Automated acceptance testing tools for web applications using Test-Driven Development

Abstract. In the last years the software engineering community pays a strong interest in agile development methods. Those methods place software testing for example the Test-Driven Development method as an important task of the development process. Agile projects rely on good test automation tools. In this paper we evaluate five test automation tools for their usage in acceptance testing for web applications using Test-Driven Development.


Keywords: test-driven development, test automation tools, acceptance testing, web application testing.

Introduction

Nowadays software engineering pays more attention to so-called agile methods. Those methods place automated tests as one of the important tasks in software engineering. One of the famous agile methods is test-driven development (TDD), where code is implemented according to prior created test cases. The industrial practice shows, that test automation is not a trivial task. Especially in TDD the popular test automation tools using the Capture&Replay method, where the interaction between the user and system under test (SUT) is recorded, cannot be applied because the system is not implemented at the time of test case design.

Different application types need different test automation methods. For example the tests of web applications are hard to automate because of the complex architecture and several technologies used within it. Speaking of acceptance tests, a web application has to be tested in a customer-like environment. This means, that the application has to be deployed on an application server.

In order to conduct acceptance tests in a test-driven development project, the test automation tool issue has to be clarified. The goal of this article is to evaluate five test automation tools for their usage in acceptance tests conducted with TDD. We use a GQM-like method, where tools are evaluated according to predefined goals. For each tool, we first create a test case and then implement code for a small user story example based on the Spring framework.

This article begins with the description of the evaluation process. Then we introduce our running example, selected tools and we evaluate each tool according to the evaluation process. At the end we summarize the evaluation and select the most appropriate test automation tool.

Evaluation process

As mentioned in the introduction we define a clear evaluation process for the tool comparison. For this purpose we first adjust the Goal Question Metric (GQM) introduced in [1]. We define the main goal and several subgoals. For each subgoal we define questions which have to be answered. Each question has to be answered within a predefined value range. Different as in GQM we do not use concrete metrics but predefined answer values of the questions for each subgoal.

The overall evaluation process is divided into three parts: 1) tool usage 2) question answering and 3) tool comparison.

As shown in Fig. 1 we first choose a tool and create test cases with the test automation tool. Then we implement the code according to the test case definition. After step 1) we answer the questions defined for all subgoals as described earlier. To give a better understanding of our evaluation process we first describe the TDD-like activities of step 1) and then briefly introduce our goals, questions and value ranges for step 2).

As shown in Fig. 1 we first choose a tool and create test cases with the test automation tool. Then we implement the code according to the test case definition. After step 1) we answer the questions defined for all subgoals as described earlier. To give a better understanding of our evaluation process we first describe the TDD-like activities of step 1) and then briefly introduce our goals, questions and value ranges for step 2).

Test-Driven Development as described in [2] and the movement of Extreme Programming [3-4] gains high attention in the software engineering community. Different as in typical iterative development models, the test activities start before and not after the code is implemented. As test cases are used as the specification of the system, the first activity in the development process is test case design. Next, code needed to pass the created test case is being implemented. If needed a refactoring of the created code can be conducted in the last step.

According to [14] TDD can be used on two test levels. The first one is unit testing, where test cases are mostly created with tools from the xUnit family. The second and important for this article test level is acceptance testing. Based on