NewLeaf Automated Testing Framework

Luke McGuire, Chris Taylor, Kasper Dugaw

College of Charleston Department of Computer Science

Introduction

In this project our team designed, implemented, and tested an automated PHP testing framework and several test cases for Moodle, an open source learning management system which we found lacked tests for external libraries included in their source code.

Framework Design

Our framework consists of three main components, the controller, test cases, and drivers.

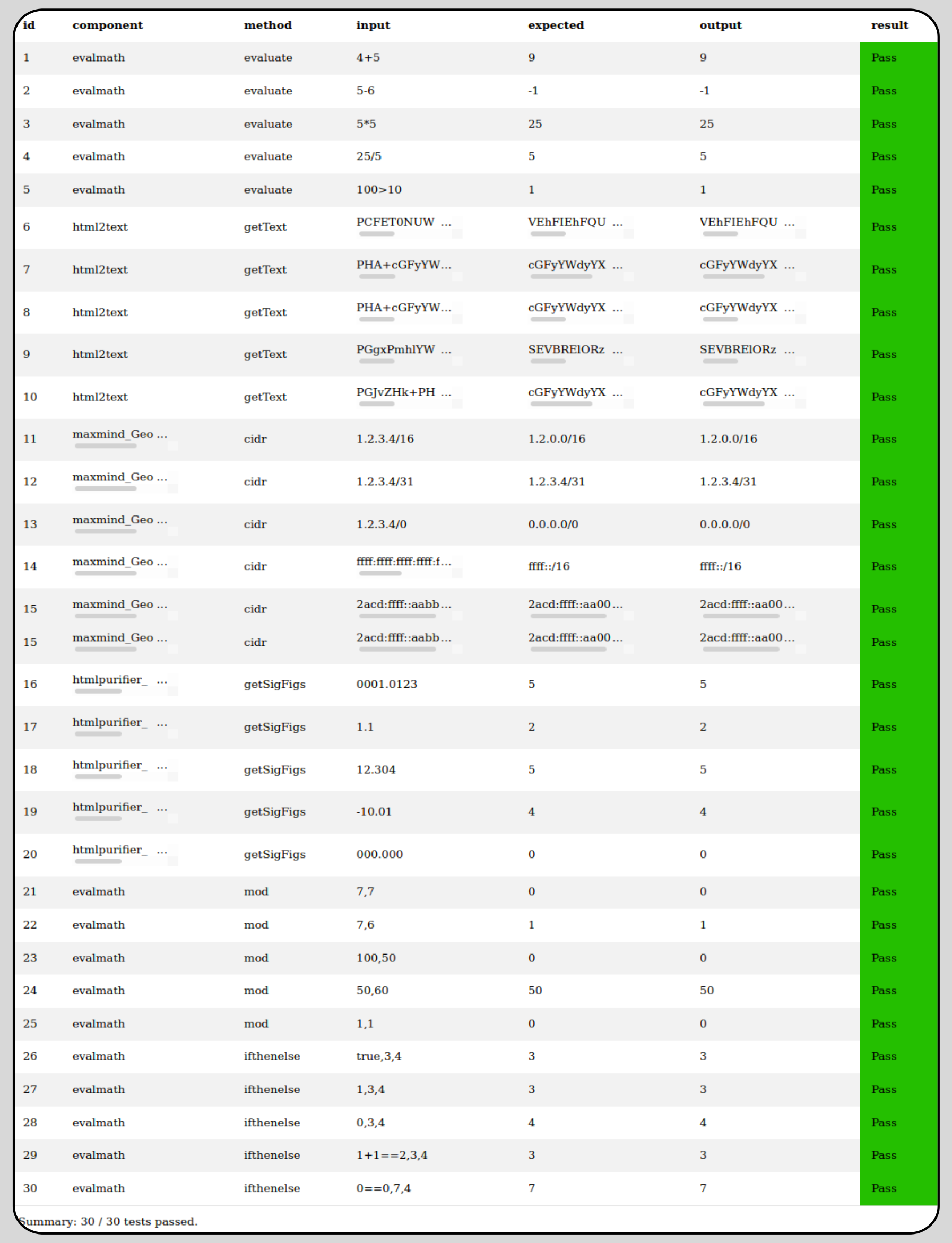
The controller is responsible for parsing and executing test cases as well as generating the results into a report.

The test cases are a set of JSON files which specify individual tests and provide the controller with the data necessary to execute them including the driver that should be used to run the test, the input that should be given to the functionality being tested, and the expected output of the provided input.

Finally, the drivers are responsible for interfacing with the Moodle code and providing the controller with the ability to test a particular method or class and returning the results of a test back to the controller.

Implementing Tests

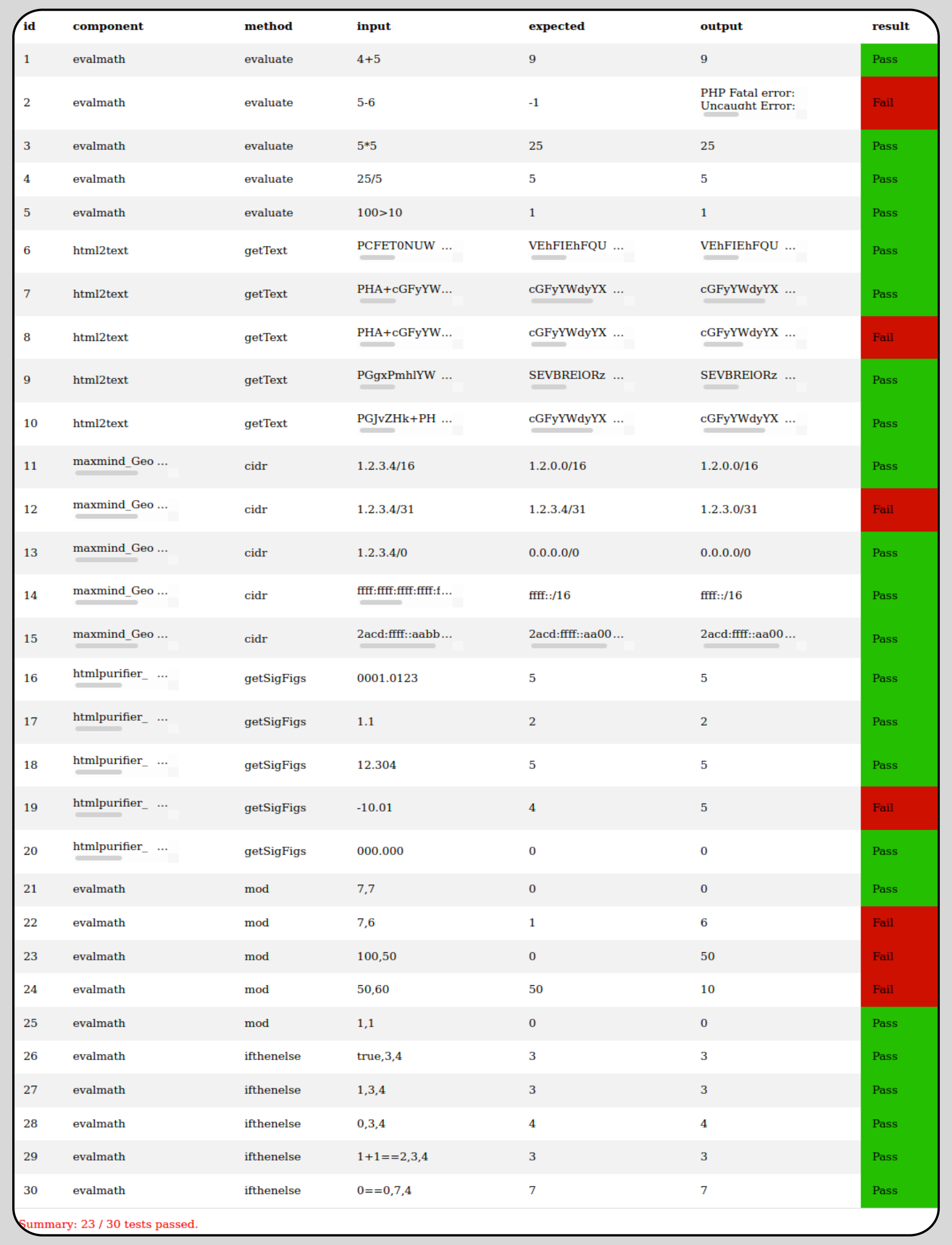
After implementing our framework we developed 30 tests for the Moodle project for 6 functions found in the project's libraries. The modular design of our framework provides a simple framework for creating new drivers and tests, and examples of these can be seen in Figures 3 and 4 respectively.



id	component	method	input	expected	output	result
1	evalmath	evaluate	4+5	9	9	Pass
2	evalmath	evaluate	5-6	-1	-1	Pass
3	evalmath	evaluate	5*5	25	25	Pass
4	evalmath	evaluate	25/5	5	5	Pass
5	evalmath	evaluate	100>10	1	1	Pass
6	html2text	getText	PCFET0NUW ...	VEhFIEhFQU ...	VEhFIEhFQU ...	Pass
7	html2text	getText	PHA+cGFyYW...	cGFyYWdyYX ...	cGFyYWdyYX ...	Pass
8	html2text	getText	PHA+cGFyYW...	cGFyYWdyYX ...	cGFyYWdyYX ...	Pass
9	html2text	getText	PGgxPmhlYW ...	SEVBRElORz ...	SEVBRElORz ...	Pass
10	html2text	getText	PGJvZHk+PH ...	cGFyYWdyYX ...	cGFyYWdyYX ...	Pass
11	maxmind_Geo ...	cidr	1.2.3.4/16	1.2.0.0/16	1.2.0.0/16	Pass
12	maxmind_Geo ...	cidr	1.2.3.4/31	1.2.3.4/31	1.2.3.4/31	Pass
13	maxmind_Geo ...	cidr	1.2.3.4/0	0.0.0.0/0	0.0.0.0/0	Pass
14	maxmind_Geo ...	cidr	ffff:ffff:ffff:ffff:f...	ffff::/16	ffff::/16	Pass
15	maxmind_Geo ...	cidr	2acd:ffff::aabb...	2acd:ffff::aa00...	2acd:ffff::aa00...	Pass
15	maxmind_Geo ...	cidr	2acd:ffff::aabb...	2acd:ffff::aa00...	2acd:ffff::aa00...	Pass
16	htmlpurifier_ ...	getSigFigs	0001.0123	5	5	Pass
17	htmlpurifier_ ...	getSigFigs	1.1	2	2	Pass
18	htmlpurifier_ ...	getSigFigs	12.304	5	5	Pass
19	htmlpurifier_ ...	getSigFigs	-10.01	4	4	Pass
20	htmlpurifier_ ...	getSigFigs	000.000	0	0	Pass
21	evalmath	mod	7,7	0	0	Pass
22	evalmath	mod	7,6	1	1	Pass
23	evalmath	mod	100,50	0	0	Pass
24	evalmath	mod	50,60	50	50	Pass
25	evalmath	mod	1,1	0	0	Pass
26	evalmath	ifthenelse	true,3,4	3	3	Pass
27	evalmath	ifthenelse	1,3,4	3	3	Pass
28	evalmath	ifthenelse	0,3,4	4	4	Pass
29	evalmath	ifthenelse	1+1==2,3,4	3	3	Pass
30	evalmath	ifthenelse	0==0,7,4	7	7	Pass

Summary: 30 / 30 tests passed.

Figure 1: Test results prior to fault injection.



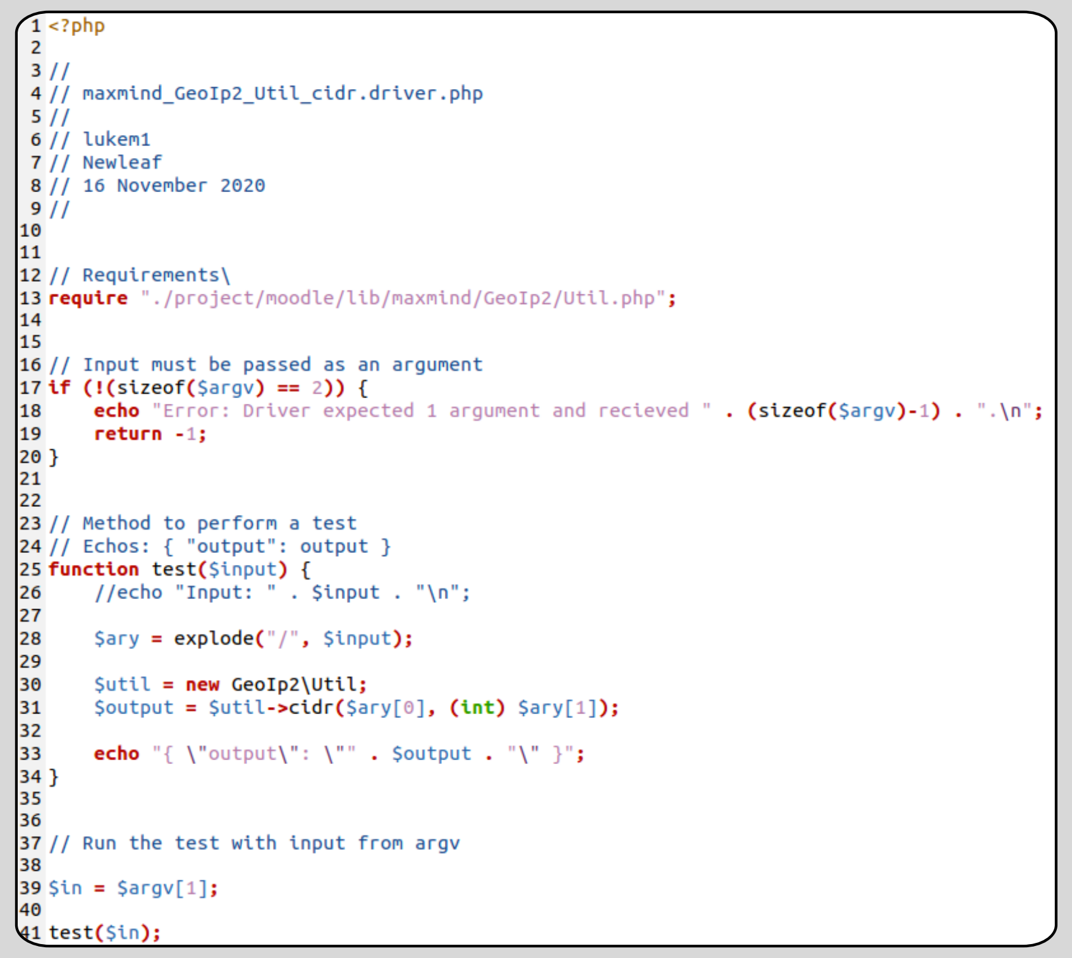
id	component	method	input	expected	output	result
1	evalmath	evaluate	4+5	9	9	Pass
2	evalmath	evaluate	5-6	-1	PHP Fatal error: Uncaught Error:	Fail
3	evalmath	evaluate	5*5	25	25	Pass
4	evalmath	evaluate	25/5	5	5	Pass
5	evalmath	evaluate	100>10	1	1	Pass
6	html2text	getText	PCFET0NUW ...	VEhFIEhFQU ...	VEhFIEhFQU ...	Pass
7	html2text	getText	PHA+cGFyYW...	cGFyYWdyYX ...	cGFyYWdyYX ...	Pass
8	html2text	getText	PHA+cGFyYW...	cGFyYWdyYX ...	cGFyYWdyYX ...	Fail
9	html2text	getText	PGgxPmhlYW ...	SEVBRElORz ...	SEVBRElORz ...	Pass
10	html2text	getText	PGJvZHk+PH ...	cGFyYWdyYX ...	cGFyYWdyYX ...	Pass
11	maxmind_Geo ...	cidr	1.2.3.4/16	1.2.0.0/16	1.2.0.0/16	Pass
12	maxmind_Geo ...	cidr	1.2.3.4/31	1.2.3.4/31	1.2.3.0/31	Fail
13	maxmind_Geo ...	cidr	1.2.3.4/0	0.0.0.0/0	0.0.0.0/0	Pass
14	maxmind_Geo ...	cidr	ffff:ffff:ffff:ffff:f...	ffff::/16	ffff::/16	Pass
15	maxmind_Geo ...	cidr	2acd:ffff::aabb...	2acd:ffff::aa00...	2acd:ffff::aa00...	Pass
16	htmlpurifier_ ...	getSigFigs	0001.0123	5	5	Pass
17	htmlpurifier_ ...	getSigFigs	1.1	2	2	Pass
18	htmlpurifier_ ...	getSigFigs	12.304	5	5	Pass
19	htmlpurifier_ ...	getSigFigs	-10.01	4	5	Fail
20	htmlpurifier_ ...	getSigFigs	000.000	0	0	Pass
21	evalmath	mod	7,7	0	0	Pass
22	evalmath	mod	7,6	1	6	Fail
23	evalmath	mod	100,50	0	50	Fail
24	evalmath	mod	50,60	50	10	Fail
25	evalmath	mod	1,1	0	0	Pass
26	evalmath	ifthenelse	true,3,4	3	3	Pass
27	evalmath	ifthenelse	1,3,4	3	3	Pass
28	evalmath	ifthenelse	0,3,4	4	4	Pass
29	evalmath	ifthenelse	1+1==2,3,4	3	3	Pass
30	evalmath	ifthenelse	0==0,7,4	7	7	Pass

Summary: 23 / 30 tests passed.

Figure 2: Test results post fault injection.

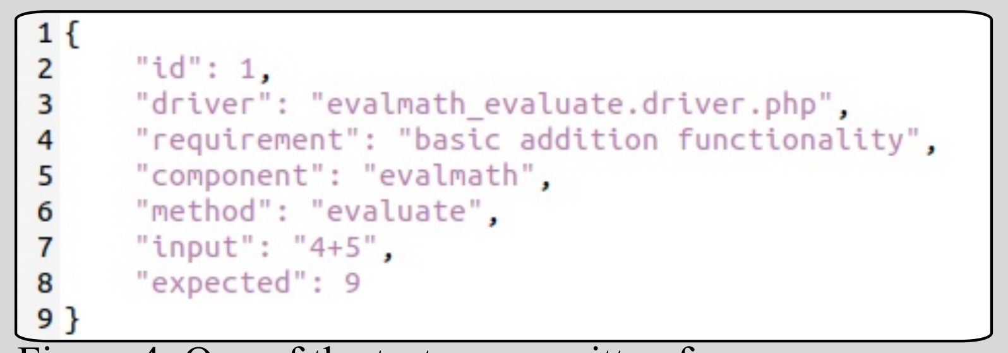
Testing and Results

To validate our testing framework we designed a system to inject faults into the Moodle source code and executed our framework before and after injecting the faults. The results of this procedure can be seen in Figures 1 and 2.



```php
<?php

//
// maxmind_GeoIp2_Util_cidr.driver.php
//
// lukem1
// Newleaf
// 16 November 2020
//


// Requirements\
require "./project/moodle/lib/maxmind/GeoIp2/Util.php";


// Input must be passed as an argument
if (!(sizeof($argv) == 2)) {
    echo "Error: Driver expected 1 argument and recieved " . (sizeof($argv)-1) . ".\n";
    return -1;
}


// Method to perform a test
// Echos: { "output": output }
function test($input) {
    //echo "Input: " . $input . "\n";

    $ary = explode("/", $input);

    $util = new GeoIp2\Util;
    $output = $util->cidr($ary[0], (int) $ary[1]);

    echo "{ \"output\": \"" . $output . "\" }";
}


// Run the test with input from argv

$in = $argv[1];

test($in);
```

Figure 3: One of the drivers written for our framework.



```json
{
    "id": 1,
    "driver": "evalmath_evaluate.driver.php",
    "requirement": "basic addition functionality",
    "component": "evalmath",
    "method": "evaluate",
    "input": "4+5",
    "expected": 9
}
```

Figure 4: One of the test cases written for our framework.

Conclusions

Over the course of this project our team learned quite a bit about the software engineering process, including project planning and management, designing and producing good tests, and how to effectively work as a team on a software project.

We were also pleased with how our testing framework itself ended up turning out. Although we had some setbacks, we were not only able to design and implement our vision, but expand upon it. After completing the basic functionality of our framework earlier than expected we were able to spend time improving or project by adding features including improved output generation, improved error handling, and scripts to automate the process of fault injection. Additionally, we were also glad we were able to produce something we felt was valuable by writing our tests for Moodle libraries that did not have pre-existing tests.

To conclude, if it's not already clear, we were very satisfied with the results of our project and our development as a team and as software engineers.

Acknowledgments

Thanks to the Moodle project and its contributors:
https://moodle.org/

Thanks to our instructor and faculty advisor:
Dr. Jim Bowring

Additional Information

Our GitHub repository, which contains the entirety of our project including source code and our final report can be found at the link below.

https://github.com/
csci-362-02-2020/New-Leaf