US 20080229288A1

(19) United States

(12) Patent Application Publication

Nelson et al.

(10) Pub. No.: US 2008/0229288 A1

(43) Pub. Date: Sep. 18, 2008

(54) # SOFTWARE PLUGIN MODULES FOR DEVICE TESTING

(76) Inventors: **Steve Nelson**, San Jose, CA (US); **Victor Ivashin**, Danville, CA (US); **Jamie C. Rasmussen**, Somerville, MA (US)

Correspondence Address:
EPSON RESEARCH AND DEVELOPMENT INC
INTELLECTUAL PROPERTY DEPT
2580 ORCHARD PARKWAY, SUITE 225
SAN JOSE, CA 95131 (US)

(21) Appl. No.: **11/923,464**

(22) Filed: **Oct. 24, 2007**

Related U.S. Application Data

(63) Continuation-in-part of application No. 11/685,592, filed on Mar. 13, 2007, Continuation-in-part of application No. 11/685,600, filed on Mar. 13, 2007, Continuation-in-part of application No. 11/685,604, filed on Mar. 13, 2007.

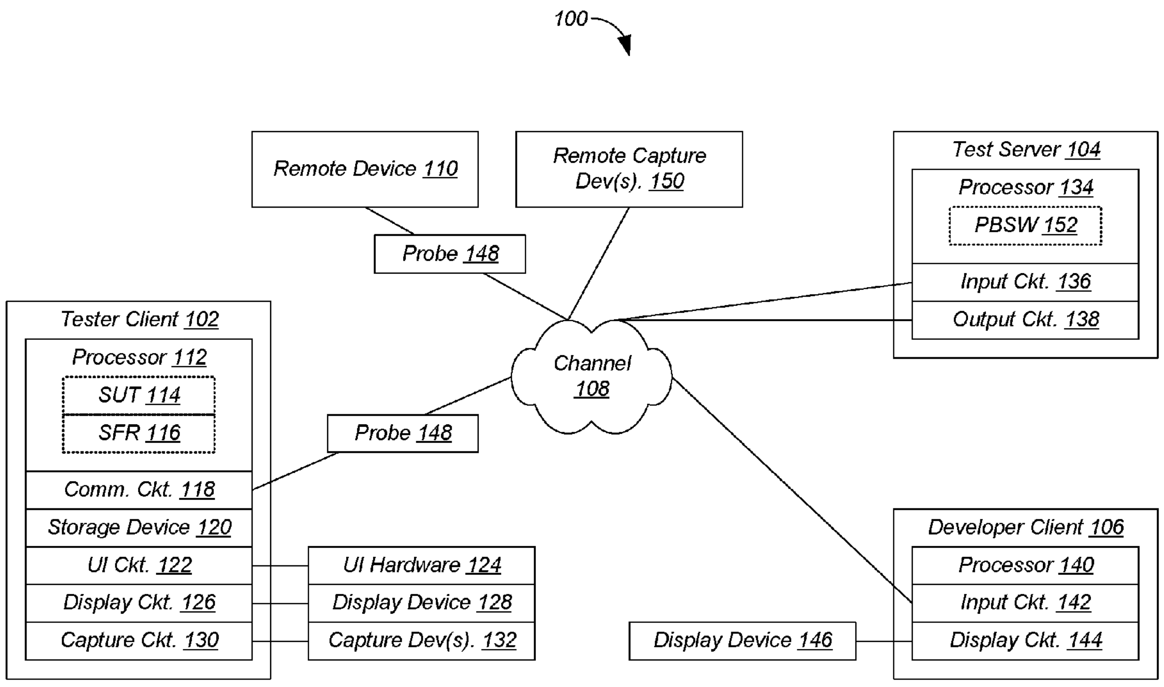
Publication Classification

(51) **Int. Cl.**
G06F 9/44 (2006.01)

(52) **U.S. Cl.** **717/127**

(57) **ABSTRACT**

Apparatus having corresponding methods and computer-readable media comprise: one or more plugins each adapted to execute on the computer, wherein each plugin comprises a monitor module adapted to monitor data exchanged between one or more applications executing on the computer and a respective device driver executing on the computer, and a capture module adapted to capture the data.



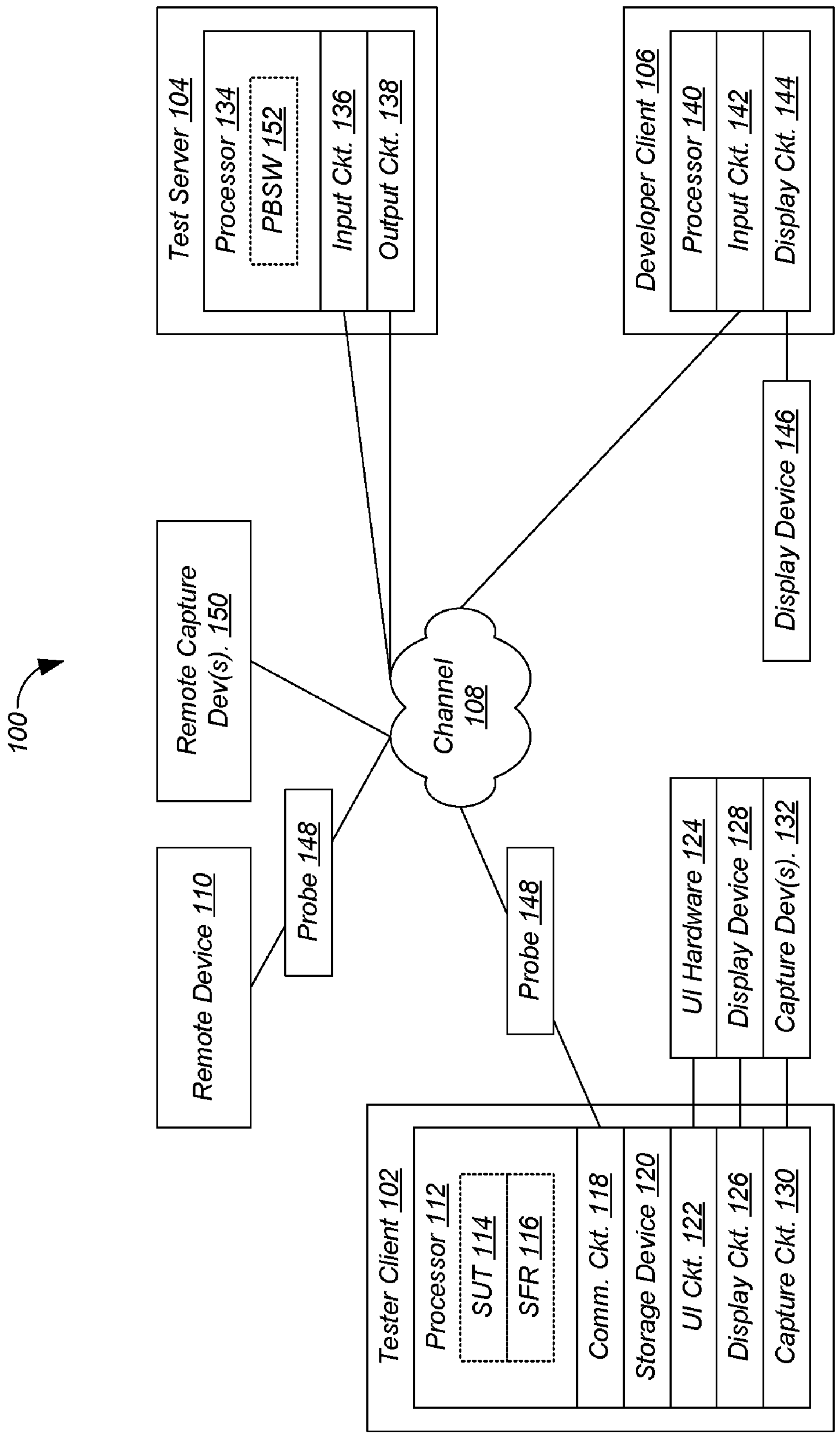

100
Test Server 104
Processor 134
PBSW 152
Input Ckt. 136
Output Ckt. 138
Developer Client 106
Processor 140
Input Ckt. 142
Display Ckt. 144
Display Device 146
Remote Capture Dev(s). 150
Channel 108
Probe 148
Remote Device 110
Probe 148
UI Hardware 124
Display Device 128
Capture Dev(s). 132
Tester Client 102
Processor 112
SUT 114
SFR 116
Comm. Ckt. 118
Storage Device 120
UI Ckt. 122
Display Ckt. 126
Capture Ckt. 130

FIG. 1

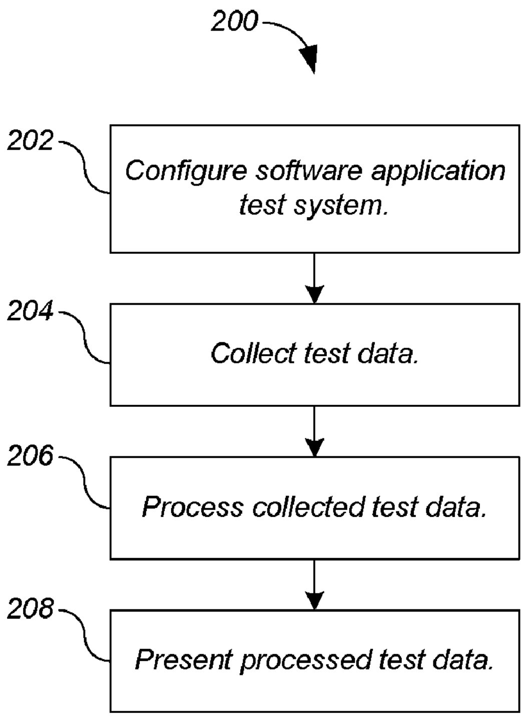

200
202
Configure software application test system.
204
Collect test data.
206
Process collected test data.
208
Present processed test data.

FIG. 2

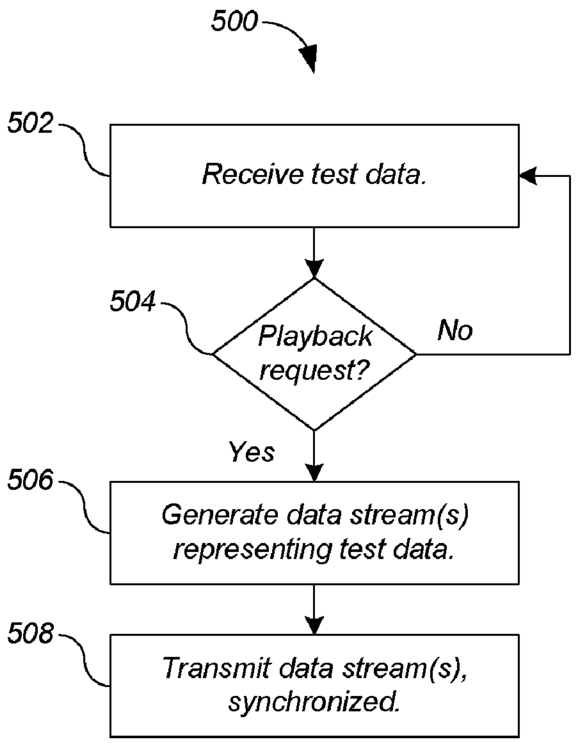

500
502
Receive test data.
504
Playback request?
No
Yes
506
Generate data stream(s) representing test data.
508
Transmit data stream(s), synchronized.

FIG. 5

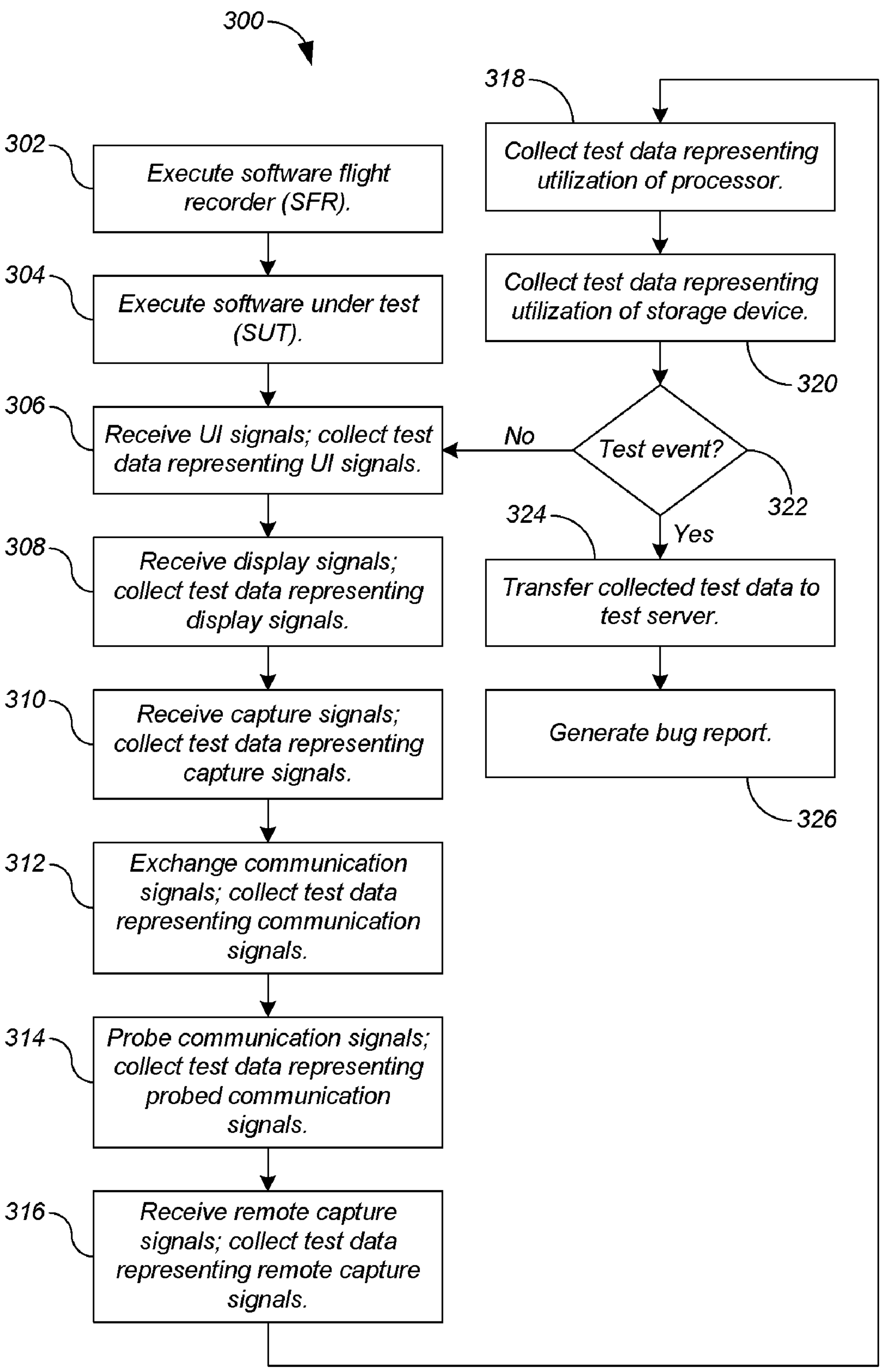
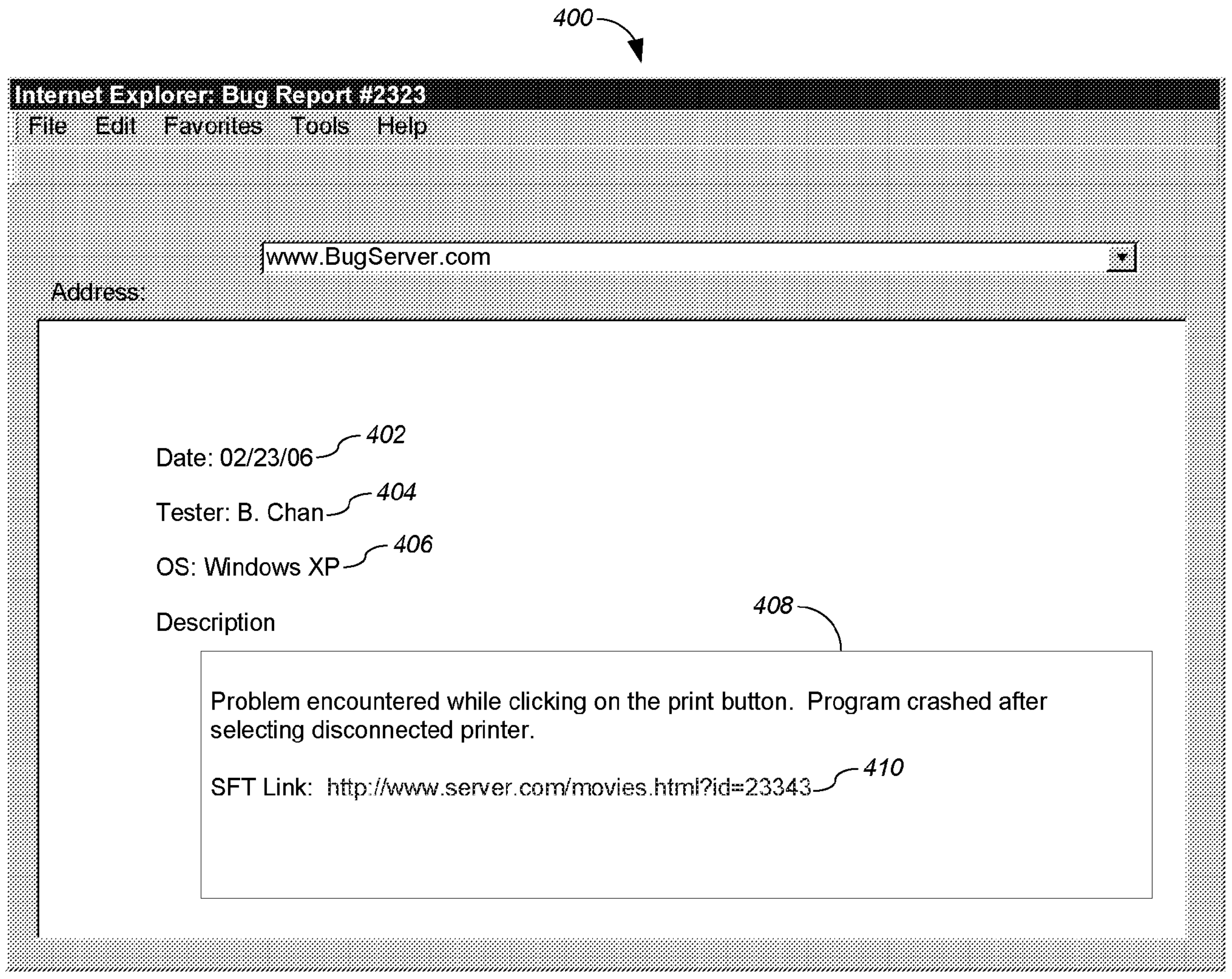

300
302
Execute software flight recorder (SFR).
304
Execute software under test (SUT).
306
Receive UI signals; collect test data representing UI signals.
308
Receive display signals; collect test data representing display signals.
310
Receive capture signals; collect test data representing capture signals.
312
Exchange communication signals; collect test data representing communication signals.
314
Probe communication signals; collect test data representing probed communication signals.
316
Receive remote capture signals; collect test data representing remote capture signals.
318
Collect test data representing utilization of processor.
Collect test data representing utilization of storage device.
320
Test event?
322
No
Yes
324
Transfer collected test data to test server.
Generate bug report.
326

FIG. 3


400
Internet Explorer: Bug Report #2323
File Edit Favorites Tools Help
Address:
402
404
406
408
410

www.BugServer.com

Date: 02/23/06

Tester: B. Chan

OS: Windows XP

Description

Problem encountered while clicking on the print button. Program crashed after selecting disconnected printer.

SFT Link: http://www.server.com/movies.html?id=23343

FIG. 4

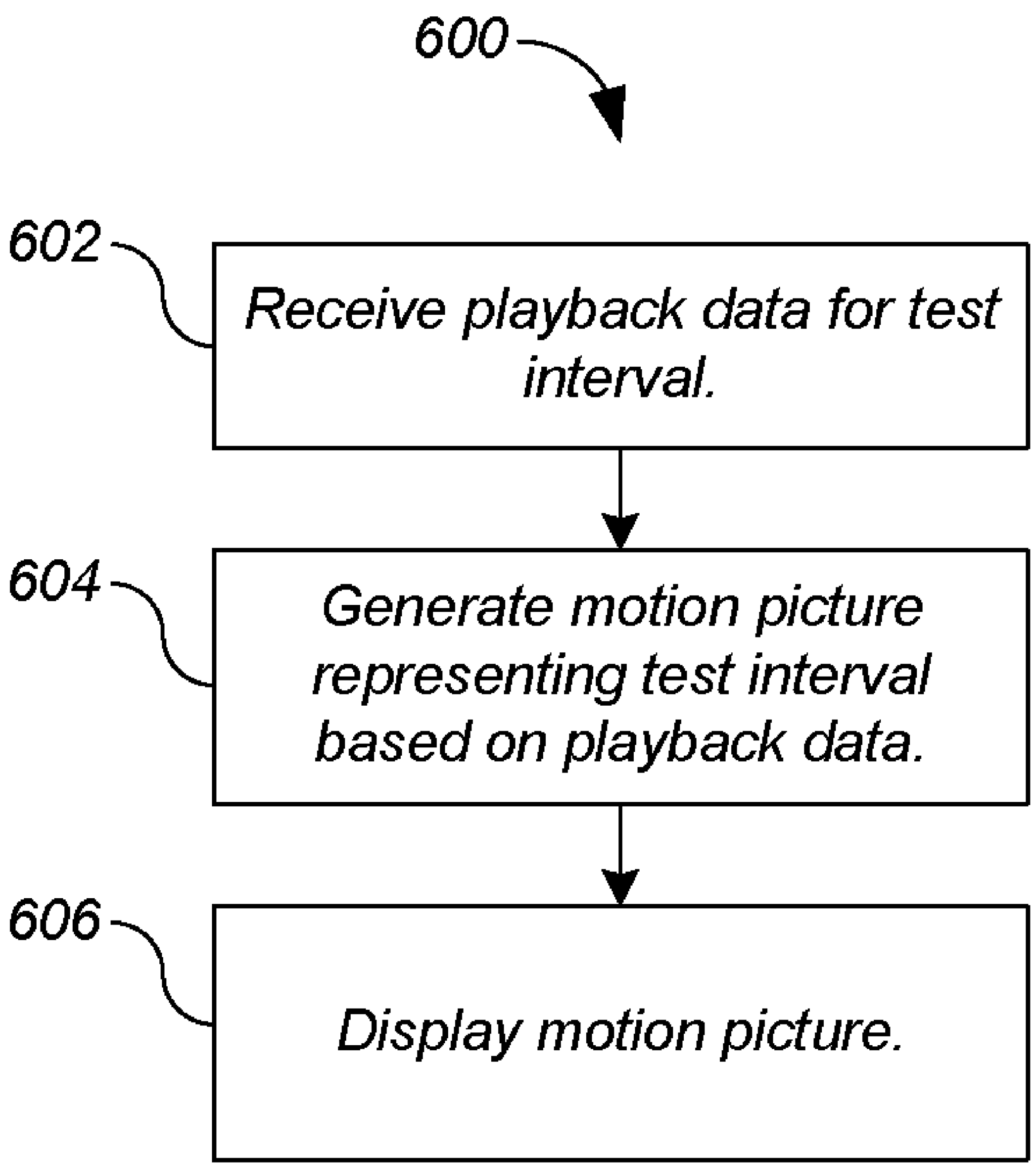
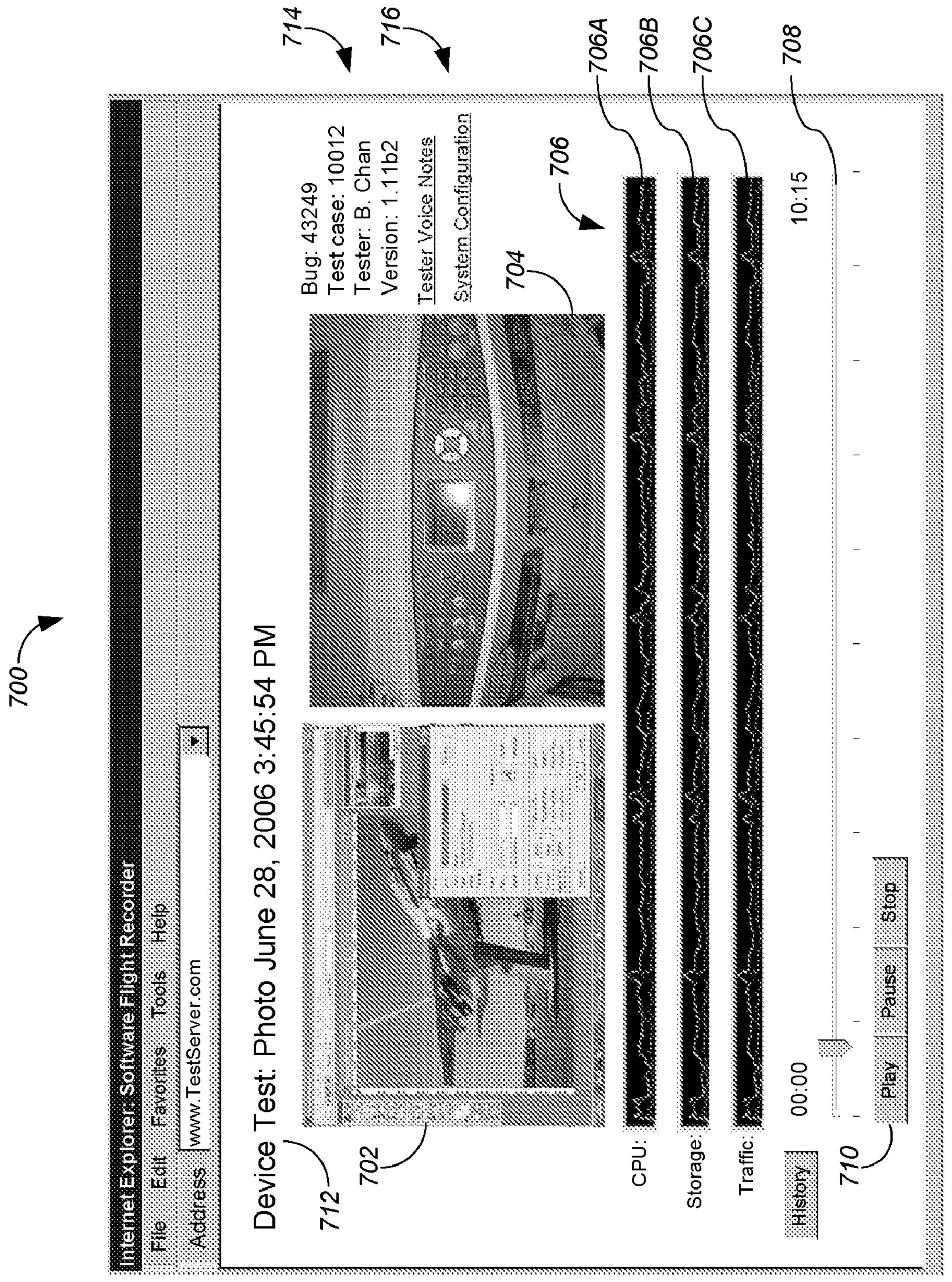

600
602
Receive playback data for test interval.
604
Generate motion picture representing test interval based on playback data.
606
Display motion picture.

FIG. 6


700
Internet Explorer: Software Flight Recorder
File Edit Favorites Tools Help
Address www.TestServer.com
712
702
704
706
706A
706B
706C
708
710
714
716
CPU:
Storage:
Traffic:
00:00
10:15
History
Play
Pause
Stop

Device Test: Photo June 28, 2006 3:45:54 PM

Bug: 43249
Test case: 10012
Tester: B. Chan
Version: 1.11b2
Tester Voice Notes
System Configuration

FIG. 7
"

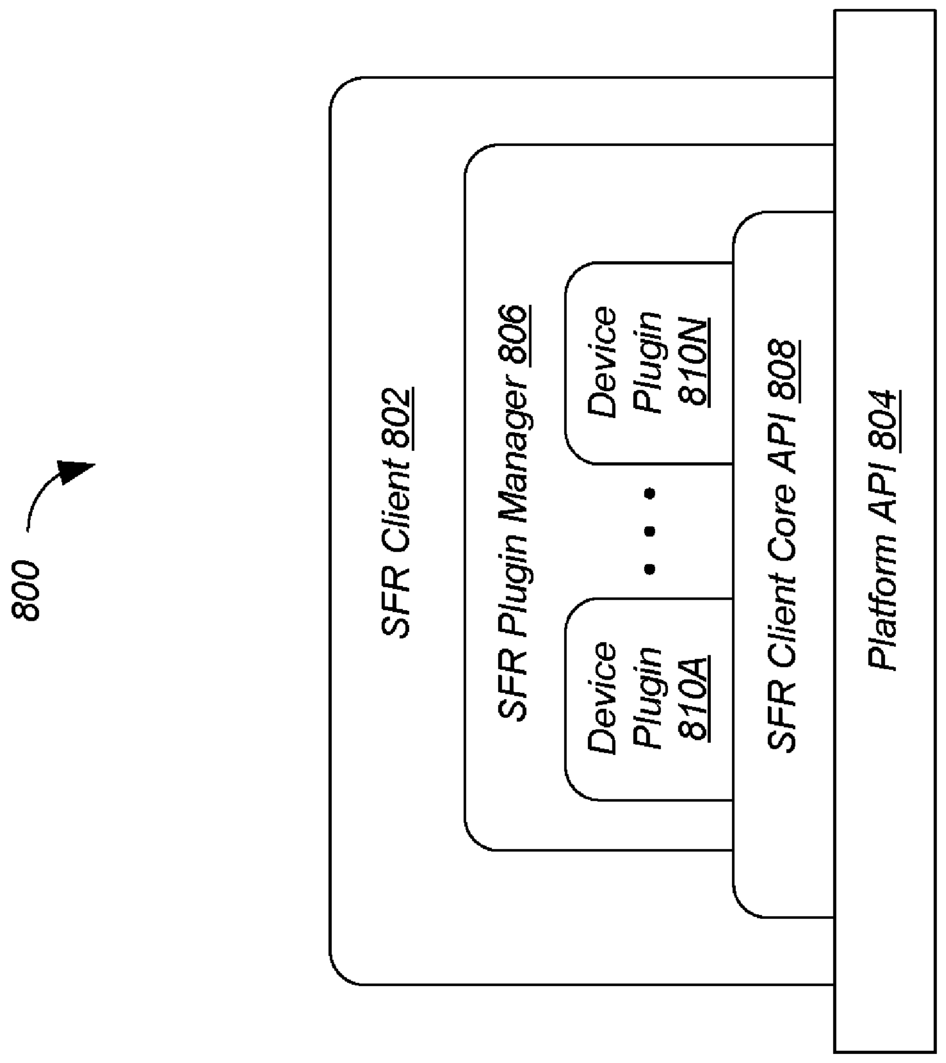

800
SFR Client 802
SFR Plugin Manager 806
Device Plugin 810A
Device Plugin 810N
SFR Client Core API 808
Platform API 804

FIG. 8

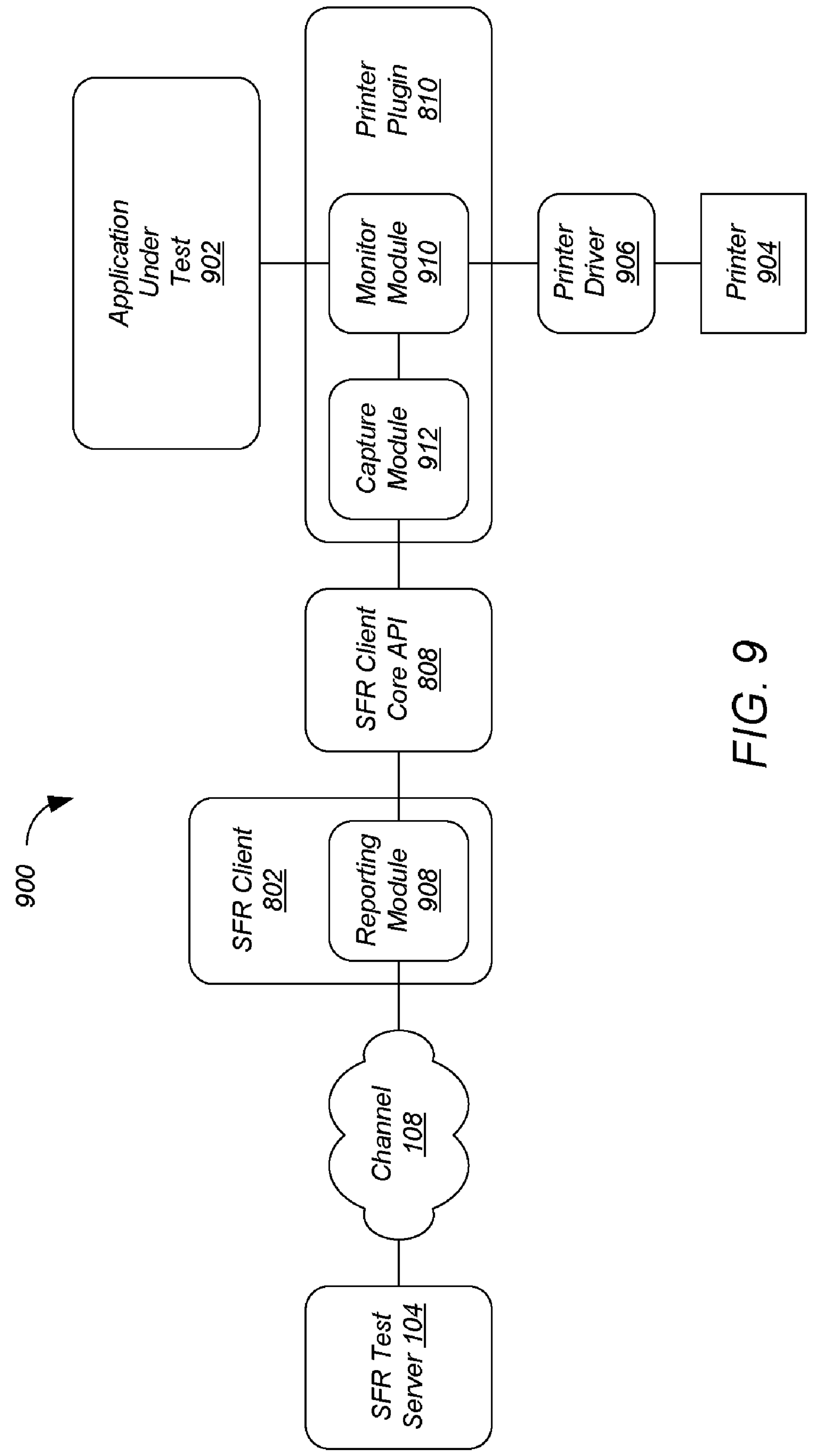

900
Application Under Test 902
Printer Plugin 810
Monitor Module 910
Capture Module 912
Printer Driver 906
Printer 904
SFR Client Core API 808
SFR Client 802
Reporting Module 908
Channel 108
SFR Test Server 104

FIG. 9

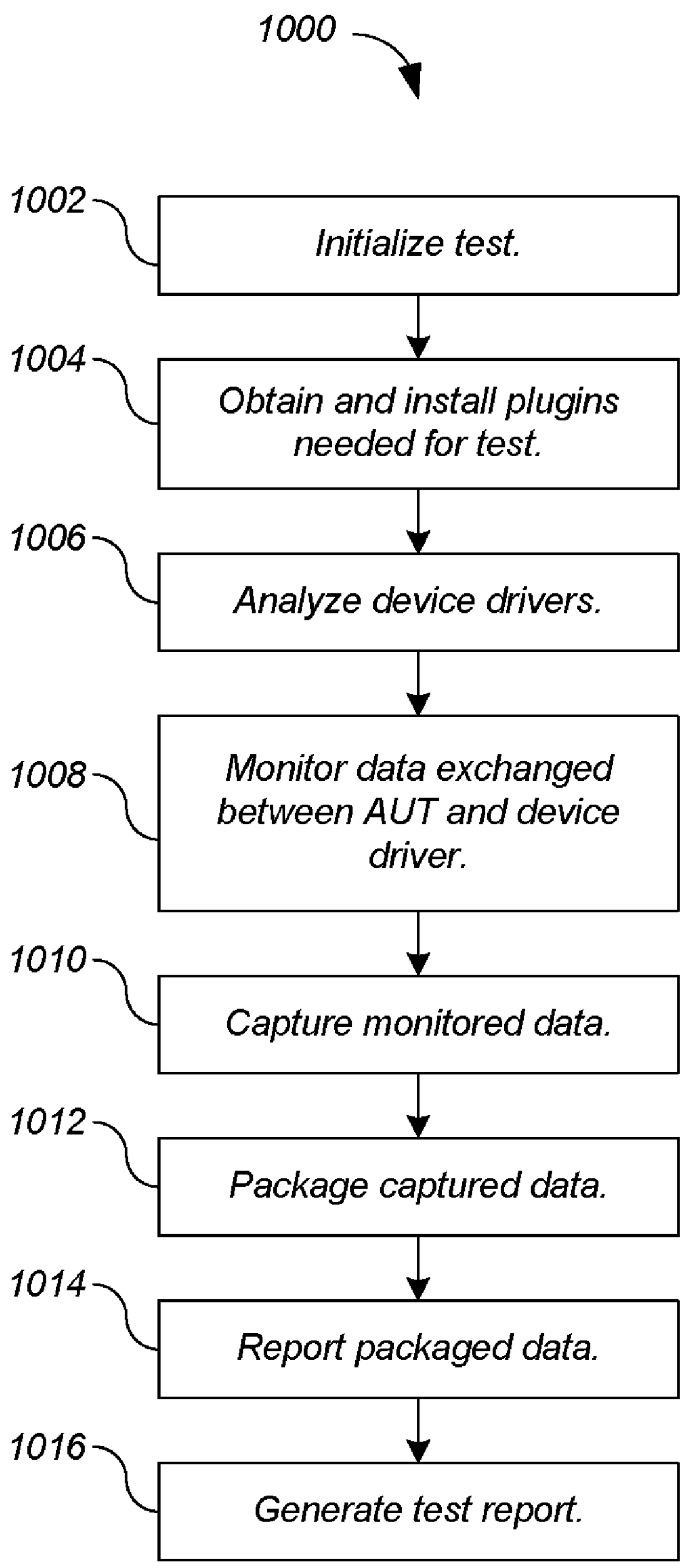

1000
1002
Initialize test.
1004
Obtain and install plugins needed for test.
1006
Analyze device drivers.
1008
Monitor data exchanged between AUT and device driver.
1010
Capture monitored data.
1012
Package captured data.
1014
Report packaged data.
1016
Generate test report.

FIG. 10

SOFTWARE PLUGIN MODULES FOR DEVICE TESTING

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation-in-part of U.S. Non-Provisional patent application Ser. No. 11/685,592 filed Mar. 13, 2007, Ser. No. 11/685,600 filed Mar. 13, 2007, and Ser. No. 11/685,604 filed Mar. 13, 2007, the disclosures thereof incorporated by reference herein in their entirety.

BACKGROUND

[0002] The present invention relates generally to application software testing. More particularly, the present invention relates to collecting test data representing data exchanged between an application executing on a computer and a device driver executing on the computer.

[0003] Software testing is a critical but time-consuming task involving testers and developers. When a tester encounters a problem, the first step is to document the problem so a developer can reproduce the problem. If the developer cannot reproduce the problem, it is unlikely the problem will be fixed. When testers and developers are located in the same location, they at least have the opportunity to work together to try to find a way to reproduce the problem. Such interaction can help the developer but is time consuming, taking the tester away from the work of testing. When testers are located away from developers, it may be impossible for the developer to work efficiently with the tester to reproduce the problem.

[0004] In addition, the software under test is likely to interact with a number of external devices such as displays, printers, and the like. To ensure a thorough test, it is desirable to record the interactions between the software and the external devices.

SUMMARY

[0005] In general, in one aspect, the invention features computer-readable media embodying instructions executable by a computer, the media comprising: one or more plugins each adapted to execute on the computer, wherein each plugin comprises a monitor module adapted to monitor data exchanged between one or more applications executing on the computer and a respective device driver executing on the computer, and a capture module adapted to capture the data.

[0006] Some embodiments comprise a reporting module adapted to transmit the data to a further computer. In some embodiments, the reporting module adds one or more timestamps to the data. In some embodiments, the data includes device settings generated by one or more of the applications for the respective device driver. Some embodiments comprise a plugin manager adapted to perform operations upon the one or more plugins, wherein the operations are selected from the group consisting of installing the one or more plugins, removing the one or more plugins, updating the one or more plugins, enabling the one or more plugins, and disabling the one or more plugins. In some embodiments, the one or more device drivers comprise at least one of: a printer driver; a scanner driver; a video driver; an audio driver; and a network driver.

[0007] In general, in one aspect, the invention features an apparatus comprising: one or more plugins each adapted to execute on the computer, wherein each plugin comprises a monitor module adapted to monitor data exchanged between one or more applications executing on the computer and a respective device driver executing on the computer, and a capture module adapted to capture the data.

[0008] Some embodiments comprise a reporting module adapted to transmit the data to a further computer. In some embodiments, the reporting module adds one or more timestamps to the data. In some embodiments, the data includes device settings generated by one or more of the applications for the respective device driver. Some embodiments comprise a plugin manager adapted to perform operations upon the one or more plugins, wherein the operations are selected from the group consisting of installing the one or more plugins, removing the one or more plugins, updating the one or more plugins, enabling the one or more plugins, and disabling the one or more plugins. In some embodiments, the one or more device drivers comprise at least one of: a printer driver; a scanner driver; a video driver; an audio driver; and a network driver.

[0009] In general, in one aspect, the invention features an apparatus comprising: one or more plugin means for executing on the computer, wherein each plugin means comprises monitor means for monitoring data exchanged between one or more applications executing on the computer and a respective device driver executing on the computer, and capture means for capturing the data.

[0010] Some embodiments comprise reporting means for transmitting the data to a further computer. In some embodiments, the reporting means adds one or more timestamps to the data. In some embodiments, the data includes device settings generated by one or more of the applications for the respective device driver. Some embodiments comprise plugin manager means for performing operations upon the one or more plugin means, wherein the operations are selected from the group consisting of installing the one or more plugin means, removing the one or more plugin means, updating the one or more plugin means, enabling the one or more plugin means, and disabling the one or more plugin means. In some embodiments, the one or more device drivers comprise at least one of: a printer driver; a scanner driver; a video driver; an audio driver; and a network driver.

[0011] In general, in one aspect, the invention features a method comprising: executing one or more plugins on a computer, wherein each plugin corresponds to a device driver executing on the computer; monitoring data exchanged between one or more applications executing on the computer and a respective one of the device drivers, and capturing the data.

[0012] Some embodiments comprise transmitting the data to a further computer. Some embodiments comprise adding one or more timestamps to the data. In some embodiments, the data includes device settings generated by one or more of the applications for the respective device driver. Some embodiments comprise performing operations upon the one or more plugins, wherein the operations are selected from the group consisting of installing the one or more plugins, removing the one or more plugins, updating the one or more plugins, enabling the one or more plugins, and disabling the one or more plugins. In some embodiments, the one or more device drivers comprise at least one of: a printer driver; a scanner driver; a video driver; an audio driver; and a network driver.

[0013] The details of one or more implementations are set forth in the accompanying drawings and the description below. Other features will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

[0014] FIG. **1** shows a software application test system comprising a tester client in communication with a test server and a developer client over a communication channel according to some embodiments of the present invention.

[0015] FIG. **2** shows a process for the software application test system of FIG. **1** according to some embodiments of the present invention.

[0016] FIG. **3** shows a test data collection process for the software application test system of FIG. **1** according to some embodiments of the present invention.

[0017] FIG. **4** shows a bug report according to some embodiments of the present invention.

[0018] FIG. **5** shows a process for test data processing for the software application test system of FIG. **1** according to some embodiments of the present invention.

[0019] FIG. **6** shows a process for test data presentation for the software application test system of FIG. **1** according to some embodiments of the present invention.

[0020] FIG. **7** shows a screenshot of an example motion picture according to some embodiments of the present invention.

[0021] FIG. **8** shows a software architecture **800** for tester client **102** of FIG. **1** according to an embodiment of the present invention.

[0022] FIG. **9** shows a software test system **900** according to an embodiment of the present invention.

[0023] FIG. **10** shows a process **1000** for software test system **900** of FIG. **9** according to an embodiment of the present invention.

[0024] The leading digit(s) of each reference numeral used in this specification indicates the number of the drawing in which the reference numeral first appears.

DETAILED DESCRIPTION

[0025] As used herein, the terms "client" and "server" generally refer to an electronic device or mechanism, and the term "message" generally refers to an electronic signal representing a digital message. As used herein, the term "mechanism" refers to hardware, software, or any combination thereof. These terms are used to simplify the description that follows. The clients, servers, and mechanisms described herein can be implemented on any standard general-purpose computer, or can be implemented as specialized devices.

[0026] Embodiments of the present invention provide ways to collect test data representing the operation of application software, tester interactions with the software, and status of the computer executing the software, including interactions with external devices such as printers and the like. The computer status can include system statistics such as CPU load, memory consumption, disk space, and the like. The tester interactions can include mouse moves, keystrokes, screen updates, and the like. This information can be recorded to a running buffer, which is saved when desired, for example when a fault is detected, when the tester or developer elects, and the like.

[0027] Embodiments of the present invention also provide ways to present the recorded information in a form usable to developers of the software. According to some embodiments of the present invention, the recorded information is assembled into a motion picture. The motion picture can include video and audio of the tester, representations of tester input, the screen updates displayed to the tester, stripcharts of computer metrics such as CPU and memory utilization, communication channel usage, and the like, and video and audio of external devices such as printers, scanners and the like. The communication channel usage information can be collected by probes, such as network probes, that can be deployed locally and/or remotely.

[0028] In some embodiments, the recorded information is uploaded to a centralized server. The recorded information can be uploaded immediately, or saved for batch upload at a later time. A bug report can be generated that includes a link, such as a URL, to the recorded information. When a developer sees a new bug report, the developer can click on the URL to bring up a web browser page with an application playback window to show the motion picture. The developer can press a "play" button on the page to start playing the motion picture. A time marker can move across the page showing correspondence between elements of the motion picture, for example between the screen updates and the stripcharts, and the like. The developer can pause the playback, move the time marker to any point along the time line, and the like.

[0029] Many applications interact with remote devices such as printers, other computers, and the like, using communication channels such as IP channels, USB channels, and the like. In some embodiments, data probes are employed to collect information describing the traffic on the channels. The data probes are invisible to both the application and the remote computer or device while collecting test data representing conditions on the communication channel. A centralized server coordinates the data probes with the software running on the tester's computer. As before, a running buffer is kept of application interaction, local system statistics and now communication channel data. When a problem is encountered, the centralized server notifies the data probes. Information is then collected and packaged as before. When playing back the information, a developer can view the communication channel data at each time on the timeline and how it corresponds to the tester's interaction with the tested application.

[0030] FIG. **1** shows a software application test system **100** comprising a tester client **102** in communication with a test server **104** and a developer client **106** over a communication channel **108** according to some embodiments of the present invention. As used herein, the terms "client" and "server" generally refer to an electronic device or mechanism, and the term "message" generally refers to an electronic signal representing a digital message. As used herein, the term "mechanism" refers to hardware, software, or any combination thereof. These terms are used to simplify the description that follows. The clients, servers, and mechanisms described herein can be implemented on any standard general-purpose computer, or can be implemented as specialized devices. Furthermore, while some embodiments of the present invention are described with reference to a client-server paradigm, other embodiments employ other paradigms, such as peer-to-peer paradigms and the like.

[0031] In some embodiments, tester client **102** is also in communication with one or more remote devices **110** over communication channel **108**. Communication channel **108** can include one or more networks, including wide-area networks such as the Internet, local-area networks (LAN), and the like, as well as direct links such as USB and the like. While embodiments of the present invention are described with respect to network communications, they are equally applicable to other forms of data communications such as direct links and the like.

[0032] Remote device(s) **110** can include computer peripheral devices such as printers, scanners, and the like, as well as other computers and the like. In some embodiments, one or more remote capture devices **150** are deployed to capture video, audio, and the like of the operation of remote device **110**.

[0033] Tester client **102** includes a processor **112** to execute software applications including software under test (SUT) **114** and software flight recorder (SFR) **116**, which records the operation of SUT **114**. Tester client **102** also includes a communication circuit **118** to communicate over communication channel **108**, a storage device **120**, a user interface (UI) circuit **122** to communicate with UI hardware **124**, a display circuit **126** to transmit signals to a display device **128**, and a capture circuit **130** to receive signals from one or more capture devices **132**. UI hardware **124** can include a keyboard, mouse, and the like. Capture devices **132** can include video cameras, microphones, and the like.

[0034] Test server **104** includes a processor **134** to execute playback software (PBSW) **152**, an input circuit **136** in communication with communication channel **108**, and an output circuit **138** in communication with communication channel **108**. Developer client **106** can include a processor **140**, an input circuit **142** in communication with communication channel **108**, and a display circuit **144** in communication with a display device **146**. Software application test system **100** can also include one or more channel probes **148** to collect information describing traffic on communication channel **108**.

[0035] Although in the described embodiments, the elements of software application test system **100** are presented in one arrangement, other embodiments may feature other arrangements, as will be apparent to one skilled in the relevant arts based on the disclosure and teachings provided herein. For example, the elements of software application test system **100** can be implemented in hardware, software, or combinations thereof.

[0036] FIG. **2** shows a process **200** for the software application test system **100** of FIG. **1** according to some embodiments of the present invention. Although in the described embodiments, the elements of process **200** are presented in one arrangement, other embodiments may feature other arrangements, as will be apparent to one skilled in the relevant arts based on the disclosure and teachings provided herein. For example, in various embodiments, some or all of the steps of process **200** can be executed in a different order, concurrently, and the like.

[0037] Referring to FIG. **2**, process **200** begins with configuration of software application test system **100** (step **202**). For example, communication connections are established between tester client **102** and test server **104** over communication channel **108**. The tester logs into the system, and downloads and installs the latest version of SFR **116** if needed. Then process **200** begins test data collection (step **204**).

[0038] FIG. **3** shows a test data collection process **300** for the software application test system **100** of FIG. **1** according to some embodiments of the present invention. Although in the described embodiments, the elements of process **300** are presented in one arrangement, other embodiments may feature other arrangements, as will be apparent to one skilled in the relevant arts based on the disclosure and teachings provided herein. For example, in various embodiments, some or all of the steps of process **300** can be executed in a different order, concurrently, and the like.

[0039] Referring to FIG. **3**, processor **112** of tester client **102** executes software flight recorder (SFR) **116** (step **302**). Processor **112** of tester client **102** also executes software under test (SUT) **114** (step **304**). SFR **116** and/or SUT **114** can be launched automatically by tester client **102**, for example when booting, under the control of a tester using tester client **102**, remotely under the control of a developer using developer client **106**, and the like.

[0040] As the tester interacts with SUT **114** using UI hardware **124** and display device **128**, user interface circuit **122** receives UI signals representing the actions of the tester, and SFR **116** collects test data representing the UI signals (step **306**). Display circuit **126** generates display signals representing a display produced in accordance with SUT **114**, and SFR **116** collects test data representing the display signals (step **308**).

[0041] In some embodiments, capture circuit **130** of tester client **102** captures signals from one or more capture devices such as video cameras and the like, and SFR **116** collects test data representing the capture signals (step **310**). Capture devices **132** can be used to generate capture signals representing video and audio of the tester and local devices such as local printers and the like. For example, the tester can create voice notes during the test.

[0042] Communication circuit **118** exchanges communication signals over communication channel **108** in accordance with SUT **114**, and SFR **116** collects test data representing the communication signals (step **312**). In some embodiments, one or more channel probes **148** also collect data representing communication signals at remote locations, and SFR **116** collects test data representing the probed communication signals (step **314**). For example, in embodiments where tester client **102** interacts with a remote device **110**, a channel probe **148** can be deployed in communication channel **108** at or near the remote device **110**. The probed data can represent traffic levels on communication channel **108** and the like.

[0043] In embodiments including one or more remote devices **110**, one or more remote capture devices **150** can collect remote capture signals, such as audio, video, and the like, of remote device **110**, and SFR **116** can collect test data representing the remote capture signals (step **316**). In some embodiments, these remote capture signals can be collected by test server **104**.

[0044] SFR **116** also collects test data representing operation of tester client **102**. For example, SFR **116** can collect test data representing utilization of processor **112** (step **318**) and utilization of storage device **120** (step **320**).

[0045] In some embodiments, SFR **116** collects the data described above, for example in a circular buffer or the like, until a test event occurs (step **322**). The test event can be automatically generated, for example as a fault of SUT **114**, generated manually by the tester or developer, or in other ways. When a test event occurs, SFR **116** transfers the collected test data to test server **104** (step **324**). The transfer can occur immediately after the test event, at a later time in a batch transfer mode, for example with test data collected at other times, and the like. In some embodiments, the tester fills out a test report which is transferred as part of the test data. The test report can include a brief description of the problem and the like.

[0046] In some embodiments, test server **104** generates a motion picture representing the test data, as described in detail below. In other embodiments, the motion picture is generated by SFR **116**, and can be transferred to test server **104**.

[0047] In some embodiments, SFR **116** also generates a bug report including a link to the location on test server **104**

where the corresponding test data and/or the corresponding motion picture is stored (step **326**). SFR **116** can transfer the bug report to the developer at developer client **106**, to a bug database, and the like. In some embodiments, the link is created after the test data is uploaded to test server **104**. For example, in embodiments where the motion picture is generated and stored by test server **104**, test server **104** then creates the link to the motion picture.

[0048] FIG. **4** shows a bug report **400** according to some embodiments of the present invention. Bug report **400** includes the date **402**, the name of the tester **404**, the operating system **406** used by SUT **114**, and a problem description **408** including the link **410**.

[0049] Referring again to FIG. **2**, after test data collection (step **204**), process **200** processes the collected test data (step **206**). While in the described embodiments, the collected test data is processed by test server **104**, in other embodiments the test data is processed elsewhere. For example, the test data can be processed by tester client **102**, thereby eliminating the need for test server **104**.

[0050] FIG. **5** shows a process **500** for test data processing for the software application test system **100** of FIG. **1** according to some embodiments of the present invention. Although in the described embodiments, the elements of process **500** are presented in one arrangement, other embodiments may feature other arrangements, as will be apparent to one skilled in the relevant arts based on the disclosure and teachings provided herein. For example, in various embodiments, some or all of the steps of process **500** can be executed in a different order, concurrently, and the like.

[0051] Referring to FIG. **5**, input circuit **136** of test server **104** receives, over communication channel **108** from tester client **102**, test data collected over a test interval (step **502**). The test data can be transferred as one or more data files. For example, the test data for a test interval can include a data file representing operation of tester client **102** during the test interval, a data file representing screen updates generated by tester client **102** during the test interval, and a data file representing traffic exchanged over communication channel **108** by tester client **102** during the test interval. Test server **104** stores the files until a developer requests a playback of the test interval (step **504**), for example by activating link **410** in bug report **400**.

[0052] When requested (step **504**), processor **134** of test server **104** generates one or more data streams representing the test data (step **506**). For the previous example, the data streams can include a data stream representing operation of tester client **102** during the test interval, a data stream representing screen updates generated by tester client **102** during the test interval, and a data stream representing traffic exchanged over communication channel **108** by tester client **102** during the test interval.

[0053] Output circuit **138** transmits the data stream(s) over communication channel **108** (step **508**) so that the data streams are synchronized when transmitted by output circuit **138**. The synchronized data streams are used by developer client **106** to present a motion picture representing the test data for the test interval to the developer, as described in detail below. In some embodiments, the data streams are generated based on the test data when requested by a developer. In other embodiments, data stream files are created before receiving a request, and then the data stream files are streamed to the developer after the request is received.

[0054] In some embodiments, instead of generating and transmitting synchronized data streams to developer client **106**, test server **104** generates one or more playback data files based on the test data files, and transfers the playback data files to developer client **106**. For example, test server **104** generates one or more motion picture files, which are assembled, synchronized and played together as a single motion picture by developer client **106**, as described in detail below.

[0055] Referring again to FIG. **2**, after processing the collected test data (step **206**), process **200** presents the processed test data (step **208**). While in the described embodiments, the test data is presented by developer client **106**, in other embodiments the test data is presented elsewhere. For example, the test data can be presented by tester client **102**, thereby eliminating the need for developer client **106**.

[0056] FIG. **6** shows a process **600** for test data presentation for the software application test system **100** of FIG. **1** according to some embodiments of the present invention. Although in the described embodiments, the elements of process **600** are presented in one arrangement, other embodiments may feature other arrangements, as will be apparent to one skilled in the relevant arts based on the disclosure and teachings provided herein. For example, in various embodiments, some or all of the steps of process **600** can be executed in a different order, concurrently, and the like.

[0057] Referring to FIG. **6**, input circuit **142** of developer client **106** receives playback data for a test interval over communication channel **108** (step **602**). For example, the playback data can represent operation of tester client **102** during the test interval, screen updates generated by tester client **102** during the test interval, and traffic exchanged over communication channel **108** by tester client **102** during the test interval. The playback data can be received in the form of synchronized data streams, data files, and the like.

[0058] Processor **140** of developer client **106** executes playback software (PBSW) **152**, which generates a motion picture representing the test interval based on the playback data (step **604**). For example, according to some embodiments of the present invention, the motion picture contemporaneously includes a stripchart area showing a stripchart representing the operation of tester client **102** and traffic exchanged over communication channel **108** by tester client **102**, and a screen update area showing screen updates for tester client **102**, where the stripchart area and the screen update area are synchronized. Playback software **152** displays the motion picture on display device **146** (step **606**). In particular, display circuit **144** of developer client **106** generates a display signal representing the motion picture, which is rendered as a display by display device **146**.

[0059] FIG. **7** shows a screenshot **700** of an example motion picture according to some embodiments of the present invention. Screenshot **700** includes a screen update area **702** to show screen updates for tester client **102** during the test interval, a video area **704** to present video and/or audio of a remote device **110** (in this case, an all-in-one printer) during the test interval, and a stripchart area **706** showing three stripcharts **706**A-C representing test data collected during the test interval. Stripchart **706**A represents utilization of processor **112** of tester client **102** during the test interval. Stripchart **706**B represents utilization of storage device **120** of tester client **102** during the test interval. Stripchart **706**C represents traffic on communication channel **108** during the test interval. In other embodiments, the motion picture can present other sorts of test data, as will be apparent to one skilled in the relevant arts based on the disclosure and teachings provided herein.

[0060] In the motion picture, the test data presented is synchronized. For the example of FIG. **7**, screen update area **702**, video area **704**, and stripchart area **706** are synchronized with each other so that, at any moment, the motion picture presents test data that occurred contemporaneously. Some embodiments include a common timeline **708** and control buttons **710** that can be manipulated to control playback of the motion picture. The motion picture can also include a test name **712** and test information **714** such as a bug number, test case number, tester name, and software version number. The motion picture can also include links **716** to tester voice notes, system configuration, and the like.

[0061] Embodiments of the present invention provide software plugin modules for device testing. One or more plugins and a plugin manager are provided. The plugins interact with an application under test (AUT) executing on a computer. The AUT can be any sort of application, such as a word processing application, a presentation application, a diagramming application, and the like. Each plugin is adapted to monitor data exchanged between the AUT and a respective device driver executing on the computer. The device drivers can include printer drivers, scanner drivers, video drivers, audio drivers, network drivers, and the like. The data can include device settings generated by the AUT for the respective device driver and the like. Each plugin captures the monitored data, which can be reported to a remote test server.

[0062] FIG. **8** shows a software architecture **800** for tester client **102** of FIG. **1** according to an embodiment of the present invention. Although in the described embodiments, the elements of software architecture **800** are presented in one arrangement, other embodiments may feature other arrangements, as will be apparent to one skilled in the relevant arts based on the disclosure and teachings provided herein.

[0063] Referring to FIG. **8**, software architecture **800** includes a software flight recorder (SFR) client **802** in communication with a platform application programming interface (API) **804**. Platform API **804** can be implemented as a computer operating system and the like. SFR client **802** includes a SFR plugin manager **806**, a SFR client core API **808**, and one or more device plugins **810**A-N. Each device plugin **810** is designed to interact with a device driver to monitor and capture data exchanged between one or more applications under test (AUT) and the device driver, as described below.

[0064] SFR client core API **808** provides an interface for plugins **810** to interact with SFR client **802**. For example, plugins **810** use SFR client **802**, via SFR client core API **808**, to send collected information to other computers, such as test server **104** and developer client **106** of FIG. **1**. Depending on the requirements of the test, test server **104** can install, uninstall, update, enable, or disable the needed plugins **810**.

[0065] SFR plugin manager **806** provides a mechanism for manipulating plugins **810**. For example, plugin manager **806** can perform various operations upon plugins **810** such as installing, removing, updating, enabling, disabling, and the like.

[0066] FIG. **9** shows a software test system **900** according to an embodiment of the present invention. Although in the described embodiments, the elements of software architecture **800** are presented in one arrangement, other embodiments may feature other arrangements, as will be apparent to one skilled in the relevant arts based on the disclosure and teachings provided herein.

[0067] According to embodiments of the present invention, each plugin **810** installs a monitor module **910** to monitor data exchanged between the application under test and the respective device driver. The device drivers can include printer drivers, scanner drivers, video drivers, audio drivers, network drivers, and the like, as is well-known in the relevant arts.

[0068] Referring to FIG. **9**, a printer plugin **810** has installed a monitor module **910** between an application under test (AUT) **902** and a printer driver **906**. Printer driver **906** is in communication with a printer **904**. AUT **902** can be any sort of application, for example such as a word processor and the like.

[0069] Each plugin **810** also includes a capture module **912** to capture the monitored data. Referring again to FIG. **9**, printer plugin **810** includes a capture module **912** that is in communication with SFR client **802** via SFR client core API **808**. SFR client **802** includes a reporting module **908** that is in communication with SFR test server **104** of FIG. **1**, for example over channel **108** of FIG. **1**.

[0070] FIG. **10** shows a process **1000** for software test system **900** of FIG. **9** according to an embodiment of the present invention. Although in the described embodiments, the elements of process **900** are presented in one arrangement, other embodiments may feature other arrangements, as will be apparent to one skilled in the relevant arts based on the disclosure and teachings provided herein. For example, in various embodiments, some or all of the steps of process **900** can be executed in a different order, concurrently, and the like. As another example, while process **100** is described for a printer driver, process **1000** is easily applied to any sort of device driver, as will be apparent to one skilled in the relevant arts based on the disclosure and teachings provided herein.

[0071] Referring to FIG. **10**, a test is initialized (step **1002**). For example, referring to FIG. **1**, a test administrator can initialize the test from test server **104**, developer client **106**, and the like. Alternatively, the test can be initialized by a tester from tester client **102**. The initialization of the test specifies the AUT(s) **902** and devices, such as printer **904**, to be tested.

[0072] Based on the test initialization, SFR plugin manager **806** obtains and installs the plugins **810** needed for the test (step **1004**). The test administrator can configure a bundle of plugins **810** needed for each test cycle. When the tester logs into test server **104**, test server **104** can provide any needed plugins. If a plugin **810** is already installed, test server **104** can ensure that the plugin **810** is the correct version. In addition, test server **104** can enable or disable plugins **810** that are already installed. As part of the installation of each plugin **810**, a monitor module **910** is inserted between AUT **902** and the respective device driver **906**, and a capture module **912** is linked to SFR client core API **808**, for example as shown in FIG. **9**.

[0073] Before the testing starts, an analysis program can execute to interrogate each driver, probing for all possible options that can be selected by the tester (step **1006**). For example, the analysis program can generate a tree of printer driver options to determine all the possible combinations of options the tester can select. This information is processed and stored on test server **104**. Test server **104** then uses this information to match the information sent by printer plugin **810**. For example, this information can be used to map the options for each test case to the total number of options to determine what options have been tested and what options still need to be tested.

[0074] During testing, monitor module **910** monitors data exchanged between AUT **902** and printer driver **906** (step **1008**). For example, the data can include the settings generated by AUT **902** for device driver **906**, data sent by AUT **902** to printer **904**, data sent by printer **904** to AUT **902**, and the like. In the current printer example, the driver settings can include paper type, paper size, print quality, number of copies, ink levels, print job time, and the like, for each print job. Of course the data collected can differ for different devices and device drivers **906**.

[0075] Capture module **912** captures the data collected by the respective monitor module (step **1010**), and passes the captured data to reporting module **908** of SFR client **802** via SFR client core API **808**. Reporting module **908** packages the captured data (step **1012**), and transmits the packaged data to test server **104** (step **1014**). For example, the captured data can be packaged into a file with timestamps to facilitate synchronization with other data captured during the test such as data captured by other plugins **810**, screen updates displayed to the tester, stripcharts of computer metrics such as CPU and memory utilization, communication channel usage, and the like, and video and audio of external devices such as printers, scanners and the like.

[0076] Based on the reported data, test server **104** can generate test reports such as graphical reports, motion pictures of the test, as described above, and the like (step **1016**). Test server **104** can generate interim test reports during the test, as well as final test reports after conclusion of the test.

[0077] The invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Apparatus of the invention can be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor; and method steps of the invention can be performed by a programmable processor executing a program of instructions to perform functions of the invention by operating on input data and generating output. The invention can be implemented advantageously in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. Each computer program can be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired; and in any case, the language can be a compiled or interpreted language. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory and/or a random access memory. Generally, a computer will include one or more mass storage devices for storing data files; such devices include magnetic disks, such as internal hard disks and removable disks; magneto-optical disks; and optical disks. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM disks. Any of the foregoing can be supplemented by, or incorporated in, ASICs (application-specific integrated circuits).

[0078] A number of implementations of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. Accordingly, other implementations are within the scope of the following claims.

What is claimed is:

1. Computer-readable media embodying instructions executable by a computer, the media comprising:

one or more plugins each adapted to execute on the computer, wherein each plugin comprises

a monitor module adapted to monitor data exchanged between one or more applications executing on the computer and a respective device driver executing on the computer, and

a capture module adapted to capture the data.

2. The computer-readable media of claim **1**, further comprising:

a reporting module adapted to transmit the data to a further computer.

3. The computer-readable media of claim **2**:

wherein the reporting module adds one or more timestamps to the data.

4. The computer-readable media of claim **1**:

wherein the data includes device settings generated by one or more of the applications for the respective device driver.

5. The computer-readable media of claim **1**, further comprising:

a plugin manager adapted to perform operations upon the one or more plugins, wherein the operations are selected from the group consisting of

installing the one or more plugins,

removing the one or more plugins,

updating the one or more plugins,

enabling the one or more plugins, and

disabling the one or more plugins.

6. The computer-readable media of claim **1**, wherein the one or more device drivers comprise at least one of:

a printer driver;

a scanner driver;

a video driver;

an audio driver; and

a network driver.

7. An apparatus comprising:

one or more plugins each adapted to execute on the computer, wherein each plugin comprises

a monitor module adapted to monitor data exchanged between one or more applications executing on the computer and a respective device driver executing on the computer, and

a capture module adapted to capture the data.

8. The apparatus of claim **7**, further comprising:

a reporting module adapted to transmit the data to a further computer.

9. The apparatus of claim **8**:

wherein the reporting module adds one or more timestamps to the data.

10. The apparatus of claim **7**:

wherein the data includes device settings generated by one or more of the applications for the respective device driver.

11. The apparatus of claim **7**, further comprising:
a plugin manager adapted to perform operations upon the one or more plugins, wherein the operations are selected from the group consisting of
installing the one or more plugins,
removing the one or more plugins,
updating the one or more plugins,
enabling the one or more plugins, and
disabling the one or more plugins.

12. The apparatus of claim **7**, wherein the one or more device drivers comprise at least one of:
a printer driver;
a scanner driver;
a video driver;
an audio driver; and
a network driver.

13. An apparatus comprising:
one or more plugin means for executing on the computer, wherein each plugin means comprises
monitor means for monitoring data exchanged between one or more applications executing on the computer and a respective device driver executing on the computer, and
capture means for capturing the data.

14. The apparatus of claim **13**, further comprising:
reporting means for transmitting the data to a further computer.

15. The apparatus of claim **14**:
wherein the reporting means adds one or more timestamps to the data.

16. The apparatus of claim **13**:
wherein the data includes device settings generated by one or more of the applications for the respective device driver.

17. The apparatus of claim **13**, further comprising:
plugin manager means for performing operations upon the one or more plugin means, wherein the operations are selected from the group consisting of
installing the one or more plugin means,
removing the one or more plugin means,
updating the one or more plugin means,
enabling the one or more plugin means, and
disabling the one or more plugin means.

18. The apparatus of claim **13**, wherein the one or more device drivers comprise at least one of:
a printer driver;
a scanner driver;
a video driver;
an audio driver; and
a network driver.

19. A method comprising:
executing one or more plugins on a computer, wherein each plugin corresponds to a device driver executing on the computer;
monitoring data exchanged between one or more applications executing on the computer and a respective one of the device drivers, and
capturing the data.

20. The method of claim **19**, further comprising:
transmitting the data to a further computer.

21. The method of claim **20**:
adding one or more timestamps to the data.

22. The method of claim **19**:
wherein the data includes device settings generated by one or more of the applications for the respective device driver.

23. The method of claim **19**, further comprising:
performing operations upon the one or more plugins, wherein the operations are selected from the group consisting of
installing the one or more plugins,
removing the one or more plugins,
updating the one or more plugins,
enabling the one or more plugins, and
disabling the one or more plugins.

24. The method of claim **19**, wherein the one or more device drivers comprise at least one of:
a printer driver;
a scanner driver;
a video driver;
an audio driver; and
a network driver.

* * * * *