MATEMATICĂ ŞI INFORMATICĂ

CREATION OF FRAMEWORKS FOR TEST AUTOMATION OF WEB APPLICATIONS

Maxim GORBATIN

CZU: 004.052 maxg1999@mail.ru

This article provides examples of how QA automation skills can be implemented in the applicative way, specifically for making frameworks for different kind of applications. Also, it describes why it is really helpful for business and why QA automation frameworks are important.

In the following article the need to implement automation testing and testing in general will be analysed

Firstly, testing helps to detect defects even on the early stages of development, therefore, it will be cheaper to fix them than if they were found on the production stage.

Secondly, testing assures that a product which is regularly tested is more qualitative that the one which is not tested.

Thirdly, automation testing helps to reduce the workload from the manual colleagues. It also helps to implement a routine work with the hard calculations much faster and more accurately than a person can do. Moreover, automation tests can be executed during the nightly runs in CI/CD.

Fourthly, automation frameworks are created in order to simplify automation processes.

Nowadays, a lot of huge companies use testing in their products. In some spheres it's crucial, like the banking system, medical system, airplane system and so on. The price of a mistake in any of these fields can be enormous. It means that application should be tested carefully; that's why testing, as a rule, is being performed on the following levels:

- **Unit testing** – testing of a function/method, small module which is often performed by a developer;
- **Integration testing** – testing which checks that modules work correctly with the integrated module and the integration didn't impact the modules which had worked before the integration;
- **System testing** – testing which includes functional (what a system should do) and non-functional testing, which includes performance testing, security testing, usability testing, etc.;

- **Acceptance testing** – the testing, the successful completion of which leads to the acceptance by the customer and paying of the team which was responsible for the creation of software.

There're a lot of techniques and approaches how to perform development and automation testing, however, the most popular ones wil be reviewed (Table 1).

Test Driven Development	Behavior Driven Development
The process starts by writing a test case.	The process starts by writing a scenario as per the expected behavior.
TDD focuses on how the functionality is implemented.	BDD focuses on the behavior of an application for the end user.
Test cases are written in a programming language.	Scenarios are more readable when compared to TDD as they are written in simple English format.[1]
Changes in how the application functions impact a lot on the test cases in TDD.[3]	BDD scenarios are not much impacted by the functionality changes.[1]
Collaboration is required only between the developers.	Collaboration is required between all the stakeholders.[2]
Might be a better approach for projects which involve API and third-party tools. [3]	Might be a better approach for projects which are driven by user actions. E.g.: e-commerce website, application system, etc.[2]
Some of the tools which support TDD are: JUnit, TestNG, NUnit, etc.	Some of the tools which support BDD are SpecFlow, Cucumber, MSpec, etc.
Tests in TDD can only be understood by people with programming knowledge.[4]	Tests in BDD can be understood by any person including the ones without any programming knowledge.[2]
TDD reduces the likelihood of having bugs in your tests.	Bugs in tests are difficult to track when compared to TDD.

Table 1 Comparison of TDD and BDD approaches.

As it can be seen above a lot of sinutarities and differences can be identified in the techniques approached, however, they're made for a qualitative process of creation of software to complete.

Therefore, in order to build a good automation process, the following recommendations should be followed:

- It should be defined whether using of automation is justified in the context of product;
- The automation instrument should be defined;
- The standards of test scenarios should be respected;

- The fundamental metrics should be defined and tracked:
  - Percentage of the found defects;
  - Pass rate;
  - The time which is needed for automation in each cycle;
  - Changes in performance.

Bibliography:

1. North, Dan (March 2006). “Introducing BDD”. Dan North. Retrieved 25 April 2019.
2. John Ferguson Smart (2014). BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle. Manning Publications. ISBN 9781617291654.
3. Beck, Kent (2002-11-08). Test-Driven Development by Example. Vaseem: Addison Wesley. ISBN 978-0-321-14653-3.
4. Koskela, L. „Test Driven: TDD and Acceptance TDD for Java Developers”, Manning Publications, 2007

Scientific director
***Valeriu UNGUREANU**, PhD*