistic exploration action types, including click and/or swipe actions. The host computing device 102 may perform a weighted random selection of the test action type based on probability values assigned by the user to the action types.

[0056] In block 518, the host computing device 102 recognizes user interface objects in the display interface 208. The user interface objects may include any user interface element

presented by the application 206, including buttons, menus, and other user interface controls, text, images including icons and/or recognizable images such as animals, and other user interface objects. The host computing device 102 may use any computer vision, artificial intelligence, or other machine learning language to recognize the user interface objects. In some embodiments, in block 520, the host computing device 102 may generate a hash value based on each recognized object. The hash value may be embodied as, for example, a unified hash code. The hash value may be used by the host computing device 102 to identify the particular objects included in the scene. In some embodiments, in block 522 the host computing device 102 may generate a hash value based on the entire scene including the recognized objects. The hash value may be used by the host computing device 102 to identify the scene and/or to determine whether the scene has changed.

[0057] In block 524, the host computing device 102 dispatches the selected test action. The host computing device 102 may use any technique to cause the application 206 to perform the selected test action. For example, the host computing device 102 may cause the test computing device 104 to generate a synthetic user interface event, for example by transmitting a command to the test computing device 104. Additionally or alternatively, the host computing device 102 may perform the test action by operating the test computing device 104 using a robotic actuator. For example, the host computing device 102 may touch or swipe a user interface object on the touch screen 152 of the test computing device 104 using a robotic finger.

[0058] In some embodiments, in block 526, the host computing device 102 may perform an advanced action based on the recognized scene of the display interface 208. The advanced action may include one or more other user interface actions, including clicks, swipes, text input, and other interactions. For example, for an account action associated with an account login screen, the host computing device 102 may dispatch actions to supply a predefined username and password into appropriate text entry fields and then select a submit button. As another example, for a download confirmation action associated with a download confirmation screen, the host computing device 102 may dispatch actions to select a button to agree to license terms or otherwise confirm download and installation of an application.

[0059] In some embodiments, in block 528, the host computing device 102 may randomly select an unvisited object included in the display interface 208 for the test action. For example, the host computing device 102 may randomly select an unvisited object using

previously configured seed value as described above in connection with block 508. The host computing device 102 may perform the selected action on the probabilistically selected, unvisited object. In some embodiments, in block 530 the host computing device 102 may increase the probabilistic weight associated with unvisited objects on a pre-recorded critical path of a script file. In other words, in those embodiments the host computing device 102 may be more likely to select user interface objects on the critical path.

[0060] In block 532, the host computing device 102 records the test coverage for the user interface objects of the current scene. In particular, the host computing device 102 may record information based on the identity of the selected user interface object, the type of test action performed on the user interface object, and the overall scene of the display interface 208. The host computing device 102 may record the test coverage in any appropriate format.

[0061] Referring now to FIG. 7, pseudocode 700 illustrates one potential embodiment of a test coverage file that may be recorded by the host computing device 102. The illustrative pseudocode 700 is an XML file; however, it should be understood that the host computing device 102 may record test coverage in any appropriate format. The illustrative test coverage file includes a Test element for a package named “a.b.c,” which may correspond to a name or other identifier of the application 206. The Test element includes a Seed attribute equal to “100,” which may be used to initialize a pseudorandom number generator. As shown in FIGS. 6 and 7, the Seed attribute stored in the test coverage file may correspond to the random seed value provided in the test configuration file. As shown in pseudocode 700, the illustrative Test element includes a Scene element that includes six Object elements. Each of the Scene and Object elements may include a Hash attribute that includes a hash value, which may be used to uniquely identify the corresponding scene and/or user interface object. The hash values may be determined as scenes and objects are visited; thus, the Object elements that include Hash attributes may be used to identify the user interface objects that have been covered by testing. Each Object element further includes an Action attribute and a Type attribute, which indicate the type of test action that has been performed on that user interface object. Thus, the illustrative pseudocode 600 describes a Scene with a hash value of 429214, and that Scene includes six objects, of which four objects have been covered by testing (with hash values of 2704, 324, 993, and 2060).

[0062] Referring back to FIG. 5, in block 534, the host computing device 102 determines whether dispatching the test action has caused the display interface 208 to show a different scene. For example, a new scene may be displayed in response to selection of a navigation control, dismissal of a dialog box, or other action. The host computing device 102 may use any technique to determine whether the scene has changed, such as determining whether the hash value of the scene has changed. If no scene change has been detected, the method 500 loops back to block 510, in which the host computing device 102 continues to perform test actions on the current scene. If a new scene has been detected, the method 500 advances to block 536. In block 536, the host computing device 102 records the traversal between scenes. For example, the host computing device 102 may record traversing to a new scene and/or backtracking to a previous scene. Thus, the host computing device 102 may perform a depth-first traversal of the user interface of the test application 206. For example, referring again to FIG. 7, in response to traversing to a new scene, the host computing device 102 may record a new Scene element corresponding to the new scene as a new element in the illustrative XML file.

EXAMPLES

[0063] Illustrative examples of the technologies disclosed herein are provided below. An embodiment of the technologies may include any one or more, and any combination of, the examples described below.

[0064] Example 1 includes a computing device for automated application testing, the computing device comprising: a frame capture module to capture a frame of a display interface generated by an application executed by a test computing device; a crash detection module to detect a crash indication as a function of the frame of the display interface; and a test evaluation module to (i) detect an application crash based on an error log of the application in response to detection of the crash indication and (ii) report a test failure in response to detection of the application crash.

[0065] Example 2 includes the subject matter of Example 1, and wherein to capture the frame of the display interface comprises to capture a key frame in response to an application event generated by the application.

[0066] Example 3 includes the subject matter of any of Examples 1 and 2, and wherein to capture the frame of the display interface comprises to capture a frame of the display interface in response to expiration of a timer.

[0067] Example 4 includes the subject matter of any of Examples 1-3, and wherein the crash indication comprises a crash dialog box.

[0068] Example 5 includes the subject matter of any of Examples 1-4, and wherein to detect the crash dialog box comprises to: detect a rectangle in the frame of the display interface; and perform optical character recognition of the rectangle to generate text contents of the rectangle in response to detection of the rectangle.

[0069] Example 6 includes the subject matter of any of Examples 1-5, and wherein to detect the rectangle comprises to detect a rectangle with two lines of text.

[0070] Example 7 includes the subject matter of any of Examples 1-6, and wherein to detect the crash dialog box further comprises to: compare the text contents to a list of known crash words to determine a number of instances of crash words in the text contents; and determine whether the number of instances of crash words exceeds a predetermined threshold number of crash words.

[0071] Example 8 includes the subject matter of any of Examples 1-7, and wherein the predetermined threshold number of crash words comprises one crash word.

[0072] Example 9 includes the subject matter of any of Examples 1-8, and wherein the crash indication comprises a blank frame in the display interface.

[0073] Example 10 includes the subject matter of any of Examples 1-9, and wherein the crash indication comprises an application flash crash.

[0074] Example 11 includes the subject matter of any of Examples 1-10, and wherein to report the test failure comprises to store the frame of the display interface.

[0075] Example 12 includes a computing device for automated application testing, the computing device comprising: a scene capture module to capture a scene of a display interface generated by an application executed by a test computing device; an object recognition module to recognize one or more user interface objects of the scene of the display interface; an action dispatch module to (i) select a test action based on the scene of the display interface and (ii) perform the test action with the scene of the display interface in response to recognition of the one or more user interface objects; and a test coverage module to record test coverage data based

on the test action, the scene, and the one or more user interface objects in response to performance of the test action.

[0076] Example 13 includes the subject matter of Example 12, and wherein to select the test action comprises to: recognize a scene type as a function of the scene; and select the test action as a function of the scene type.

[0077] Example 14 includes the subject matter of any of Examples 12 and 13, and wherein the scene type comprises an account login screen and the test action comprises an account login action.

[0078] Example 15 includes the subject matter of any of Examples 12-14, and wherein the scene type comprises a download confirmation screen and the test action comprises a download confirmation action.

[0079] Example 16 includes the subject matter of any of Examples 12-15, and wherein to select the test action comprises to probabilistically select a test exploration action.

[0080] Example 17 includes the subject matter of any of Examples 12-16, and wherein to probabilistically select the test exploration action comprises to probabilistically select a test exploration action from a plurality of test exploration actions, wherein each of the plurality of test exploration actions is associated with a corresponding predefined probability value.

[0081] Example 18 includes the subject matter of any of Examples 12-17, and wherein the test exploration action comprises a click action or a swipe action.

[0082] Example 19 includes the subject matter of any of Examples 12-18, and wherein: the test exploration action comprises the click action; and to select the click action comprises to select a click action from a plurality of click actions, wherein each of the plurality of click actions is associated with a corresponding predefined probability value.

[0083] Example 20 includes the subject matter of any of Examples 12-19, and wherein the test exploration action comprises a click action selected from a plain click action, a home click action, a back click action, or a menu click action.

[0084] Example 21 includes the subject matter of any of Examples 12-20, and wherein: the click action comprises the plain click action; to perform the test action comprises to (i) probabilistically select a first object of the one or more user interface objects, wherein the first object is not previously visited and (ii) dispatch a click action to the first object; and to record the test coverage data comprises to record the first object as visited.

[0085] Example 22 includes the subject matter of any of Examples 12-21, and wherein the action dispatch module is further to increase a probability value associated with a critical path object of the one or more user interface objects, wherein the critical path object is included in a predefined test script.

[0086] Example 23 includes the subject matter of any of Examples 12-22, and wherein to recognize the one or more user interface objects comprises to determine a hash value corresponding to each of the one or more user interface objects.

[0087] Example 24 includes the subject matter of any of Examples 12-23, and wherein to recognize the one or more user interface objects of the scene of the display interface comprises to determine a hash value corresponding to the scene of the display interface.

[0088] Example 25 includes the subject matter of any of Examples 12-24, and wherein the test coverage module is further to: determine whether the scene of the display interface changes in response to the performance of the test action; and record test coverage data indicative of a traversal between scenes in response to a determination that the scene of the display interface changes.

[0089] Example 26 includes a method for automated application testing, the method comprising: capturing, by a computing device, a frame of a display interface generated by an application executed by a test computing device; detecting, by the computing device, a crash indication as a function of the frame of the display interface; detecting, by the computing device, an application crash based on an error log of the application in response to detecting the crash indication; and reporting, by the computing device, a test failure in response to detecting the application crash.

[0090] Example 27 includes the subject matter of Example 26, and wherein capturing the frame of the display interface comprises capturing a key frame in response to an application event generated by the application.

[0091] Example 28 includes the subject matter of any of Examples 26 and 27, and wherein capturing the frame of the display interface comprises capturing a frame of the display interface in response to expiration of a timer.

[0092] Example 29 includes the subject matter of any of Examples 26-28, and wherein detecting the crash indication comprises detecting a crash dialog box.

[0093] Example 30 includes the subject matter of any of Examples 26-29, and wherein detecting the crash dialog box comprises: detecting a rectangle in the frame of the display interface; and performing optical character recognition of the rectangle to generate text contents of the rectangle in response to detecting the rectangle.

[0094] Example 31 includes the subject matter of any of Examples 26-30, and wherein detecting the rectangle comprises detecting a rectangle including two lines of text.

[0095] Example 32 includes the subject matter of any of Examples 26-31, and wherein detecting the crash dialog box further comprises: comparing the text contents to a list of known crash words to determine a number of instances of crash words in the text contents; and determining whether the number of instances of crash words exceeds a predetermined threshold number of crash words.

[0096] Example 33 includes the subject matter of any of Examples 26-32, and wherein the predetermined threshold number of crash words comprises one crash word.

[0097] Example 34 includes the subject matter of any of Examples 26-33, and wherein detecting the crash indication comprises detecting a blank frame in the display interface.

[0098] Example 35 includes the subject matter of any of Examples 26-34, and wherein detecting the crash indication comprises detecting an application flash crash.

[0099] Example 36 includes the subject matter of any of Examples 26-35, and wherein reporting the test failure comprises storing the frame of the display interface.

[0100] Example 37 includes a method for automated application testing, the method comprising: capturing, by a computing device, a scene of a display interface generated by an application executed by a test computing device; recognizing, by the computing device, one or more user interface objects of the scene of the display interface; selecting, by the computing device, a test action based on the scene of the display interface; performing, by the computing device, the test action with the scene of the display interface in response to recognizing the one or more user interface objects; and recording, by the computing device, test coverage data based on the test action, the scene, and the one or more user interface objects in response to performing the test action.

[0101] Example 38 includes the subject matter of Example 37, and wherein selecting the test action comprises: recognizing a scene type as a function of the scene; and selecting the test action as a function of the scene type.

[0102] Example 39 includes the subject matter of any of Examples 37 and 38, and wherein: recognizing the scene type comprises recognizing an account login screen; and selecting the test action comprises selecting an account login action.

[0103] Example 40 includes the subject matter of any of Examples 37-39, and wherein: recognizing the scene type comprises recognizing a download confirmation screen; and selecting the test action comprises selecting a download confirmation action.

[0104] Example 41 includes the subject matter of any of Examples 37-40, and wherein selecting the test action comprises probabilistically selecting a test exploration action.

[0105] Example 42 includes the subject matter of any of Examples 37-41, and wherein probabilistically selecting the test exploration action comprises probabilistically selecting a test exploration action from a plurality of test exploration actions, wherein each of the plurality of test exploration actions is associated with a corresponding predefined probability value.

[0106] Example 43 includes the subject matter of any of Examples 37-42, and wherein selecting the test action comprises probabilistically selecting a click action or a swipe action.

[0107] Example 44 includes the subject matter of any of Examples 37-43, and wherein: selecting the test action comprises selecting the click action; and selecting the click action comprises selecting a click action from a plurality of click actions, wherein each of the plurality of click actions is associated with a corresponding predefined probability value.

[0108] Example 45 includes the subject matter of any of Examples 37-44, and wherein: selecting the test action comprises selecting the click action; and selecting the click action comprises probabilistically selecting a plain click action, a home click action, a back click action, or a menu click action.

[0109] Example 46 includes the subject matter of any of Examples 37-45, and wherein: selecting the click action comprises selecting the plain click action; performing the test action comprises (i) probabilistically selecting a first object of the one or more user interface objects, wherein the first object is not previously visited and (ii) dispatching a click action to the first object; and recording the test coverage data comprises recording the first object as visited.

[0110] Example 47 includes the subject matter of any of Examples 37-46, and further comprising increasing a probability value associated with a critical path object of the one or more user interface objects, wherein the critical path object is included in a predefined test script.

[0111] Example 48 includes the subject matter of any of Examples 37-47, and wherein recognizing the one or more user interface objects comprises determining a hash value corresponding to each of the one or more user interface objects.

[0112] Example 49 includes the subject matter of any of Examples 37-48, and wherein recognizing the one or more user interface objects of the scene of the display interface comprises determining a hash value corresponding to the scene of the display interface.

[0113] Example 50 includes the subject matter of any of Examples 37-49, and further comprising: determining, by the computing device, whether the scene of the display interface changes in response to performing the test action; and recording, by the computing device, test coverage data indicative of a traversal between scenes in response to determining that the scene of the display interface changes.

[0114] Example 51 includes a computing device comprising: a processor; and a memory having stored therein a plurality of instructions that when executed by the processor cause the computing device to perform the method of any of Examples 26-50.

[0115] Example 52 includes one or more machine readable storage media comprising a plurality of instructions stored thereon that in response to being executed result in a computing device performing the method of any of Examples 26-50.

[0116] Example 53 includes a computing device comprising means for performing the method of any of Examples 26-50.

[0117] Example 54 includes a computing device for automated application testing, the computing device comprising: means for capturing a frame of a display interface generated by an application executed by a test computing device; means for detecting a crash indication as a function of the frame of the display interface; means for detecting an application crash based on an error log of the application in response to detecting the crash indication; and means for reporting a test failure in response to detecting the application crash.

[0118] Example 55 includes the subject matter of Example 54, and wherein the means for capturing the frame of the display interface comprises means for capturing a key frame in response to an application event generated by the application.

[0119] Example 56 includes the subject matter of any of Examples 54 and 55, and wherein the means for capturing the frame of the display interface comprises means for capturing a frame of the display interface in response to expiration of a timer.

[0120] Example 57 includes the subject matter of any of Examples 54-56, and wherein the means for detecting the crash indication comprises means for detecting a crash dialog box.

[0121] Example 58 includes the subject matter of any of Examples 54-57, and wherein the means for detecting the crash dialog box comprises: means for detecting a rectangle in the frame of the display interface; and means for performing optical character recognition of the rectangle to generate text contents of the rectangle in response to detecting the rectangle.

[0122] Example 59 includes the subject matter of any of Examples 54-58, and wherein the means for detecting the rectangle comprises means for detecting a rectangle including two lines of text.

[0123] Example 60 includes the subject matter of any of Examples 54-59, and wherein the means for detecting the crash dialog box further comprises: means for comparing the text contents to a list of known crash words to determine a number of instances of crash words in the text contents; and means for determining whether the number of instances of crash words exceeds a predetermined threshold number of crash words.

[0124] Example 61 includes the subject matter of any of Examples 54-60, and wherein the predetermined threshold number of crash words comprises one crash word.

[0125] Example 62 includes the subject matter of any of Examples 54-61, and wherein the means for detecting the crash indication comprises means for detecting a blank frame in the display interface.

[0126] Example 63 includes the subject matter of any of Examples 54-62, and wherein the means for detecting the crash indication comprises means for detecting an application flash crash.

[0127] Example 64 includes the subject matter of any of Examples 54-63, and wherein the means for reporting the test failure comprises means for storing the frame of the display interface.

[0128] Example 65 includes a computing device for automated application testing, the computing device comprising: means for capturing a scene of a display interface generated by an application executed by a test computing device; means for recognizing one or more user interface objects of the scene of the display interface; means for selecting a test action based on the scene of the display interface; means for performing the test action with the scene of the display interface in response to recognizing the one or more user interface objects; and means for

recording test coverage data based on the test action, the scene, and the one or more user interface objects in response to performing the test action.

[0129] Example 66 includes the subject matter of Example 65, and wherein the means for selecting the test action comprises: means for recognizing a scene type as a function of the scene; and means for selecting the test action as a function of the scene type.

[0130] Example 67 includes the subject matter of any of Examples 65 and 66, and wherein: the means for recognizing the scene type comprises means for recognizing an account login screen; and the means for selecting the test action comprises means for selecting an account login action.

[0131] Example 68 includes the subject matter of any of Examples 65-67, and wherein: the means for recognizing the scene type comprises means for recognizing a download confirmation screen; and the means for selecting the test action comprises means for selecting a download confirmation action.

[0132] Example 69 includes the subject matter of any of Examples 65-68, and wherein the means for selecting the test action comprises means for probabilistically selecting a test exploration action.

[0133] Example 70 includes the subject matter of any of Examples 65-69, and wherein the means for probabilistically selecting the test exploration action comprises means for probabilistically selecting a test exploration action from a plurality of test exploration actions, wherein each of the plurality of test exploration actions is associated with a corresponding predefined probability value.

[0134] Example 71 includes the subject matter of any of Examples 65-70, and wherein the means for selecting the test action comprises means for probabilistically selecting a click action or a swipe action.

[0135] Example 72 includes the subject matter of any of Examples 65-71, and wherein: the means for selecting the test action comprises means for selecting the click action; and the means for selecting the click action comprises means for selecting a click action from a plurality of click actions, wherein each of the plurality of click actions is associated with a corresponding predefined probability value.

[0136] Example 73 includes the subject matter of any of Examples 65-72, and wherein: the means for selecting the test action comprises means for selecting the click action; and the

means for selecting the click action comprises means for probabilistically selecting a plain click action, a home click action, a back click action, or a menu click action.

[0137] Example 74 includes the subject matter of any of Examples 65-73, and wherein: the means for selecting the click action comprises means for selecting the plain click action; the means for performing the test action comprises (i) means for probabilistically selecting a first object of the one or more user interface objects, wherein the first object is not previously visited and (ii) means for dispatching a click action to the first object; and the means for recording the test coverage data comprises means for recording the first object as visited.

[0138] Example 75 includes the subject matter of any of Examples 65-74, and further comprising means for increasing a probability value associated with a critical path object of the one or more user interface objects, wherein the critical path object is included in a predefined test script.

[0139] Example 76 includes the subject matter of any of Examples 65-75, and wherein the means for recognizing the one or more user interface objects comprises means for determining a hash value corresponding to each of the one or more user interface objects.

[0140] Example 77 includes the subject matter of any of Examples 65-76, and wherein the means for recognizing the one or more user interface objects of the scene of the display interface comprises means for determining a hash value corresponding to the scene of the display interface.

[0141] Example 78 includes the subject matter of any of Examples 65-77, and further comprising: means for determining whether the scene of the display interface changes in response to performing the test action; and means for recording test coverage data indicative of a traversal between scenes in response to determining that the scene of the display interface changes.

Claims:

1. A computing device for automated application testing, the computing device comprising:
 - a frame capture module to capture a frame of a display interface generated by an application executed by a test computing device;
 - a crash detection module to detect a crash indication as a function of the frame of the display interface; and
 - a test evaluation module to (i) detect an application crash based on an error log of the application in response to detection of the crash indication and (ii) report a test failure in response to detection of the application crash.
2. The computing device of claim 1, wherein to capture the frame of the display interface comprises to capture a key frame in response to an application event generated by the application.
3. The computing device of claim 1, wherein:
 - the crash indication comprises a crash dialog box; and
 - to detect the crash dialog box comprises to: (i) detect a rectangle in the frame of the display interface, and (ii) perform optical character recognition of the rectangle to generate text contents of the rectangle in response to detection of the rectangle.
4. The computing device of claim 3, wherein to detect the crash dialog box further comprises to:
 - compare the text contents to a list of known crash words to determine a number of instances of crash words in the text contents; and
 - determine whether the number of instances of crash words exceeds a predetermined threshold number of crash words.
5. The computing device of claim 1, wherein the crash indication comprises a blank frame in the display interface or an application flash crash.

6. The computing device of any of claims 1-5, wherein to report the test failure comprises to store the frame of the display interface.

7. A computing device for automated application testing, the computing device comprising:

a scene capture module to capture a scene of a display interface generated by an application executed by a test computing device;

an object recognition module to recognize one or more user interface objects of the scene of the display interface;

an action dispatch module to (i) select a test action based on the scene of the display interface and (ii) perform the test action with the scene of the display interface in response to recognition of the one or more user interface objects; and

a test coverage module to record test coverage data based on the test action, the scene, and the one or more user interface objects in response to performance of the test action.

8. The computing device of claim 7, wherein to select the test action comprises to:
recognize a scene type as a function of the scene; and
select the test action as a function of the scene type.

9. The computing device of claim 8, wherein:
the scene type comprises an account login screen and the test action comprises an account login action; or

the scene type comprises a download confirmation screen and the test action comprises a download confirmation action.

10. The computing device of claim 7, wherein to select the test action comprises to probabilistically select a test exploration action.

11. The computing device of claim 10, wherein to probabilistically select the test exploration action comprises to probabilistically select a test exploration action from a plurality of test exploration actions, wherein each of the plurality of test exploration actions is associated with a corresponding predefined probability value.

12. The computing device of claim 10, wherein the test exploration action comprises a click action selected from a plain click action, a home click action, a back click action, or a menu click action.

13. The computing device of claim 12, wherein:
the click action comprises the plain click action;
to perform the test action comprises to (i) probabilistically select a first object of the one or more user interface objects, wherein the first object is not previously visited and (ii) dispatch a click action to the first object;
to record the test coverage data comprises to record the first object as visited; and
the action dispatch module is further to increase a probability value associated with a critical path object of the one or more user interface objects, wherein the critical path object is included in a predefined test script.

14. The computing device of any of claims 7-13, wherein to recognize the one or more user interface objects comprises to:
determine a hash value corresponding to each of the one or more user interface objects; or
determine a hash value corresponding to the scene of the display interface.

15. The computing device of any of claims 7-13, wherein the test coverage module is further to:
determine whether the scene of the display interface changes in response to the performance of the test action; and
record test coverage data indicative of a traversal between scenes in response to a determination that the scene of the display interface changes.

16. A method for automated application testing, the method comprising:
capturing, by a computing device, a frame of a display interface generated by an application executed by a test computing device;
detecting, by the computing device, a crash indication as a function of the frame of the display interface;

detecting, by the computing device, an application crash based on an error log of the application in response to detecting the crash indication; and

reporting, by the computing device, a test failure in response to detecting the application crash.

17. The method of claim 16, wherein detecting the crash indication comprises detecting a crash dialog box.

18. The method of claim 17, wherein detecting the crash dialog box comprises:
detecting a rectangle in the frame of the display interface;
performing optical character recognition of the rectangle to generate text contents of the rectangle in response to detecting the rectangle;
comparing the text contents to a list of known crash words to determine a number of instances of crash words in the text contents; and
determining whether the number of instances of crash words exceeds a predetermined threshold number of crash words.

19. A method for automated application testing, the method comprising:
capturing, by a computing device, a scene of a display interface generated by an application executed by a test computing device;
recognizing, by the computing device, one or more user interface objects of the scene of the display interface;
selecting, by the computing device, a test action based on the scene of the display interface;
performing, by the computing device, the test action with the scene of the display interface in response to recognizing the one or more user interface objects; and
recording, by the computing device, test coverage data based on the test action, the scene, and the one or more user interface objects in response to performing the test action.

20. The method of claim 19, wherein selecting the test action comprises:
recognizing a scene type as a function of the scene; and
selecting the test action as a function of the scene type.

21. The method of claim 19, wherein selecting the test action comprises probabilistically selecting a test exploration action.

22. The method of claim 19, further comprising:
determining, by the computing device, whether the scene of the display interface changes in response to performing the test action; and
recording, by the computing device, test coverage data indicative of a traversal between scenes in response to determining that the scene of the display interface changes.

23. A computing device comprising:
a processor; and
a memory having stored therein a plurality of instructions that when executed by the processor cause the computing device to perform the method of any of claims 16-22.

24. One or more machine readable storage media comprising a plurality of instructions stored thereon that in response to being executed result in a computing device performing the method of any of claims 16-22.

25. A computing device comprising means for performing the method of any of claims 16-22.

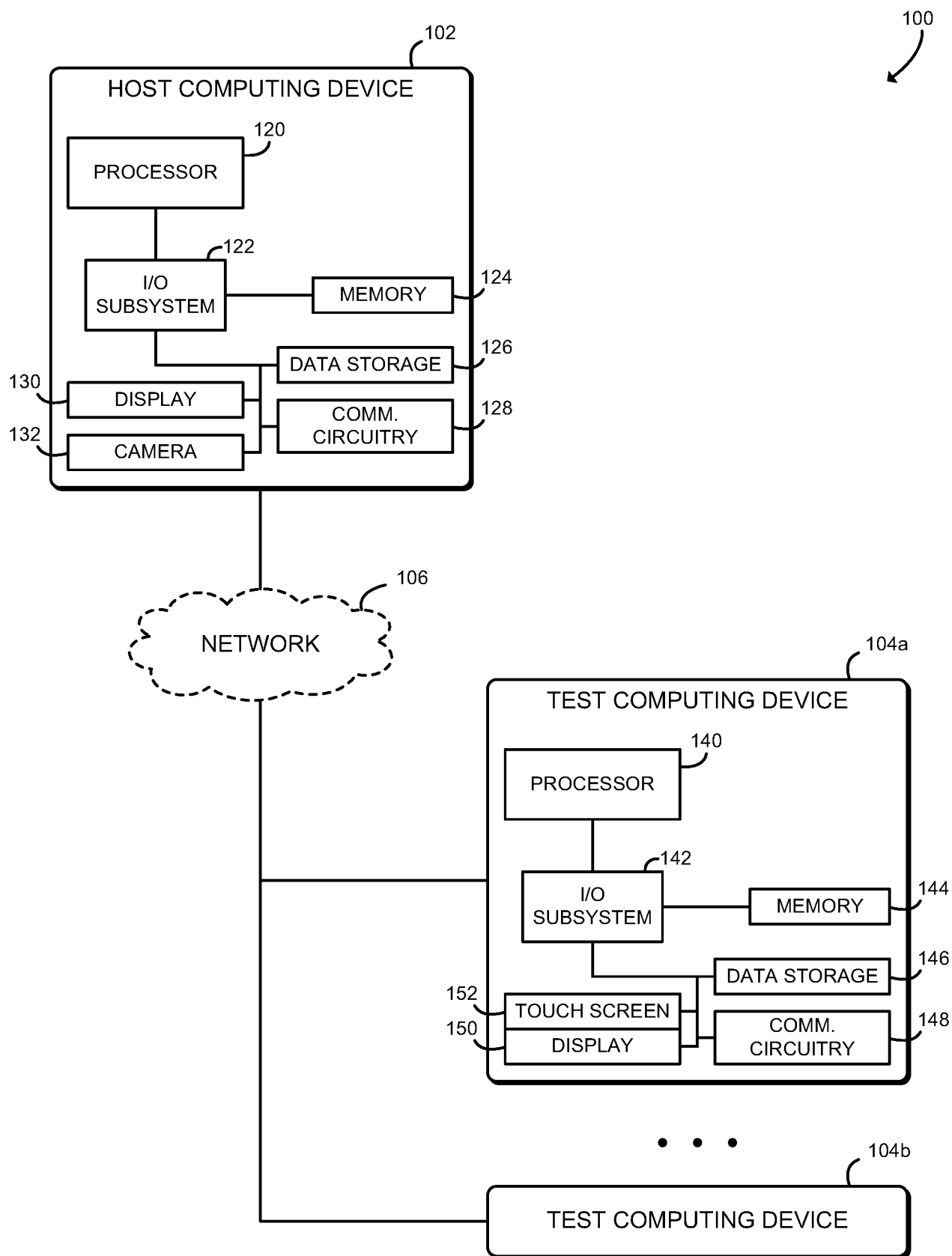


FIG. 1

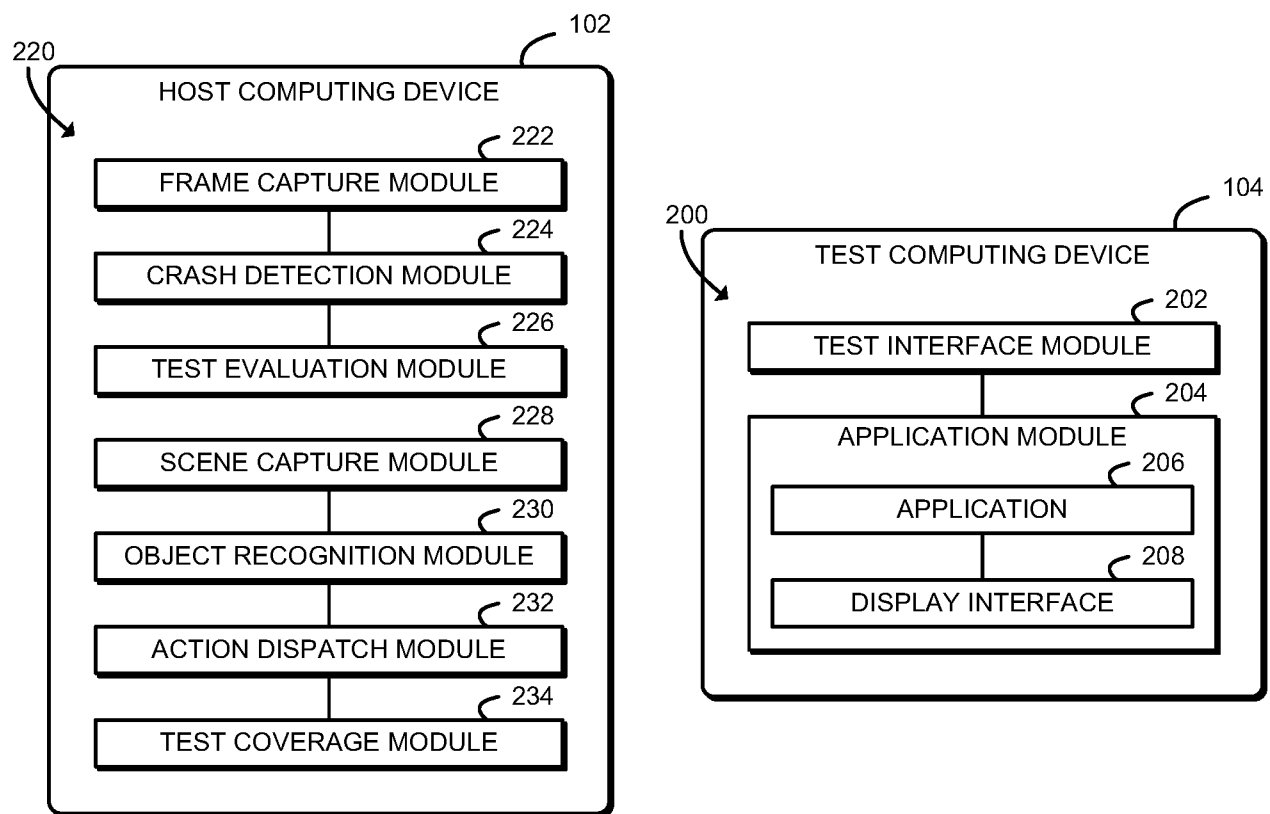


FIG. 2

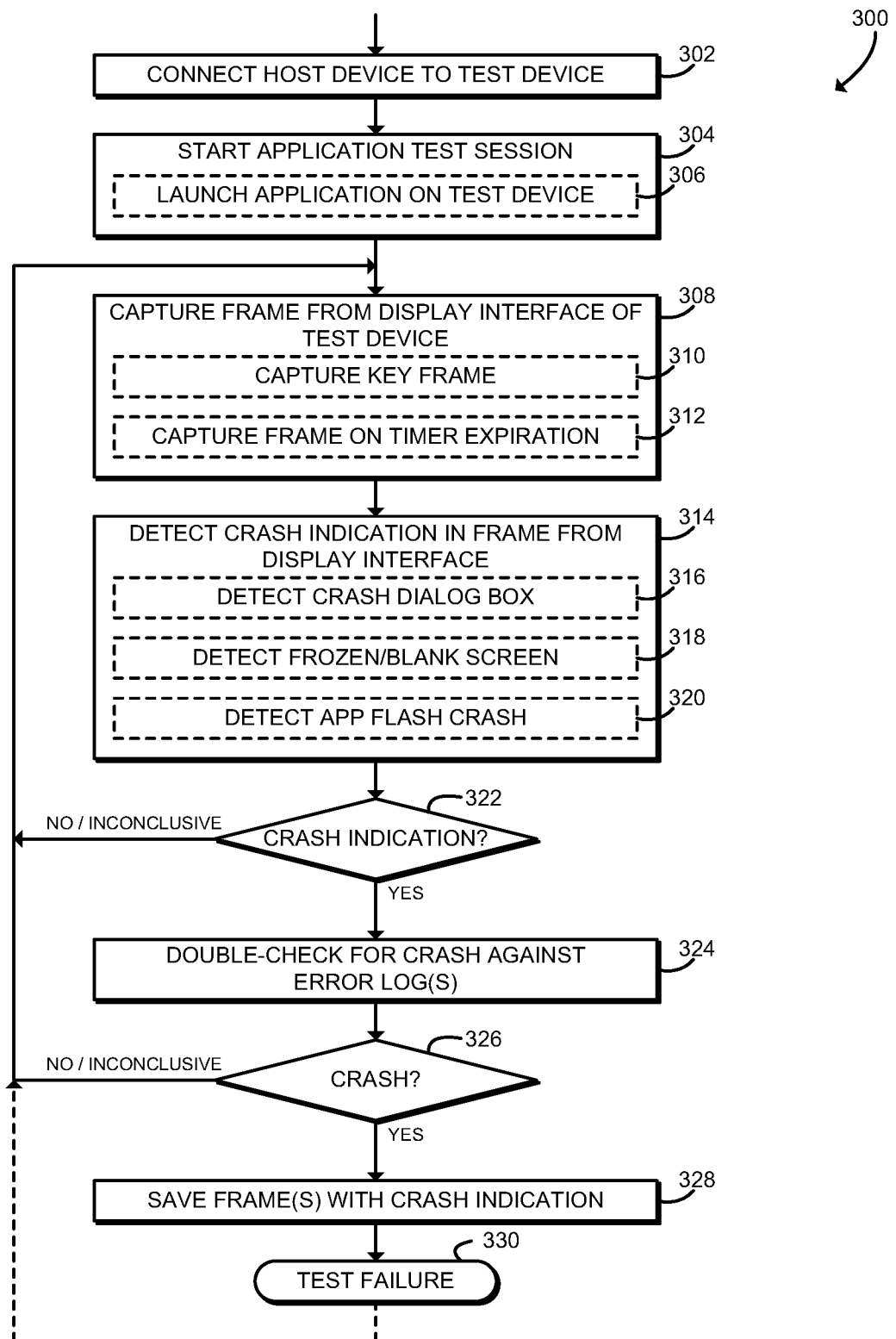


FIG. 3

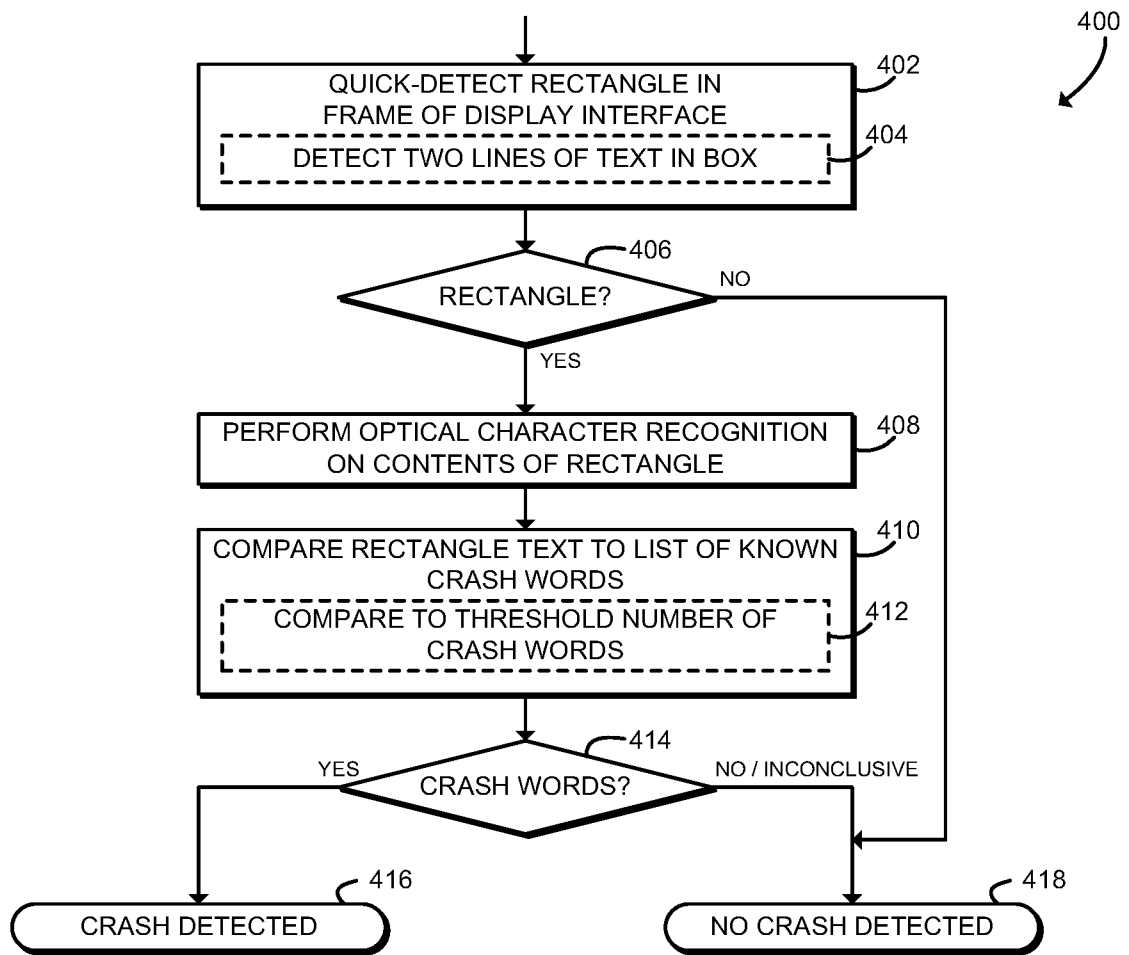


FIG. 4

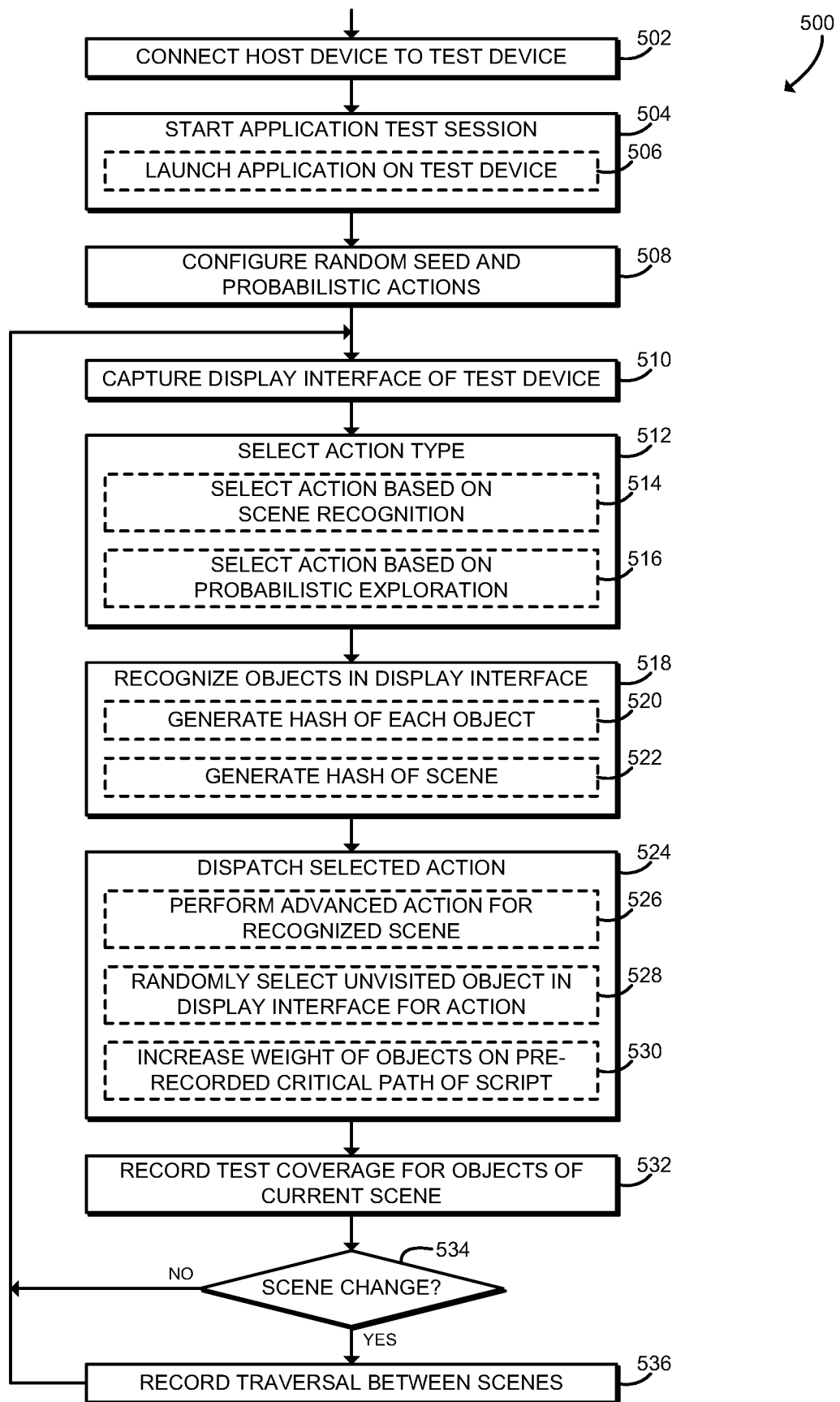


FIG. 5

600



```
<?xml version="1.0" ?>
<TestRoot>
  <Test PackageName="a.b.c" Seed="100">
    <Action Type="ACCOUNT" Value = "YES">
      <Account Name="abc@mail.com" Password="abc" />
    </Action>
    <Action Type="CLICK" Value="90%">
      <Click Type="PLAIN" Value="80%" />
      <Click Type="HOME" Value="20%" />
    </Action>
    <Action Type="SWIPE" Value="10%" />
  </Test>
</TestRoot>
```

FIG. 6

700



```
<?xml version="1.0" ?>
<TestRoot>
  <Test PackageName="a.b.c" Seed="100">
    <Scene Hash="429214" ObjectNum="6">
      <Object Hash="2704" Action="Click" Type="Button" />
      <Object Hash="N" Action="Swipe" Type="Down" />
      <Object Hash="324" Action="Click" Type="Button" />
      <Object Hash="993" Action="Click" Type="Button" />
      <Object Hash="N" Action="Click" Type="Menu" />
      <Object Hash="2060" Action="Click" Type="Button" />
    </Scene>
  </Test>
</TestRoot>
```

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2015/098840

A. CLASSIFICATION OF SUBJECT MATTER

G06F 11/36(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

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

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

CNPAT, WPI, EPODOC, CNKI: program, application, app, test, coverage, capture, print, snap shot, screen, interface, image, picture, detect, recognition, crash, dialog

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 102637144 A (QIZHI SOFTWARE BEIJING CO., LTD.) 15 August 2012 (2012-08-15) description, paragraphs [0021]-[0023], [0034]-[0048], and figures 1-3	1,5,6,16,23-25
X	CN 104484273 A (GUANGZHOU BOGUAN INFORMATION SCIENCE&TECHNOLOGY CO., LTD.) 01 April 2015 (2015-04-01) description, paragraphs [0075]-[0099], [0135]-[0168], and figures 1-6, 8	7-15,19-25
Y	CN 102637144 A (QIZHI SOFTWARE BEIJING CO., LTD.) 15 August 2012 (2012-08-15) description, paragraphs [0021]-[0023], [0034]-[0048], and figures 1-3	2-4,6,17,18,23-25
Y	CN 104484273 A (GUANGZHOU BOGUAN INFORMATION SCIENCE&TECHNOLOGY CO., LTD.) 01 April 2015 (2015-04-01) description, paragraphs [0075]-[0099], [0135]-[0168], and figures 1-6, 8	2-4,6,17,18,23-25
X	CN 104516812 A (TENCENT TECHNOLOGY SHENZHEN CO., LTD.) 15 April 2015 (2015-04-15) description, paragraphs [0032]-[0086], and figures 1-5	7-15,19-25
A	CN 104866414 A (TENCENT TECHNOLOGY SHENZHEN CO., LTD.) 26 August 2015 (2015-08-26) the whole document	1-25



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

13 September 2016

Date of mailing of the international search report

22 September 2016

Name and mailing address of the ISA/CN

STATE INTELLECTUAL PROPERTY OFFICE OF THE
P.R.CHINA
6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing
100088, China

Facsimile No. (86-10)62019451

Authorized officer

LI,Na

Telephone No. (86-10)62414068

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2015/098840**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 104252410 A (ALIBABA GROUP HOLDING CO., LTD.) 31 December 2014 (2014-12-31) the whole document	1-25
A	US 5335342 A (TIBURON SYSTEMS, INC.) 02 August 1994 (1994-08-02) the whole document	1-25

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2015/098840

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	102637144	A	15 August 2012	CN	102637144	B	06 May 2015
CN	104484273	A	01 April 2015	WO	2016095554	A1	23 June 2016
CN	104516812	A	15 April 2015	None			
CN	104866414	A	26 August 2015	None			
CN	104252410	A	31 December 2014	None			
US	5335342	A	02 August 1994	None			