6CCS3PRJ Final Year
Automatic Test Generation from a Specification

Final Project Report

Author: Hani Kazmi

Supervisor: Dr Christian Urban

Student ID: 1201492

February 4, 2015

Abstract

The Internet, and more specifically the World Wide web, is a rapidly evolving platform. With the advent of HTML5 and the current abundance of processing resources, websites have been evolving into 'web apps', which require extensive testing to ensure lack of issues. These tests currently consist of manually written 'unit tests', which target individually modules and raise an error if future work breaks the module. Of particular note are 'web APIs', sets of endpoints used to communicate between components of web apps. They are generally strictly defined, and therefore there is scope for automatic testing to ensure compliance with their specification.

This paper focuses on a subset of these APIs: those classified as 'restful', and creates a framework consisting of three components - a language to formally define the API, automatic tests to reaffirm that the specification matches a given implementation, and unit test generation from the aforementioned specification. The paper will use the API provided by the company 'Livedrive' as a case study.

Originality Avowal

Hani Kazmi

February 4, 2015

Chapter 1

Introduction

1.1 Motivations

The Internet has long been an important means of communication. Historically, this has been dominated by websites hosted on the World Wide Web: static, unchanging pages which display information and are navigated using URLs.[1] As these sites were simple, ensuring that they were reliable was a simple task. With the advent of fast broadband and increased processing power, websites have slowly been evolving into more self contained entities, colloquially known as 'web apps'.

Generally powered by HTML5[2][1] and JavaScript[3], web apps are far more dynamic. Due to the many moving pieces now involved on a website along with the increasing file sizes[4], testing has become unwieldy. This is currently overcome by writing 'Unit Tests'[2] which alert the programmer when a future change breaks existing functionality. This is tedious work, and covering all possible cases is difficult.

A common way of creating web apps is by having a client side application written in JavaScript, which communicates with a backend that can be written in a wider variety of languages. These two components may be designed and implemented by different people, and indeed different companies altogether. Therefore, there is generally a well defined API[5] used to communicate between them using HTTP[6]. If this API can be strictly defined, there is scope

[1]Uniform Resource Locater: a reference to a resource on the Internet.
[2]A revision of the HTML standard which added many elements needed for dynamic websites.
[3]A scripting language implemented by virtually all web browsers, updated as part of the HTML5 specification
[4]Website trends, 2010-2015, `http://httparchive.org/trends.php?s=All&minlabel=Nov+15+2010&maxlabel=Jan+15+2015`
[5]Application Programming Interface.
[6]Hypertext Transfer Protocol.

to automate part of the testing process using it.

1.2 Aims

The aim of this paper is to automate Web API testing as much as possible. A three-fold approach will be taken to this problem:

1. A language to formally define APIs. This will allow the API to be computationally modified and reasoned about.
2. A framework which automatically generates a battery of tests to check that an API implementations matches its specification. This will act as a basic sanity check for the implementation, as well as allow the implementation to be monitored for changes.
3. A framework which automatically generates Unit Tests for a given API specification. While all manual tests cannot be eliminated, a large subset can be inferred and constructed from the API definition. This will be accomplished by extending the definition language to allow example requests, and inferring any additional required information form the context.

1.3 Scope

Due to the time constrains of this project, the framework will be limited to dealing with APIs following a RESTful[7] architectural style. Specifically, the web API 'Onyx' used by the company Livedrive[8] for communication between their C#[9] backend and web frontend will be used as a case study and focal point for the paper.

While the project will be tested against the full Onyx API, its contents are a trade secret for Livedrive. Therefore, a small restful API modeling any needed functionality of Onyx will be created and used for the purposes of the report.

[7]Representational State Transfer: An architectural style for creating scalable web services
[8]`http://www.livedrive.com`
[9]A programming languages developed by Microsoft, regularly used for writing large scale web backends.

References

[1] Html5: A vocabulary and associated apis for html and xhtml. http://www.w3.org/TR/html5/, 2014. [Online; accessed 4-February-2015].

[2] American National Standards Institute. *IEEE Standard for Software Unit Testing.* The Institute of Electrical and Electronics Engineers, Inc, 1986.