RAW: Runtime Automatic Workarounds*

Antonio Carzaniga†, Alessandra Gorla†, Nicolò Perino†, and Mauro Pezzè†‡
{antonio.carzaniga|gorlaa|perinon|mauro.pezze}@usi.ch

†University of Lugano
Lugano, Switzerland

‡University of Milano-Bicocca
Milan, Italy

ABSTRACT

Faults in Web APIs may escape the testing process, and therefore affect thousands of Web applications. As a consequence, users of these applications might suffer from related failures for a long time until proper fixes are released by the Web API developers. In this paper we present RAW, a tool that tries to find workarounds automatically and at runtime, thereby reducing the negative impact of faults in Web applications. Runtime and automatically deployed workarounds serve as a temporary relief for application users while proper fixes are developed and released.

1. INTRODUCTION

Popular Web APIs, like Google Maps, YouTube and Facebook, are used in a myriad of combinations of browsers, operating systems and contexts, and evolve over time to meet new user needs. Testing all combinations of browsers and operating systems for all evolving versions and in the context of all possible applications is impossible. Faults that escape testing can persist for a long time, and may cause runtime failures that can affect many users. While waiting for permanent fixes, application and API developers often look for workarounds to temporary solve or at least mitigate the runtime problems. Finding workarounds manually and sharing them through forums is slow and inefficient. Application and API developers may not share workarounds immediately, and may not identify suitable workarounds right away. Thus, applications may fail repeatedly and for a long time before benefiting from proper workarounds.

In a recent paper, we presented a technique that exploits the implicit redundancy that can be found in many software applications, to automatically generate workarounds for software libraries [1]. In this paper, we present RAW, a tool that applies the technique to Javascript Web APIs. RAW relies on a repository of equivalent sequences of API operations, and generates candidate workarounds by choosing an alternative sequence equivalent to the failing one. Equivalent sequences are sequences of API operations that should produce the same or compatible effects from the end-users viewpoint. To generate effective workarounds automatically, we observed that many workarounds that solve known problems follow some common patters. We then framed such patterns in a set of classes that we use to identify equivalent sequences. RAW chooses candidate workarounds by prioritizing sequences equivalent to the failing one according to their estimated likelihood of solving the problem at hand.

*This work has been partially funded by the Swiss National Fund with the project WASH.

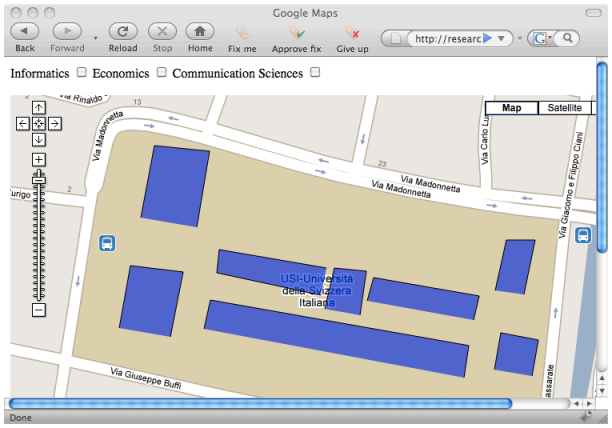
ICSE '10, May 2-8 2010, Cape Town, South Africa

2. AUTOMATIC WORKAROUNDS

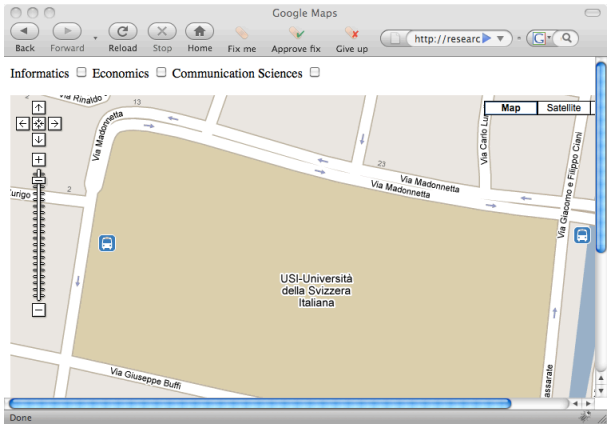
RAW implements a layer that mediates the interactions between clients and Web servers. This layer acts as a proxy, intercepting the pages requested by the clients and enabling the failure reporting system. The reporting system consists of a simple button through which users can signal a failure. This signal triggers the automatic generation of workarounds. Specifically, when a user reports a page failure, the automatic workaround system looks for alternative sequences of invocations corresponding to sequences observed within the application code, which is part of the intercepted page. The system then deploys a candidate workaround by returning a new page in which the application code is appropriately rewritten to incorporate the chosen equivalent sequences. If the new page does not incur the previous failure or otherwise meets the user's expectations, the user can continue the interactive session as if the failure never occurred. Otherwise the user may reiterate the process by signaling another failure. In this second case, the automatic workaround system looks for a different sequence. The search for an effective workaround is ultimately controlled by the user who may decide to give up at any moment.

The search for workarounds relies on a repository of equivalent sequences and a priority mechanism. Equivalent sequences belong to three categories: functionally null, invariant, and alternative operations. *Functionally null* operations are operations with no functional effects. This is the case, for example, of operations that affect only timing or scheduling. *Invariant* operations are sets of operations whose combination has a functionally null effect. *Alternative* operations are two or more sequences of operations that produce the same results.

The repository of equivalent sequences is populated by Web API and application developers, who can derive a first set of equivalent sequences from templates, and incrementally add equivalent sequences while fixing new faults. The



Faulty page: The page shows the polygons corresponding to the buildings with no selected checkboxes



Page automatically corrected by RAW: The page does not show the polygons when checkboxes are not selected

Figure 1: Issue n. 1264 in Google Maps

priority mechanism weights equivalent sequences according to their success rate as workarounds, and indicates the sequences that are more likely to solve the problem.

3. USING RAW

We implemented RAW as a Firefox extension. The extension does not affect the normal Web browsing activity, since it does not interfere with the browser until it is activated by the user in reaction to a page failure.

RAW extends the Firefox interface by adding three buttons to the toolbar: *Fix me*, *Approve fix* and *Give up*, as shown in the screenshots in Figure 1. We illustrate the functionality of these buttons through an example that refers to a known (and now fixed) problem of the Google Maps API, reported as issue n. 1264 in the Google Maps bug-report system.[1] Figure 1 presents two screenshots of a simple Web application affected by this issue. The application shows the map of the campus of the University of Lugano, and offers a set of checkboxes to display the buildings of the faculties as polygonal overlays. The Map and the polygons are created with the Google Maps API. Initially, the page should not display the polygons, which should become visible only after selecting the checkboxes on top of the map. Moreover, the polygons should scale according to the zoom level. As illustrated by the screenshot on the left-hand side of Figure 1, the initial page is not displayed correctly: all polygons are visible with no selected checkbox. Zooming into the page would also show that the polygons do not scale as expected.

With a standard browser interface, users who suffer from these problems do not have any way to react immediately. They can report the problem to the application developers and hope for a fix sometime in the future. With RAW, users experiencing the problem can ask for the help of the tool by pressing the *Fix me* button in the toolbar. The *Fix me* button activates RAW to look for a workaround. The application of a (tentative) workaround modifies the JavaScript code of the map, and reloads the page. If not satisfied by the reloaded page, the user may insist in requesting a new workaround by pressing the *Fix me* button until satisfied by the reloaded page, or they might give up. The screenshot on the right-hand side of Figure 1 shows the correct application behavior as fixed by RAW. If satisfied by the reloaded page, the user may signal the successful workaround by pressing the *Approve fix* button. If not satisfied, they can signal the failure through the *Give up* button. These two buttons do not affect the users immediately, but can help RAW generate a successful workaround faster the next time the problem is signaled.

4. ARCHITECTURE OF RAW

RAW includes a client- and a server-side subsystem. The client-side subsystem is composed of the browser extension that implements the user interface described in the previous section. The server-side subsystem implements the main functionality, and runs on a centralized server. Different clients using different Web APIs and application servers may share the same RAW server to automatically solve runtime problems. The RAW server-side is composed of three main components: *ES Repository*, *JS Rewriter*, *WA Generator*.

The ES Repository is the core of RAW and contains a set of program-rewriting rules that specify equivalent sequences. The JS Rewriter rewrites the Javascript code responsible for a page failure by applying a rewriting rule. It relies on sed[2] to implement the substitutions specified by the rule. The ES Repository contains several rules, and each rule may be applied in many ways to the same Javascript code, but only a few applications of rules may generate valid workarounds. The WA Generator is responsible for selecting the rules that can be applied to the APIs in the original Javascript code, and identify the ones more likely to generate valid workarounds. It uses a priority scheme that depends on the success rate of the rules to rank them.

5. REFERENCES

[1] A. Carzaniga, A. Gorla, and M. Pezzè. Healing web applications through automatic workarounds. *International Journal on Software Tools for Technology Transfer*, 10(6):493–502, December 2008.

[1] http://code.google.com/p/gmaps-api-issues/issues/detail?id=1264

[2] http://www.gnu.org/software/sed