# IVVES

**Industrial-grade Verification and Validation of Evolving Systems**

Labelled in ITEA3, a EUREKA cluster, Call 5

ITEA3 Project Number 18022

# D5.2 – Architecture of the IVVES online experimentation and training platform

Due date of deliverable: March 31, 2021
Actual date of submission: March 31, 2021

**Start date of project: 1 October 2019** **Duration:** 39 months

**Organisation name of lead contractor for this deliverable:** OU

| | |
|---|---|
| **Author(s):** | Pekka Aho, Tanja E.J. Vos |
| **Status:** | Final (living document) |
| **Version number:** | 1.0 |
| **Submission Date:** | 31-03-2021 |
| **Doc reference:** | IVVES_Deliverable_D5.2_V1.0.docx |
| **Work Pack./ Task:** | WP5 / Task 5.1 |
| **Description:** *(max 5 lines)* | This deliverable describes the architecture for the IVVES framework, an online platform providing public IVVES artefacts to the people outside of the consortium. |

| **Nature:** | **R**=Report | | |
|---|---|---|---|
| **Dissemination Level:** | **PU** | Public | **X** |
| | **PP** | Restricted to other programme participants | |
| | **RE** | Restricted to a group specified by the consortium | |
| | **CO** | Confidential, only for members of the consortium | |

**DOCUMENT HISTORY**

| Release | Date | Reason of change | Status | Distribution |
|---|---|---|---|---|
| V0.1 | 12/02/2021 | First draft | Draft | WP5 |
| V0.2 | 23/03/2021 | Added architectural plans and examples | Draft | PMT |
| V1.0 | 31/03/2021 | Approved by PMT, to be submitted to ITEA3 | Final | ITEA3 |

# Table of Contents

# Glossary

| Abbreviation / acronym | Description |
|---|---|
| IVVES artefact | An outcome of IVVES project. Types of artefacts we will have are: tools, data, courses, tutorials, tool chains, use cases and success stories. |
| | |
| | |

# 1. Executive Summary

The IVVES framework will be an online platform that acts as a front-end for the public IVVES artefacts to the people outside of the consortium. By making sure that all the public IVVES outcomes come with a well-defined and clear package of training and possibilities to try them out (experiment with them), we hope to give dissemination of the IVVES outcomes a real boost. Confidential material will not be shared on the platform, it will be shared using other means, for example the internal repository of the project.

This document describes the architecture of the platform.

# 2. Architecture for the IVVES experimentation framework

The IVVES framework is built on open source Moodle platform on address https://learn.ivves.eu/. This web site acts as front end and user interface towards both IVVES partners that act as content providers, and end users that may be either IVVES partners or external users.

The content of this framework consists of the public outcomes of IVVES project, called artefacts that may be for example:

- tools or techniques that have been developed or extended in the project,
- data that can be used for AI or machine learning (ML),
- courses or tutorials related to the project topics,
- tool chains consisting of one or more IVVES tools,
- use cases or success stories about using the IVVES tools or techniques.

The structure and at least part of the content of each artefact is provided directly through the Moodle platform. However, some content, for example videos, source code or online versions of the tools for experimentation, is going to be linked from the Moodle to external servers, as visualized in the high level architecture below.

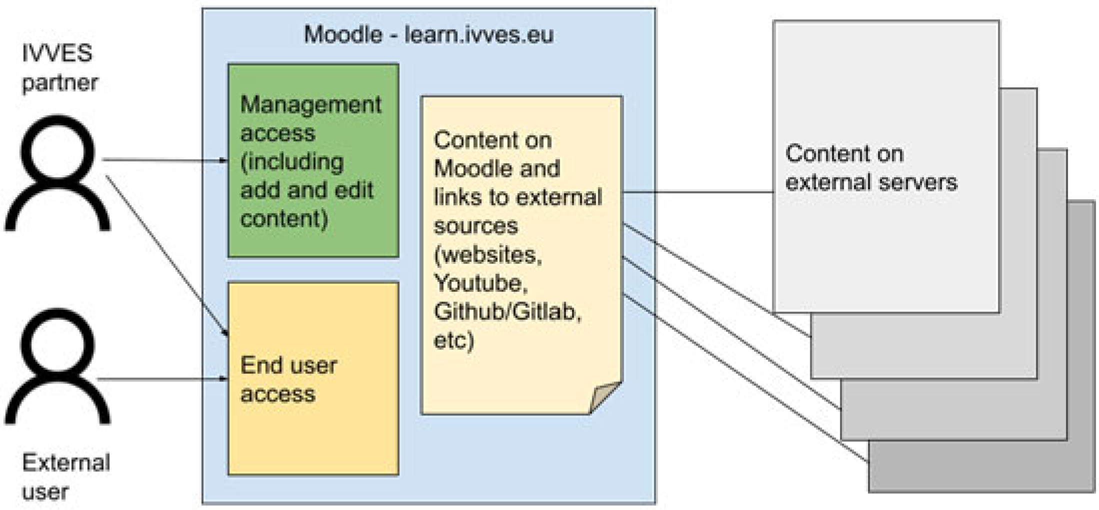



***Figure 1 High level architecture of the IVVES platform***

Access to edit the artefacts can be managed per artefact (called course in Moodle), as well as visibility to the end users. The options for end user access include allowing guest access, self-enrolment or only specifically allowing artefact editors to grant user access to the artefact. As the plan is to provide only public content on the platform, the most viable options seem to be allowing guest access or self-enrolment.

As an example artefact, we will present a course about open source TESTAR tool. This course has content including text and pictures directly on Moodle (shown in Figure 2), links to videos on Youtube (shown in Figure 3), link to source code on Github, link to external web pages for more information (shown in Figure 2), and link and instructions to a hosted environment to experiment with the tool (shown in Figure 4) if the end user does not want to install the tool on his/her own environment.

## Introduction

GUI Testing is a testing technique where one tests the System Under Test (SUT) solely through its graphical user interface. The way to find errors is to thoroughly observe the status of the GUI after each step of the test. In practice, this type of testing is often carried out manually, a tester following a predefined test case document and verifying whether the application responds to all inputs as expected.

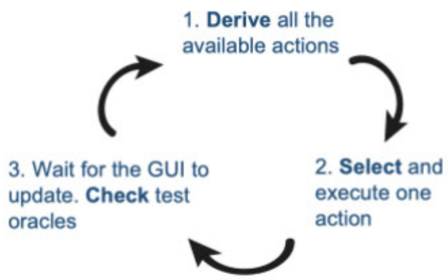
To automate GUI testing approaches there are basically two approaches:

- **script-based** test automation, where test cases or test sequences are defined prior to test execution. That means the test cases are static in the sense that they are not adapted dynamically during the test execution.
- **scriptless** test automation, where the test cases or sequences are generated dynamically, one step at a time, during the test execution, based on the observed state of the system under test.

In this course we concentrate on scriptless test automation in general and on the TESTAR tool specifically.

**TESTAR** is an open source tool for scriptless test automation through graphical user interface (GUI). TESTAR currently supports the various AI / ML based features for active learning of state models during automated exploration of the GUI of the system under test (SUT) and reinforcement learning based action selection strategies.

TESTAR at the highest level performs 3 steps in a continuous cycle.




In this course we will explain each of these steps in details and show you how you can add more intelligence to each step.

TESTAR website

Link to the TESTAR website

***Figure 2: An example of an artefact content with text and pictures***

## Experiment:

Run TESTAR and select the following protocol that has been configured to test the SUT Windows Notepad:

**01_desktop_generic_pure_random**

Select, for example, to test 5 sequences each of 20 actions, and startGenerate-mode.

If you run TESTAR for while, the results should include sequence(s) that found a crash (unexpected close of the SUT). This is a good example of the short-comings of a dumb monkey that does not know how the SUT should behave. If it does not know that Close or Exit action should actually stop the execution of the SUT, it will assume it was a crash.

After TESTAR has finished the test run and returned to the settings dialog, check the test results from TESTAR output folder. In the generated folder (the name starts with a date, then a time, and then notepad), you should see a folder named **sequences_unexpectedclose**, and from that folder you can see which of the generated sequences resulted in unexpected close of the SUT process. You can also look into the generated HTML reports of those sequences and check what was the last executed action of the sequence.

Check the video on Youtube to see how to complete this exercise:

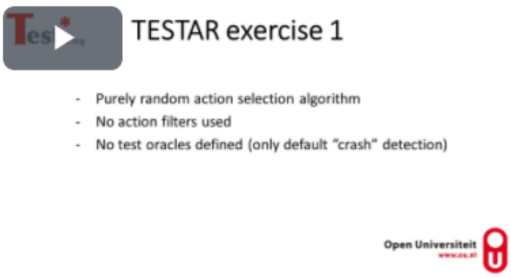



***Figure 3: An example of an artefact with video on Youtube***

## Option 1: Using a web browser to run a virtual machine that hosts TESTAR.

Go to https://qdesktop.testar.org/#/ and enter with

user: demo
password: testardemo

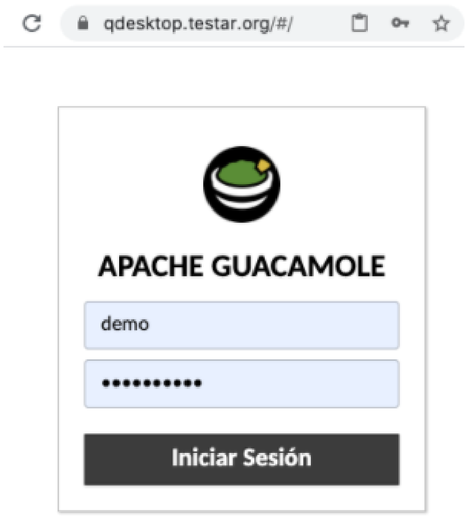



***Figure 4: An example of an artefact with a hosted version of tool for experimenting***

# 3. Conclusions

This document described the architecture of the IVVES experimentation framework based on open source Moodle software.