Chapter 1.15
Automated Software Testing

Paula Donegan
Instituto Atlântico, Brazil

Liane Bandeira
Instituto Atlântico, Brazil

Cristina Matos
Instituto Atlântico, Brazil

Paula Luciana da Cunha
Instituto Atlântico, Brazil

Camila Maia
Instituto Atlântico, Brazil

ABSTRACT

This chapter approaches paramount aspects related to test automation, introducing the importance of implementation in the software market and essential bases, such as adjustment to the organizational reality and establishment of an efficient strategy. Types of tools and directives for a successful implantation are presented. Test automation has been considered the main measure taken to enhance test efficiency — fundamental in the software-development process. Responsible for verifying and/or validating the quality of the executable product compared to performed documentation and client requirements. Therefore, with the chapter content here provided, we aim to provide the reader with an understanding of test automation and grant relevant orientations to assist implementing it.

INTRODUCTION

Given the growing complexity of applications and new technologies, such as the advent of the client/server environment (in particular Web applications), the effort necessary for application testing has increased.

To assure that software conforms to requirements, various test stages may be identified: unit, integration, system, and acceptance. Bugs' impact increases with the evolution of the test stage in

which they are found, in other words, the cost of detecting errors during unit test is less than integration and system tests.

Each use case has test objects that may need to be retested several times during the project, demanding resources. These retests normally are required when a new functionality is added or when a bug is corrected, because there is no guarantee that the changes made will impact negatively on other parts already constructed. Therefore, the assistance of a tool capable of repeating a test already executed in the past is quite interesting.

Besides, multiple execution paths and diversity of possible inputs and outputs of an application complicate the execution of manual tests, which may be simplified by automation. In addition, performance, load and volume tests are examples of tests that are difficult to be accomplished without the help of automated testing tools. There are also some types of tests that are almost impossible to be executed manually, for example, a test to verify a system's performance with thousands or millions of simultaneous accesses or having to use an enormous amount of data.

Automating software tests speeds development and reduces retesting effort spent in each stage, thus reducing time and cost. However, this reduction is normally noticed only after a while, because there are high investments in the implantation stage, such as organizational needs, training, and tools acquisition. Automation allows increase of amplitude and depth of developed tests.

Testing automation might or might not be helpful. It allows one to take advantage of idle machine time (i.e., the period in which the developer is not working) to execute tests. Therefore, test execution can be more effective and waste less resources.

BACKGROUND

Automated software testing is an activity that seems to have obvious benefits: tests may be executed swiftly, are more consistent, and may be repeated various times without increasing cost. However, it is not a trivial activity and requires effective planning and elaborate test-case definition, as well as other characteristics, which will be explained in more detail later in this chapter. Benefits and risks, possible tools, an implantation methodology and directives for script generation are also described.

An automated test between different phases of the development process has the purpose of verifying if what was constructed from that stage backwards is correct and is adequate as an input for the next stage. An example would be a programmer testing a software component before doing the integration of components.

The generated test process is automated and capable of ensuring that the system logically operates according to the code by executing the functions through a series of test cases.

With a tool, you can expect the test script to conduct the verification processes and return results that verify whether the product under test meets code logic.

A test engineer usually follows a procedure to decide whether a problem found is a defect. However, an automated test tool makes decisions based on methods invocation, during which it detects errors and defects. Thus, the tool makes an effort to remind the developers of the importance of adopting a good error-handling technique.

But, can a tool verify that all test tasks have been performed? The answer is based on the requirements of your organization and on the architecture of the software project (Li & Wu, 2004).