Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Crawl-based analysis of web applications: Prospects and challenges

Arie van Deursen [a,*], Ali Mesbah [b], Alex Nederlof [a]

[a] Delft University of Technology, The Netherlands
[b] University of British Columbia, Canada

A R T I C L E   I N F O

Article history:
Received 15 September 2014
Accepted 15 September 2014
Available online 28 September 2014

Keywords:
Test automation
Web crawling
Software evolution

A B S T R A C T

In this paper we review five years of research in the field of automated crawling and testing of web applications. We describe the open source Crawljax tool, and the various extensions that have been proposed in order to address such issues as cross-browser compatibility testing, web application regression testing, and style sheet usage analysis. Based on that we identify the main challenges and future directions of crawl-based testing of web applications. In particular, we explore ways to reduce the exponential growth of the state space, as well as ways to involve the human tester in the loop, thus reconciling manual exploratory testing and automated test input generation. Finally, we sketch the future of crawl-based testing in the light of upcoming developments, such as the pervasive use of touch devices and mobile computing, and the increasing importance of cyber-security.

Personal Message to Paul Klint, from Arie van Deursen

From 1990–1994 and 1996–2005 I had a great time working in the research group headed by Paul Klint at CWI. The work on Crawljax described in this paper mostly dates from after my period at CWI. Nevertheless, the success of Crawljax owes a lot to Paul.

Paul has set an example to many by his enthusiasm for programming. Paul is always programming: in Spring and in Summer, in Lisp, in ASF, in ASF+SDF, in C, in ToolBus script, in Java, and, these days, in Rascal. Even this week (early August 2013), while many of us are secretly contributing to this special issue devoted to him, Paul committed to `GitHub` *every* day.

It is this enthusiasm for programming that has inspired many of his students and co-workers. Thank you Paul for great times at CWI: Your influence goes well beyond the papers you have written. The Software Engineering Research Group at Delft University of Technology has been shaped by your approach to research.

1. Introduction

Modern society critically depends on highly interactive web applications, which hence must be reliable, maintainable, and secure. Unfortunately, the increasing complexity of today's web applications poses substantial challenges into their dependability.

* Corresponding author.
E-mail addresses: arie.vandeursen@tudelft.nl (A. van Deursen), amesbah@ece.ubc.ca (A. Mesbah), alex@nederlof.com (A. Nederlof).

http://dx.doi.org/10.1016/j.scico.2014.09.005
0167-6423/© 2014 Elsevier B.V. All rights reserved.

While static analysis of client and server code of web applications can provide valuable insight in their dependability, the highly dynamic nature of today's client-side (JavaScript) code makes dynamic analysis indispensable.

One of the key technologies facilitating these dynamic web applications is Ajax,[1] an acronym for "Asynchronous JavaScript and XML". With Ajax, web-browsers not only offer the user navigation through a sequence of HTML pages, but also responsive rich interaction via graphical user interface components by means of asynchronous processing.

While the use of Ajax technology positively affects user-friendliness and interactiveness of web applications [1], it comes at a price: Ajax applications are notoriously error-prone due to, e.g., their stateful, asynchronous, and event-based nature, the use of (loosely typed) JavaScript, the client-side manipulation of the browser's Document-Object Model (DOM), and client-server communication based on deltas rather than the exchange of full pages [1].

In our research during the past five years we have gained considerable experience with the use of *crawl-based* dynamic analysis of web applications [2]. In particular, we have developed Crawljax,[2] a tool that can click through an arbitrary web application in order to build up a model of the potential user interactions [3,4]. Subsequently, this model can be validated against *invariants*, expressing desirable properties (such as the use of valid HTML code only) the system under test should have at any state [5,6].

The goal of this paper is to explore the prospects and challenges of crawl-based analysis. To that end, we first provide a brief survey of related work, covering our own Crawljax work as well as work by others. Based on that survey, we subsequently explore some of the key open problems in crawl-based analysis, laying out avenues for further research.

2. Crawling interactive web applications

2.1. Challenges

Web crawlers are almost as old as the World Wide Web itself. The first crawler was implemented by Matthey Gray in the spring of 1993. It was called the "Wanderer" and its goal was to measure the size of the web.[3] Soon after that in 1994, the first crawlers that indexed the web appeared [7].

As the web evolved it became less about document sharing and more about interactive content to even full-blown applications. JavaScript, the dominant browser language, can dynamically generate or load content. Because of this, crawling the web by just following links is not sufficient anymore [8]. To be able to crawl and fetch the dynamic content of a web application, a crawler has to interact with JavaScript in the browser.

With JavaScript-enabled crawling, the result of a crawl is a model of the user interaction: A click on some element in the browser can bring the web application in a given state, and exhaustively attempting to execute all possible clicks builds a model of the ways in which a user can interact with the application.

This introduces a number of challenges:

State Explosion: Any click can result in a new state. Even a small web application can have an infinite number of states (think of a simple TODO-list application with states for every possible todo item). Furthermore, content may be time based, or may differ per visitor.

State Navigation: Even though browsers have page forward and backward functionality build in, this is only tied to the application state if the developers choose to. And even if they do, it is a cumbersome error-prone task. This is why web applications often have a different state model than the one that can be derived from the URLs, making the navigation hard to automate [9]. This means that crawlers cannot expect to go to the previous state when they press the back button. They need a more robust system of navigating through the application.

Triggering State Changes: State changes can be caused by many kinds of events in a web page. Clickables are not limited to `<a href="example.com">` elements. JavaScript allows one to add a click handler to practically any HTML element. Besides clicks, other events may cause a state change, such as hover, mouse-in, mouse-out, drag and drop, double click and right click, as well as touch and touch-gesture events for tablets and smartphones.

To reach all possible states, the crawler could invoke all possible events on all possible elements. But even then the combination of those elements might be the key to going to the next state. For example, some applications have special states for when a user holds a keyboard key and then click an element. The challenge for crawlers is to either try many of these combinations, or to be smart and discover which elements are listened to by JavaScript. Although finding which elements in JavaScript have listeners is possible, this does not cover the case of input combinations.

Unreachable States: The term "The Deep Web" comes from the traditional crawlers meaning the part of the web that cannot be found by following links [8,10]. Although JavaScript-enabled crawlers can find more, they face some of the

[1] Jesse Garret, "Ajax: A New Approach to Web Applications", February 18, 2005, http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications.

[2] http://crawljax.com/.

[3] http://www.mit.edu/people/mkgray/growth/.