Building a small-scale multiplatform automated software testing facility

Maxim Mozgovoy
Active Knowledge Engineering Lab
The University of Aizu
Aizuwakamatsu, Fukushima, Japan
mozgovoy@u-aizu.ac.jp

Testing is an integral part of software development pipeline. Various types strategies of testing have been discussed in literature at least from late 1960s [1]. However, the practice of continuous automated testing as a part of daily routine, and recognition of testing code as a part of project's codebase gained momentum much later, and usually associated with the "rediscovery" of test-driven development by Kent Beck [2].

Daily testing, like other agile development practices, is much easier to adopt with the support of specialized tools. For example, one may argue that while it is not overly difficult to setup a simple automated build server, the emergence of out-of-the-box systems like Jenkins or TeamCity has greatly contributed to popularity of continuous integration practices. Gradually, typical testing scenarios also received external support — initially with testing frameworks like xUnit, and later at the level of major development environments (such as Visual Studio and IntelliJ IDEA) and automated build systems.

However, most readily available functionality is designed to support mostly low-level *unit testing*, while setting up testing of larger software components (*integration testing*) still requires much effort from the developers. This is not surprising, since integration tests are more project-specific, and it is harder to provide a truly universal testing framework.

Still, the last two decades were marked with the appearance of tools aimed to assist integration testing. For example, the testing of a website functionality can be greatly simplified with the use of Selenium WebDriver [3]. Thus, the task of setting up automated integration tests is perhaps less daunting nowadays than ever.

The author of the present talk has first-hand experience of setting up and maintaining an integration testing scaffolding for a mobile game project. The general overview of the system we designed and certain specific challenges we had to overcome are discussed in our works [4, 5]. Here I want to focus mostly on personal experience associated with setting up a testing facility, its maintenance, and on practical use cases important for our team.

Such topics are rarely discussed in literature. They are often considered as "technicalities" not related directly to central ideas of proper testing organization. Furthermore, discussion of purely technical issues is like shooting at moving targets: many of them quickly become irrelevant as technologies grow mature. However, we can observe recurrent patterns in these issues, indicating potentially problematic areas.

Automated testing framework has become an essential component of our software development pipeline, so for us the benefits associated with automated testing greatly outweigh the cost of efforts necessary to fine tune the system and keep it running. However, it is useful to know potential issues that might appear in a project similar to ours.

References

[1] A. I. Llewelyn and R. F. Wickens, "The testing of computer software," in *Software Engineering, Report on a conference sponsored by the NATO SCIENCE COMMITTEE, Garmisch, Germany*, 1968, pp. 7–11.
[2] K. Beck, *Test Driven Development*. New Jersey: Pearson Education (US), 2002.
[3] P. Ramya, V. Sindhura, and P. V. Sagar, "Testing using selenium web driver," in *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, Feb. 2017 - Feb. 2017, pp. 1–7.
[4] M. Mozgovoy and E. Pyshkin, "Mobile farm for software testing," in *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, 2018, pp. 31–38.
[5] E. Pyshkin and M. Mozgovoy, "So You Want to Build a Farm: An Approach to Resource and Time-Consuming Testing of Mobile Applications," *ICSEA 2018*, 2018.