February 04, 2016

TRACKING APPLICATION CRASHES

Kamakshi Kodur

Karin Lundberg

Michael Thomas Baxley

TRACKING APPLICATION CRASHES

ABSTRACT

An application testing system can be utilized to track, store, and report application crashes. The application can be a web browser application. The system receives a list of uniform resource locators (URLs) for testing. The system performs a set of actions (a test) at a URL from the list of URLs. The system detects if a crash occurs at the URL. If the crash occurs at the URL, then the system stores the URL and information associated with the crash. The system then performs another set of actions at a next URL from the list of URLs. The system stops the tests if a threshold number of crashes have occurred or each URL in the list of URLs is tested. Finally, the system generates a report for each crash that occurred at a URL for all the tested URLs.

PROBLEM STATEMENT

An application, such as a web browser, running on a client device can encounter crashes. If the application crashes a high number of times, then the end-user using the client device may experience problems using the application and become dissatisfied. Testing prior to application release to an end user, including running certain event sequences on the application, and fixing any occurring issues may improve customer satisfaction.

There are various testing frameworks for testing applications. For example, monkey testing can test for stability by providing a way to run random input events on an application. Also, many frameworks for testing applications usually stop after a crash or restart the test from

the beginning instead of continuing from where the last crash occurred. Therefore, there are opportunities for alternate testing frameworks.

DETAILED DESCRIPTION

The systems and techniques described in this disclosure relate to an application testing system that tracks, stores, and reports application crashes. The system can be implemented for use in an Internet, an intranet, or another client and server environment. The system can be implemented locally on a client device or implemented across a client device and server environment. The client device is a testing device that may have a test-script software installed at the device. The testing device sends a performance report of the software tested under various conditions to the server for analysis. The testing device can be any electronic device such as a computer, a laptop, a mobile device, a smartphone, a tablet, a handheld electronic device, etc.

Fig. 1 illustrates an example method 100 that performs tests on a list of uniform resource locators (URLs) at a software application, for example, a web browser application. The method is implemented by the application testing system which runs on the testing device. The system receives 110 a list of URLs for testing. The list may be stored at a memory of the testing device which runs the application.

The system then performs 120 a set of actions at a URL, for example, a first URL, from the list of URLs. The set of actions can be, for example, scrolling, taking a screenshot, clicking on buttons, typing text, and clicking on links. The set of actions may be selected from a predetermined list of actions stored at the memory of the testing device. The predetermined list may be downloaded while installing a test-script software at the testing device. The set of actions

performed may be selected randomly per a monkey test or in a particular sequence from the predetermined list.

The system detects 130 if a crash occurs at the first URL upon performing a particular action. A crash may occur when the application stops functioning properly, for example, when a failure or a memory warning occurs. The system stores 140 the first URL and information associated with the crash. The information associated with the crash may be, for example, type of action that led to the crash, type of crash, time of crash, and recorded screenshot of the webpage when the crash occurred. If no crash is detected 130, the system proceeds to step 150.

The system detects 150 if a threshold number of crashes have occurred or each URL in the list of URLs is tested. If no, then the system repeat steps from 120. The system then proceeds to a next URL, for example, a second URL, and repeats steps from 120. The system may perform 120 another set of actions at the second URL. The another set of actions may be same or different from the set of actions performed at the first URL. Thus, the system performs a set of test actions at each URL, from the list of URLs, in a sequential manner.

If the detection 150 step results in a yes, then the system generates 160 a report for each crash at a URL. The report may have information specifying which URLs caused crashes, failures, or memory warnings. The report may also quote differences between the screenshots of the recorded web pages and live web pages. The system generates and stores the report at the memory of the testing device. The report may be sent to a host device that may perform an analysis on the report. Host device may be a server, a computer, a laptop, etc.

Fig. 2 is a block diagram of an exemplary environment that shows components of a system for implementing the techniques described in this disclosure. The environment includes

testing device(s) 210, server(s) 230, and network 240. Network 240 connects testing device(s) 210 to server(s) 230. Testing device 210 is an electronic device. Testing device 210 may be capable of requesting and receiving data/communications over network 240. Example testing devices 210 are personal computers (e.g., laptops), mobile communication devices, (e.g. smartphones, tablet computing devices), set-top boxes, game-consoles, embedded systems. Testing device 210 may execute an application, such as a web browser 212 or 214 or a native application 216. Web applications 213 and 215 may be displayed via a web browser 212 or 214. Server 230 may be a web server capable of sending, receiving and storing web pages 232. Web page(s) 232 may be stored on or accessible via server 230. Web page(s) 232 may be associated with web application 213 or 215 and accessed using a web browser, e.g., 212. When accessed, webpage(s) 232 may be transmitted and displayed on a testing device, e.g., 210. Resources 218 and 218' are resources available to the testing device 210 and/or applications thereon, or server(s) 230 and/or web page(s) accessible therefrom, respectively. Resources 218' may be, for example, memory or storage resources; a text, image, video, audio, JavaScript, CSS, or other file or object; or other relevant resources. Network 240 may be any network or combination of networks that can carry data communication.

The subject matter described herein can be implemented in software and/or hardware (for example, computers, circuits, or processors). The subject matter can be implemented on a single device or across multiple devices (for example, a client device and a server device). Devices implementing the subject matter can be connected through a wired and/or wireless network. Such devices can receive inputs from a user (for example, from a mouse, keyboard, or touchscreen) and produce an output to a user (for example, through a display and/or a speaker). Specific

examples disclosed are provided for illustrative purposes and do not limit the scope of the disclosure.

DRAWINGS

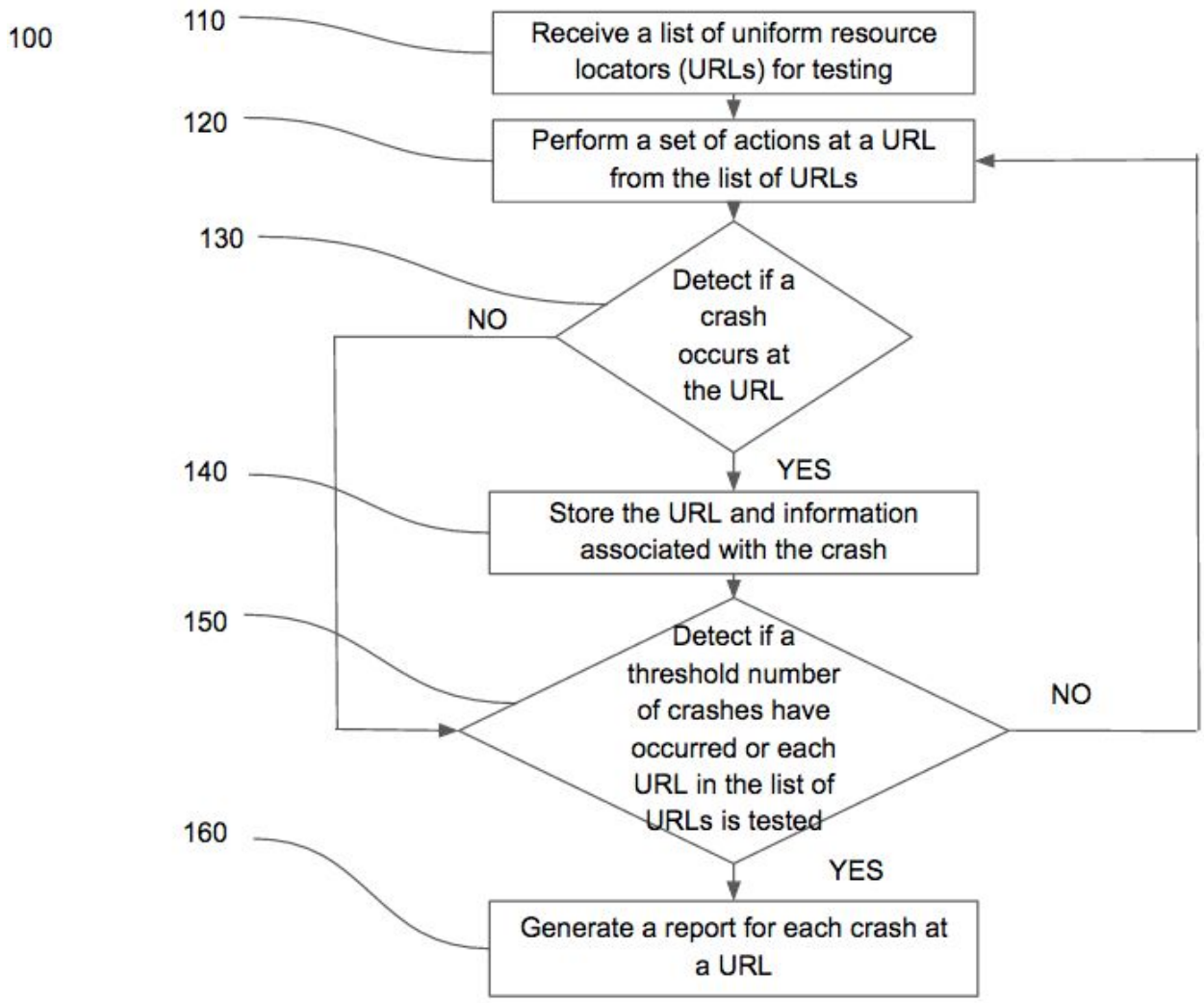

Fig.1

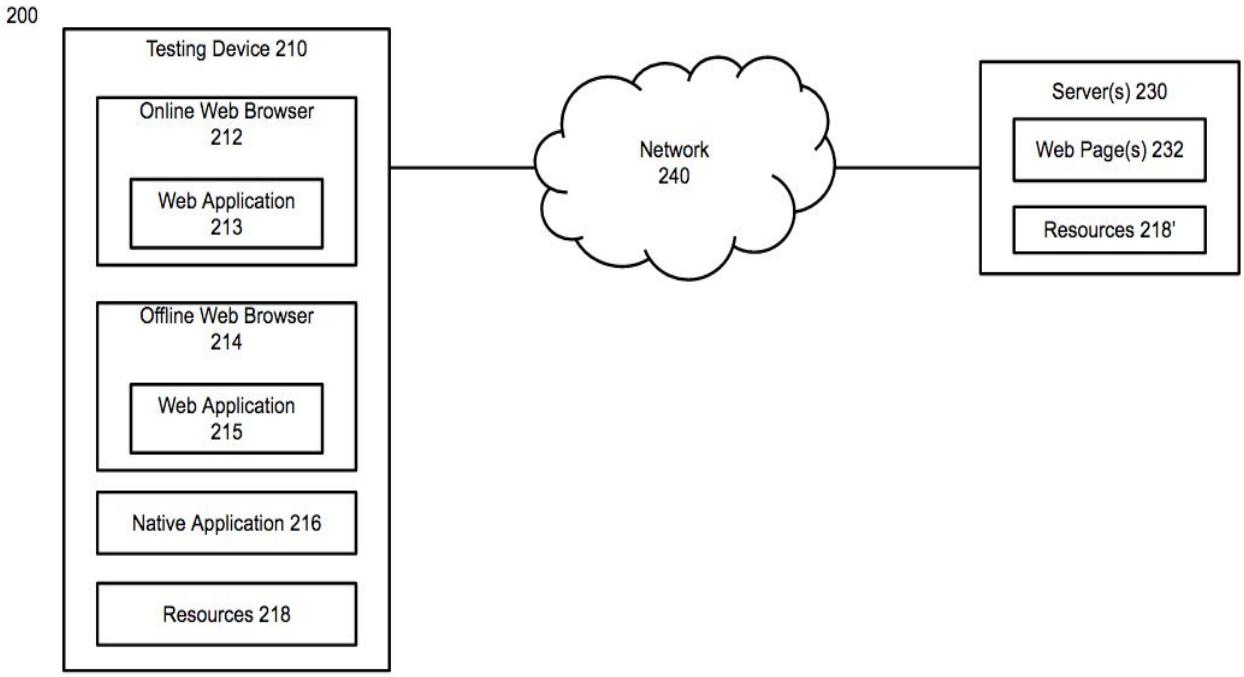
200
Testing Device 210
Online Web Browser 212
Web Application 213
Offline Web Browser 214
Web Application 215
Native Application 216
Resources 218
Network 240
Server(s) 230
Web Page(s) 232
Resources 218'

Fig. 2