|  |  |
| --- | --- |
| Document Information | |
| Document name | Moodle Automation testing Assessment – Tool Recommendation |
| Project Name | Moodle M&D |
| Project Number | D71 P160614-00 |
| Project Account Code | 538637 |
| Author | Anto Inigo |
| Issue Date | 10 .05.2017 |
| Status |  |
| Version | 0.2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Change Control | | | |
| Date | **Author** | **Version** | **Description of Change** |
| 10/05/2017 | Anto Inigo | 0.1 | Creation – First draft |
| 10/05/2017 | Anto Inigo | 0.2 | Revision |
| 17/05/2017 | Anto Inigo | 0.3 | Revision |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

This document may include information that is **CONFIDENTIAL** and shall not be disclosed outside UCL and shall not be duplicated, used, or disclosed in whole or in part for any purpose other than to evaluate and implement procedures defined within this document.

CONTENTS

[1. BACKGROUND 3](#_Toc482108225)

[2. SCOPE 3](#_Toc482108226)

[2.1 TOOL 1 3](#_Toc482108227)

[2.2 TOOL 2 4](#_Toc482108228)

[2.3 MAINTANANCE AND RISKS 5](#_Toc482108229)

[2.4 RECOMMENDATION 5](#_Toc482108230)

[2.5 APPENDIX: BEHAT VS SELENIUM WEBDRIVER (JAVA) COMPARISION 6](#_Toc482108231)

# BACKGROUND

**Moodle** is an open source learning platform designed to provide educators, administrators and learners with a single robust, secure and integrated system to create personalised learning environments. At the UCL, Learning and Teaching Applications (LTA) team releases quarterly updates of Moodle core, plugins and local customisations. These require testing efforts of 3 days for three times a year and 6 days once a year. Hence, there is requirement to automate some of the manual test cases which are currently run frequently during and in-between the releases by the LTA and Digital Education team. This document provides an assessment of two tools and recommends one.

# SCOPE

Development of an automated test suite for Moodle test cases which are currently executed manually.

## TOOL 1

**Behat**: Since Moodle v3.0, Behat is considered an official Cucumber implementation in PHP and is part of one big family of behavioural driven development tools. More information about Behat can be found here (<http://behat.org/en/latest/>). Behat allows us to specify Moodle functionalities (aka features) as a human-readable list of steps and parse this steps to execute actions to simulate user interaction. It executes the actions against a headless browsers (without Javascript support, only curl-kind requests) or user simulation tools like Selenium, which interacts with browsers and allows Javascript events simulation as shown below.



More information about Behat integration with Moodle can be found here: (https://docs.moodle.org/dev/Behat\_integration).

Whenever a new module or plugin or newer version of Moodle is available, Moodle provides associated Behat acceptance test scripts with them. Behat test scripts are written in human friendly Gherkin Language and PHP integration code have been developed by Moodle developers. These test scripts are developed as features (written in Behat) by Moodle developers. These Moodle-Behat acceptance test scripts work in such a way that Moodle server creates a temporary data table instance from the Acceptance test scripts as a local test environment and performs test operations on it.

To run Behat tests, Moodle instance under test like DEV or UAT, server access and Behat setup has to be created. These require skills in Behat, PHP and familiarity with the development and server environment. Tester has to setup all these before creating or running the test scripts shipped by Moodle. This is an onetime initial set up and may take upto 5 days for an experienced person. This can be used to run regression Behat tests which were shipped with Moodle. Once the test environment is setup, there is an option to run a single, multiple, group or entire test cases or test suite. While the log files can give useful information to identify the reasons behind failed test cases, someone may have to run the failed test cases manually. More information about running Behat Acceptance tests can be found here [https://docs.moodle.org/dev/Running\_acceptance\_test.](https://docs.moodle.org/dev/Running_acceptance_test.P) Plugins and modules created or customized locally may not have Behat feature files and associated step definitions implemented in PHP, like the ones provided by Moodle itself. If that is the case, then Behat feature files should be written with associated step definitions implemented in PHP which then can be integrated into the Moodle server. The time required to develop the above is difficult to estimate since it depends on the complexity of the functionalities.

## TOOL 2

**Selenium WebDriver (JAVA):** Selenium WebDriver is a tool which automates browser actions. Here Selenium would be interacting directly with Moodle like an end user who is using Moodle through a web browser. Selenium test scripts can be written from the manual test scripts which are frequently used by the Moodle team to test recurring issues or new functionalities or problem areas.

While there is no need to directly work with Moodle server from the command line, one might need a stable staging website (like DEV or UAT environment), which is similar to the live one with all the test environment setup. Once this is setup, automated test scripts can be written from currently used manual test cases. For example, if there is a recurring problem with an assignment submission, a custom test suite can be generated to capture where the failure might occur. This situation would be similar to the end users who are interacting with the web front end. A tester would have the option to run failed test cases manually or just watch the execution of automated test cases. Test data can be taken from external files such as excel files. On a day to day basis, these test scripts can be run individually or as group from an IDE like Eclipse. It is also possible to run the whole test suite from a central repository by using tools like Jenkins, Maven after every major update to get bigger return on investment.

A proof of concept Selenium WebDriver automation of assignment submission was developed by using the manual test scripts used by the Moodle team. This has been demonstrated to some members of this project team to identify the suitability of this tool. Based on the ease of use, output format, look and feel of the test run, availability of future technical resource pool, the team felt comfortable with using the Selenium WebDriver automation.

## MAINTANANCE AND RISKS

In a nutshell, both Behat and Selenium WebDriver need certain amount of technical knowledge in their respective areas. In both cases, one would expect problems like missing link, missing page or page loading issues. This may not be bad, since these could be the reason why the application under test is failing users. In addition, one may need to update web element locators, if there is a small change in the web interface. Currently, there is no existing expertise in Behat within any ISD teams. However, the ISD Quality Services Team may offer assistance from their resource pool as there are few other ongoing automation testing projects by using Selenium WebDriver.

## RECOMMENDATION

**Behat** would be ideal for testing whole application including the locally created plugins to run on server side, as this would sit with the other Behat test scripts which were pre-packaged with Moodle. However**, Selenium WebDriver** driver would make more sense if the requirement is to check recurring and high priority issues which are not captured by existing Behat scripts and currently executed manually, from an end user perspective. As discussed above, Selenium WebDriver would further remove the requirement to execute manual test cases at some level.

It is always recommended to choose an automation testing tool for which there is a support team to take care of issues or queries. It plays a very vital role to address any difficulties or uncertainties that may arise. Hence, it is important to consider a wider test tool strategy given there is a skill shortage in Behat and in Selenium WebDriver within LTA and Moodle teams. From a testing resource pool perspective, there are few other Selenium WebDriver projects currently running with in the ISD and Quality Services may be able to offer assistance. Hence, I would recommend Selenium WebDriver as a tool to test Moodle as a web application for the chosen high priority issues while the Behat tests which are part of Moodle can be used to test the entire system. Bigger maintenance or update can be done by a Selenium expert, day to day operations and smaller fixes may be done locally as discussed in the meeting to get someone trained on Selenium WebDriver.

I would also recommend LTA staff to prioritise upskilling in Selenium and Behat.

## APPENDIX: BEHAT VS SELENIUM WEBDRIVER (JAVA) COMPARISION

|  |  |  |
| --- | --- | --- |
|  | Behat | Selenium |
| Framework | PHP framework (PHP implementation of cucumber) for web based applications. | Automates browser.  Multi browsers and multiple programming languages like Java, C#, Python, Ruby, PHP, Perl…can be used. |
| **Full suite** of Test cases for complete Moodle Testing | Mostly comes with Moodle.  No Java; Human Friendly Gherkin Language  PHP integration code SHOULD have been developed by Moodle developer.  Test scripts should be developed by Moodle developers with associated feature files and PHP implementation. | Not recommended. |
| Custom test Suite for locally created plugins | 1. As above step definitions need to be developed by a developer/tester in PHP if not already defined.  2. Options available to run one or more selected test cases, | 1. Step definitions need to be developed by a developer, in this case in Java.  2. Options available to run one or more selected test cases. |
| Test Environment | 1. Run at Command line  2. Server instance required  3. Could be difficult for non-experts to work at command line with text editors  4. Since everything is run at the server and at Command line, the observer may not see /understand what is really happening at the background. Hence, a Manual testing may be required.  5. 1000s of test cases can be run at once in addition to few selected test cases run. | 1. IDEs like Eclipse  2. Web application run  3. Test data can be taken from excel files  4. No manual test run is required. Just watch test execution in addition to the inspection of error messages.  5. Since it is a Java program continuous integration tools like Jenkins can be done with ISDs resource pool to run complete suite in addition to few test cases run on daily basis. |
| Local Expertise within Moodle and LTA | 1. N/A at the moment | 1.N/A at the moment |
| Test Team’s Expertise | N/A at the moment | Two other projects are being successfully built by Quality services within the ISD. |
| Day to Day usage and Maintenance | Training required | Training required |
| Major Maintenance | Should be done by expert | Should be done by an expert.  Might be convenient for Quality services to pool resources. |
|  |  |  |
| **Recommendation** | LTA can run most of the Behat tests packaged with Moodle. | Automate (1) the mostly used manual test cases currently run by the Moodle team and (2) the test cases which the LTA would prefer the Moodle team to run at every release. |