Continuous Piloting of an Open Source Test Automation Tool in an Industrial Environment

Pekka Aho[1], Tanja E. J. Vos[1,3], Sami Ahonen[2], Tomi Piirainen[2], Perttu Moilanen[2], and Fernando Pastor Ricos[3]

[1] Open Universiteit, The Netherlands,
[2] Ponsse Group, Kajaani, Finland,
[3] Universitat Politcnica de València, Spain.

Abstract. Piloting an academic test automation tool in an industrial environment provides valuable feedback on practical applicability of the tool, but often requires a significant collaboration effort and an iterative process of feedback, development, and evaluation. In this paper, we propose an approach for continuous delivery of TESTAR open source test automation tool into an industrial continuous integration (CI) environment for piloting and evaluation.

Keywords: GUI testing · continuous delivery · continuous integration

1 INTRODUCTION

The market pressure towards faster delivery and higher quality software increases the popularity of continuous integration (CI) processes in industry [1]. A key facilitator of CI is automated testing that should be executed on several levels of system abstraction.

Piloting an academic test automation tool in an industrial environment provides valuable feedback on practical applicability of the tool, but often requires a significant collaboration effort and an iterative process of feedback, development, and evaluation. In this paper, we propose an approach for continuous delivery of TESTAR [2] tool into an industrial CI environment for piloting and evaluation. Although our continuous piloting approach is not restricted to GUI testing tools, we focus on automated testing through GUI.

2 CONTINUOUS PILOTING OF TESTAR

2.1 TESTAR

TESTAR [2], testar.org, is an open source tool for automated testing through graphical user interface (GUI). TESTAR is traversal-based, generating tests during execution (also referred as online testing), meaning that test cases do not have to be defined prior to test execution. Instead, each test step is generated based on the actions that are available in that specific time and state of the GUI.

The default action selection mechanisms is based on random monkey testing, but there are more advanced strategies available, for example based on genetic programming [3] and machine learning [4].

Depending on the complexity of the system under test (SUT), and sometimes on the technologies used in the development of the SUT, TESTAR might require SUT-specific instructions how to reach all parts of the GUI and how to restrain TESTAR to keep testing the interesting parts of the GUI. In practice, the process of developing the instructions is iterative, trying out the instructions by executing TESTAR on the SUT and improving the instructions until the results meet the expectations.

The challenging and complex SUT used in the pilot required further development of TESTAR core functionalities and SUT-specific instructions for GUI exploration, implemented as a TESTAR protocol class in Java. Therefore, it was important to take the tool updates into account in the piloting environment.

2.2 Ponsse Group and System Under Test

Ponsse Plc is one of the worlds leading manufacturers of forest machines for the cut-to-length method, cutting the tree trunks in the forest to lengths that suit their intended use. At the same time, the information systems inform the end users of the types and quantities of timber they will next receive from the forest.

The SUT of this TESTAR pilot is Ponsse Opti4G control system of the forestry machine. Opti4G has been in the market for a long time, and updates are still being delivered. Opti4G runs on a specific rugged PC hardware with touch screen and Windows Embedded operating system. The long lifetime of the products results in a lot of legacy code, and the specific hardware with touch screen involves non-standard GUI specific code, introducing challenges for TESTAR and GUI test automation.

2.3 Piloting Environment

Figure 1 illustrates the high level architecture of the piloting environment. The blue components are located in Ponsse premises, connected to Ponsse internal network. The yellow components are outside the internal network of Ponsse.

TESTAR is developed on public GitHub repository. The branch used for Ponsse specific development is not public for security reasons. Most of the modifications have been SUT-specific instructions for TESTAR, but some generic features, such as handling of multiple SUT processes and using image recognition to find specific widgets, have been merged into master branch of TESTAR.

Test execution environment is running Windows Embedded 32-bit operating system on a rugged PC hardware with a touch screen. There is a CI agent running and connected to CI server.

Ponsse is using internal installation of Visual Studio Team Services for CI. There are two triggers for CI server to start executing TESTAR build definition: 1) CI polls for TESTAR changes (commits) on GitHub, and 2) Scheduled trigger every office day at 6 am for testing the nightly SUT build.

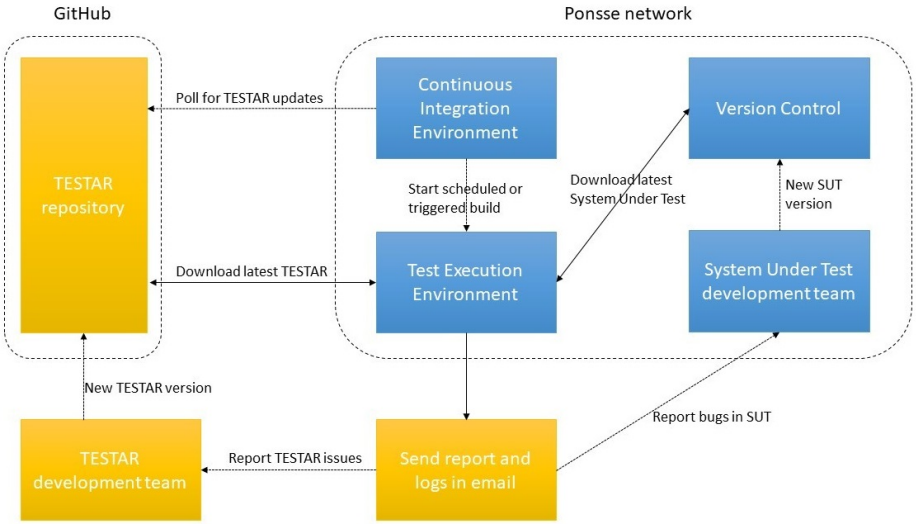

Fig. 1. The architecture of the continuous TESTAR piloting environment.

The build definition for TESTAR CI pipe had the following steps that are executed by the CI agent:

1. Download and install the latest SUT version,
2. Download and build the latest TESTAR tool source code from Github (specific Ponsse branch),
3. Scripts for setting up a test user account for SUT,
4. Execute test scenarios with various TESTAR settings and given number and length of test sequences,
5. Archive and send the output folder of TESTAR as email attachment to tool developers (test reports and logs).

The TESTAR tool developers analyze the test reports and logs to improve the tool and SUT-specific GUI exploration instructions. When suspicious SUT behavior is observed, it is reported to the SUT development team.

3 RELATED WORK

Alegroth et al [1] present an empirical industrial study on evaluating applicability of visual GUI testing (VGT) and EyeAutomate tool in an industrial CI environment. The main challenges observed during the study were high maintenance costs of the VGT test scripts and high costs for analysis of false positive test results.

Lowell and Stell-Smith [5] present an approach using open source Marathon GUI testing tool to record user actions into test scripts and discuss the maintenance of the recorded test scripts in continuous integration process.

Xie and Memon [6] present an approach for continuous integration testing of web-based community-driven GUI-based open source software. The approach uses open source GUITAR tool for GUI testing with varying test sets to limit the test execution time. A quick and dirty test set for crash testing is executed on each GUI code commit, functional reference testing set for smoke testing on each days GUI build, and comprehensive GUI testing set on a major GUI release.

However, none of these approaches consider updating the testing tool during the process.

4 CONCLUSIONS AND DISCUSSION

The proposed continuous piloting approach enables rather independent development of the tool that is being evaluated in the pilot. This decreases the collaboration effort required from the industrial partner. Triggering a new test run at Ponsse CI when a new commit of the TESTAR tool has been pushed into Ponsse-specific branch gives quick feedback for the TESTAR developers.

The next steps of this industry-academia collaboration include using the continuous piloting for introducing new model extraction functionality of TESTAR into Ponsse environment. In the future, the goal is to automatically detect GUI changes by comparing extracted models of consequent SUT versions.

5 ACKNOWLEDGMENT

This work has been funded through the ITEA3 TESTOMAT Project and EU H2020 DECODER project (www.testomatproject.eu, www.decoder-project.eu).

References

[1] E. Alegroth, A. Karlsson, and A. Radway, Continuous Integration and Visual GUI Testing: Benefits and Drawbacks in Industrial Practice, Proc. 2018 IEEE 11th ICST, 9-13 Apr 2018, Vaesteroas, Sweden.

[2] T.E.J Vos, P. Kruse, N. Condori-Fernandez, S. Bauersfeld, and J. Wegener, TESTAR: Tool Support for Test Automation at the User Interface Level, Int. Journal of Information System Modeling and Design, IJISMD 2015, vol.6(3), 2015, pp. 46-83.

[3] A.Esparcia-Alczar, F. Almenar, T. Vos, and U. Rueda, Using genetic programming to evolve action selection rules in traversal-based automated software testing: results obtained with the TESTAR tool, Memetic Computing 10(3): 257-265 (2018).

[4] A. Esparcia-Alcazar, F. Almenar, M. Martinez, U. Rueda, and T. Vos, Q-learning strategies for action selection in the TESTAR automated testing tool, Proc. 6th META2016, Marrakech, Morocco, 2016.

[5] C. Lowell and J. Stell-Smith, Successful Automation of GUI Driven Acceptance Testing, Int. Conf. on Extreme Programming and Agile Processes in Software Engineering (XP 2003), LNCS, vol. 2675 pp 331-333, Springer, 2003.

[6] Q. Xie and A. Memon, Model-Based Testing of Community-Driven Open-Source GUI Applications, 22nd IEEE Int. Conf. on Software Maintenance (ICSM 2006), 24-27 Sep 2006, Philadelphia, PA, USA.