Project Name:	Moodle LMS
Product Name:	Moodle Testing Framework
Product Release Version:	Version 3.7.2

Document Version 4.0

Created by: Alex Laughlin, Andrew, and Chandler Long

INTRODUCTION

The Goal:

The testing framework will run on Ubuntu (Linux/Unix). The testing framework will be invoked by a single script from within the top level folder using “./scripts/runAllTests.sh” and will access a folder of test case specifications, which will contain a single test case specification file for each test case. Each of these files will conform to a test case specification template that you develop based on the example template below. Each test case specification file thus contains the meta-data that your framework needs to setup and execute the test case and to collect the results of the test case execution.

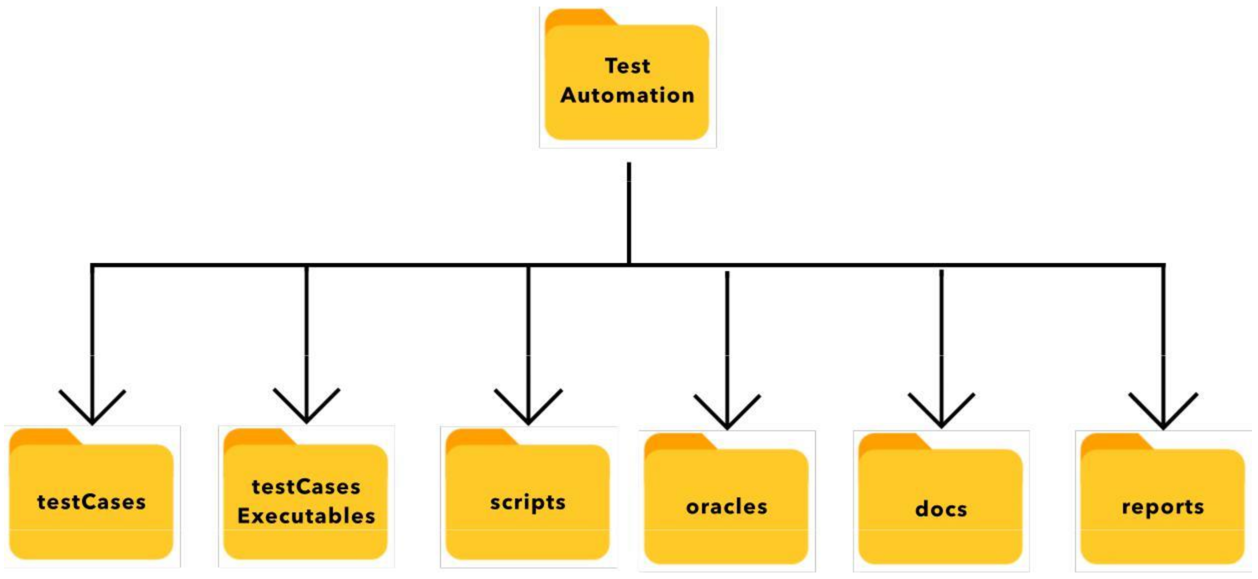
The Project:

Moodle is a free and open-source learning management system (LMS) written in PHP and distributed under the GNU General Public License. Developed on pedagogical principles, Moodle is used for blended learning, distance education, flipped classroom and other e-learning projects in schools, universities, workplaces and other sectors. Moodle was originally developed by Martin Dougiamas to help educators create online courses with a focus on interaction and collaborative construction of content, and it is in continual evolution. The first version of Moodle was released on 20 August 2002. Nowadays the Moodle Project is led and coordinated by Moodle HQ, an Australian company of 50 developers which is financially supported by a network of eighty-four Moodle Partner service companies worldwide. Moodle's development has also been assisted by the work of open-source programmers. Moodle as a learning platform can enhance existing learning environments. As an E-learning tool, Moodle has a wide range of standard and innovative features such as a calendar and a Gradebook. Moodle is a leading virtual learning environment and can be used in many types of environments such as education, training and development and in business settings.

FILE STRUCTURE

Framework Directories:

Directories inside of the testing framework are structures in such a way that allows the framework to reach into certain directory and extract information. The framework as also capable of storing files so that can be referenced later.



testCases:

The “testCases” directory houses all the tests cases that can be run in the framework

testCasesExecutables:

The “testCasesExecutables” directory houses all the methods that are extracted from the Moodle directory. Files inside of “testCasesExecutables” are initialized when there are new methods that need to be tested. These methods are then instantiated in the driver.

scripts:

The “scripts” directory houses all the scripts needed to properly run the testing framework. This is where ./runAllTests.sh is stored which runs all test cases in the framework. Driver.php is also located in this directory. Driver.php initializes all the classes that are need to properly run Moodle methods and calls upon them.

oracles:

The “oracles” directory houses all the expected results that are obtained from the testCase.txt files.

docs:

The “docs” directory houses the README.rm file.

reports:

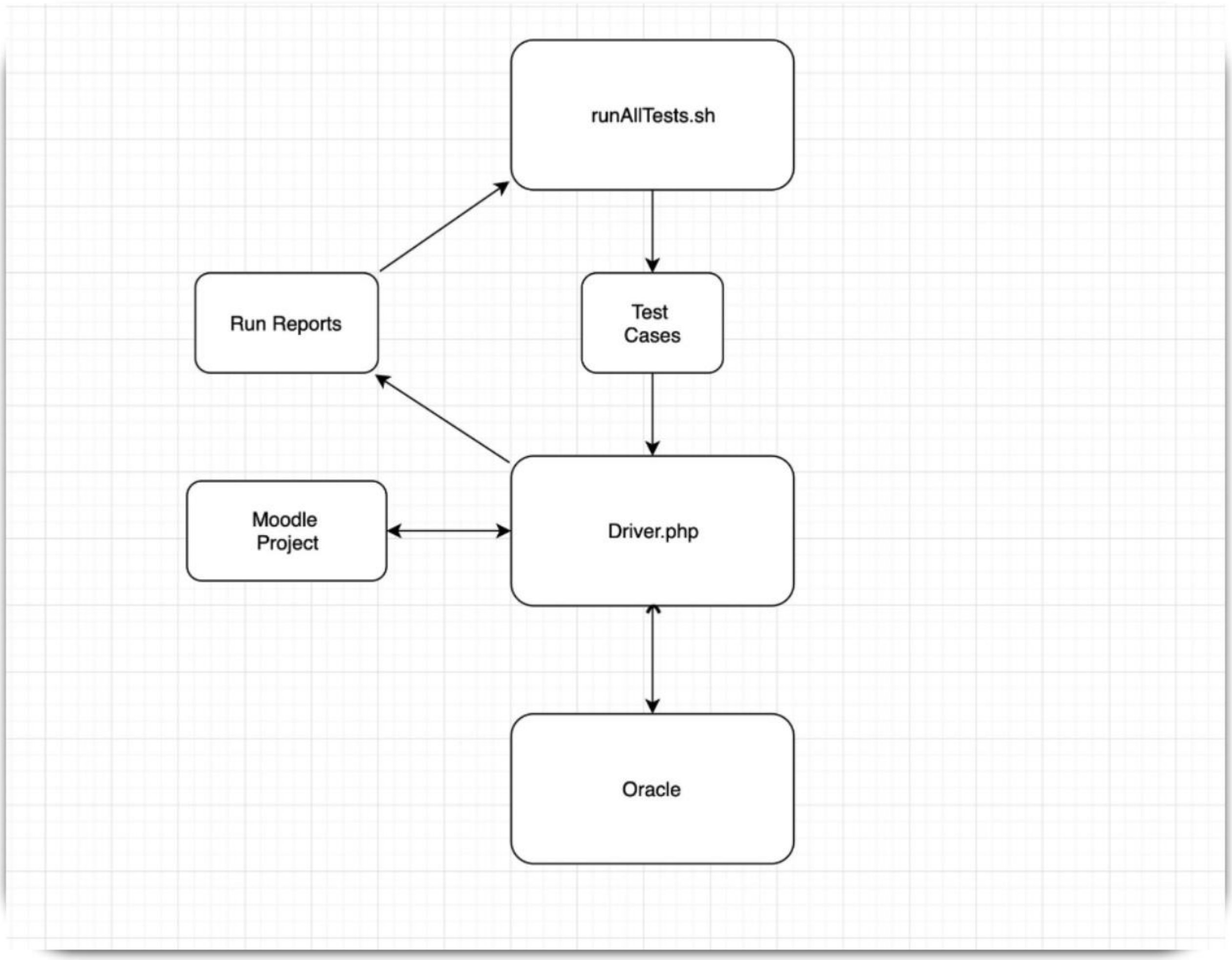
The “reports” directory houses the generated HTML file which is produced by the testing framework.

ARCHITECTURE

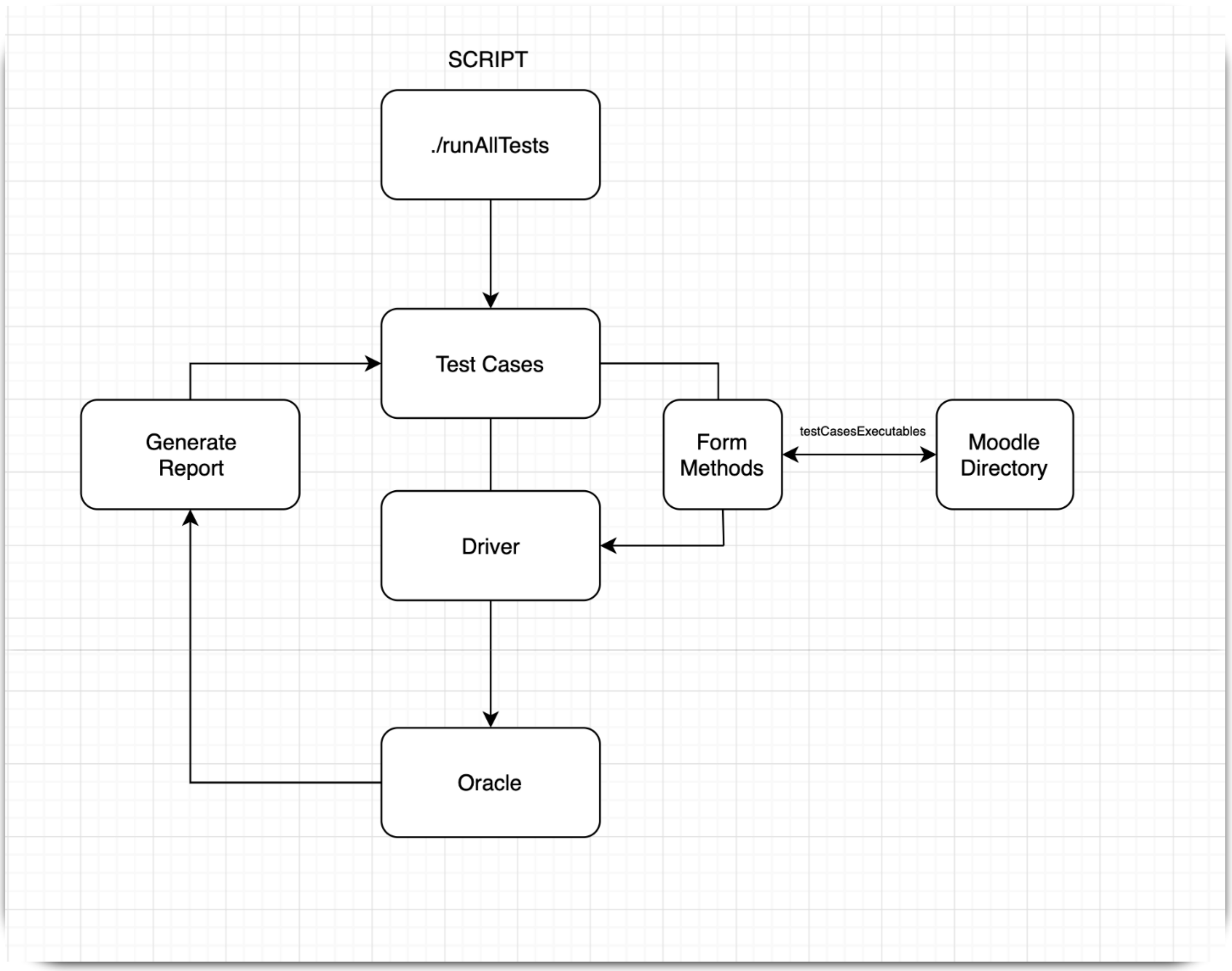
Overview:

The architecture of the automatic testing framework has changed since of that of deliverable 3. In deliverable 3 we expressed that were were in need of a way to instantiate our classes in order to proper test the majority of classes available in the massive project that is Moodle.

Previous Architecture:



Current Architecture:



FAULT INJECTION

Below is a list of faults that would cause the script to fail:

No test cases

Since our script uses the test case for specify the method to test, if the test case directory is empty the script will not run and exit the loop since there is no test cases to run and there is no method to be determined to test.

No methods

In the test case, if there is no method specified in the test case, then the script will be unable to determine what method to test. This could cause a problem if there is only one test case. If there is only one test case in the folder and no method is specified in the test case then this will cause a fault in the script since there is no method to test.

Wrong test case format

There are multiple ways that this could cause a fault. The test case is required to follow a very specific format. If this format is broken or it does not follow the format correctly then this could cause a fault. One possibility is that the oracle can be formatted incorrectly in which it does not match to the output that results from the tokenize method. This can cause the oracle to fail even though you gave it the correct values.

Wrong method format

If the method is not constructed properly then this can cause a fault in the script in which it will fail.

If the test case input starts with

Before test cases are stored, they must undergo a few checks that make sure the values that are being obtained are the ones needed to execute the method. In order to properly label these values, notes are included at the top of every test case which start with “#”. When the text file is being processes, lines that start with the character “#” are ignored. A major fault in our testing framework is exposed here, if a test case input values happens to be the character “#”, the line will be ignored resulting in a faulty test case.

CONCLUSION

Moodle is an open source software designed to assist in education. Most of the methods in the Moodle code rely on objects such as students, grades, courses, etc. This requires our test cases to have elaborate set up and break down methods. These set up methods are in place to instantiate the necessary objects to test the method, and the tear down objects reset to a good state so the tests can be run again without conflicting information. Programming these set-up methods require an intimate understanding of the software and the method being tested, as all of the necessary objects must be created with specific parameters to allow the method to run. This was initially challenging to our team because Moodle was not a familiar software at the beginning of this project. After spending more time than we would like to admit familiarizing ourselves with the Moodle code, we know understand the methods well enough to program the set-up and tear-down methods.