Propose Automated Software Testing Tools to Test Given Application and Report Bugs

T. Rajani Devi
Andhra Pradesh, India

Abstract

Testing is the process of executing a program with the intension of finding errors.

we want to propose automated testing tools to test given application and report bugs.

Thorough testing is crucial to the success of a software project. Testing or find defects or bugs is time consuming, expensive often repetitive, and subject to human error. Automated testing helps teams test faster. Allow them to test substantially more code, improves test accuracy, and frees up QA engineers so they can focus on tests require manual attention and their unique human skills.

1.INTRODUCTION

Thorough testing is crucial to the success of a software project. If your software doesn't work properly, changes are strong that most people won but or use it…atleast not for long. but testing or find defects- or bugs- is time consuming, expensive, often repetitive, and subject to human error. Automated testing, in which quality assurance teams use software tools to run detailed, Repetitive, and data-intensive tests automatically, helps teams improve software quality and make the most of their always-limited testing resources. Automated testing helps teams test faster. Allows them to test substantially more code, improves test accuracy, and frees up QA engineers so they can focus on tests require manual attention and their unique human skills.

Use these best practices to ensure that your testing is successful and you get their successful and you get the maximum return on investment (ROI).
1.Decide what test cases to automate
2.Test early and test often
3.Select the right automated testing tool
4.divide your automated testing efforts
5. create good, quality test data
6. Create automated tests that are resistant to change in the UI

1.1. Decide What Test Cases to Automate

It is impossible to automate all testing, the first step to successful automation is to determine what test cases should be automated first.

The benefit of automated testing is correlated with how many times test can be repeated.Tests that are only performed a few times are better left for manual testing.

Good test cases for automation are those that are run frequently and require large amounts of data to perform the same action.

You can get the most benefit out of your automated testing efforts by automating:
.Repetitive tests that run for multiple builts
.Tests that are highly subject to human error
.tests that require multiple data sets
.Frequently used functionality that introduces high risk conditions
.Tests that are impossible to perform manually
.Tests that run on several different hardware or software platforms and configurations

IJERT

6 steps of getting started with automated testing

Success in test automation requires careful planning and design work. start out by creating an automation plan. This plan allows you to identify the set of tests to automate, and serve as a guide for future tests. first, you should define your goal for automated testing and determine which types of tests to automate. there are few different types of testing, and each has its place in the testing Process.for instance, unit testing is used to test a small part of the intended application. load testing is performed when you need to know how a web service responds under a heavy workload to test a certain piece of the application's UI, you would use functional or GUI testing.

After determining your goal and which types of tests to automate you should decide what actions your automated tests will perform. don't just create tests steps that test various aspects of the application's

Behavior at one time. Large, complex automated tests are difficult to edit and debug. It is best to divide your tests into several logical, smaller tests. This structure makes your test environment more coherent and manageable and allows you to test code, test and processes. you will get more opportunities to update your automated tests just by adding small tests that address new functionality. Test the functionality of your application as you add it, rather than waiting until the whole feature is implemented.

When creating tests, try to keep them small and focused on one objective. For example, use separate tests for read-only versus read/write tests. this separation allows you to use these individual tests repeatedly without including them in every automated test. once you create several simple automated tests. you can group your tests into one larger automated test. you can organize automated tests by the application's functional area major/minor division in the application, common functions or a base set of test data. if an automated test refers to others tests you may need to create a test tree, where you can tests in a specific order.

1.2. Test Early and Test Often

To get the most out of your automated testing, testing should be started as early as possible in the development in the development cycle and run as often as needed. The earlier testers get involved

In the life cycle of the project the better, and the more you test, the more bugs you find. you can implement automated unit testing on day one and then you can gradually built your automated test suite. bugs detected early are a lot cheapter to fix than those discovered later in production or deployment.

IJERT

1.3. Select the right automated testing tool

Selecting an automated testing tool is essential for test automation. there are a lot of automated testing tools on he market, and it is important to choose the tool that best suits your overall requirements. Consider these key points when selecting an automated testing tool:

.Support for your platform and technology. are you testing. Netc# or WPF

Application and on what operating systems?

.Flexibility for testers of all skills levels. can your QU department write automated test scripts or is there a need for keyword testing?

.Feature-rich but also easy to create automated tests. does the automated testing tools support record-and-playback test creation as well as manual creation of automated tests does it include features for implementing checkpoints to verify values databases, or key functionality of your application?

.create automated tests that are reusable, maintainable and resistant to changes in the application UI. Will your automated tests break if your UI changes?

For detailed information about selecting automated testing tools for automated testing see selecting automated testing tools.

1.4. Divide your automated testing efforts

Usually, the creation of different tests is based on the skill level of the QA engineers. it is important to identify the level of experience and skills for each of your team members and divide your automated testing efforts accordingly. For instance, writing automated test scripts requires expert knowledge of scripting languages. thus, in order to perform that task, you should have QA engineers that know the script language provided by the automated testing tool.

Some team members may not be versed in writing automated test scripts. These QA engineers may be better at writing test cases. it is better when an automated testing tool has way to create automated tests that does not require an in-depth knowledge of scripting languages, like test complete's "keywords tests" capability. a keyword test(also known as keyword-driven testing) is a simple series of keywords with a specified action. With keyword tests you can simulate keystrokes. Click buttons, select menu items, call object methods and properties, and do a lot more. Keyword tests are often seen as an alternative to automated test scripts. Unlike scripts, they can be easily used by technical users and allow users of all levels to create robust and powerful automated tests.

You should also collaborate on your automated testing project with other QA engineers in your department. testing performed by a team is effective for finding defects and the right automated tool should allow you to share your projects with several testers.

1.5. Create good, quality test data

Good test data is extremely useful for data-driven testing. The data that should be enters into input fields during an automated test is usually stored in an external file. This data might be read from a database or any other data source like text or XML files, Excel sheets, and database tables. a good automated testing tool actually understands the contents of the data files and iterates over the contents in the automated test. Using external data makes your automated tests reusable and easier to maintain. to add different testing scenarios, the data files can be easily extended with new data without needing to edit the actual automated test. creating test data for your automated tests is boring but you should invest time and effort into creating data that is well structured. with good test data available writing automated tests become a lot easier. The earlier you create good-quality data, the easier it is to extend existing automated tests along with the application's development.

1.6. Create automated tests that are resistant to changes in the UI

Automated tests created with scripts or keywords are dependent on the application under test. The user interface of the application may change between builts, especially in the early stages. These changes may effect the test results, or your automated tests may longer work with future versions of the application. The problem is that automated testing tools use a series of properties to identify and locate an object. Some times a testing tool relies on location coordinates to find the object. For instance, if the control caption or its location has changed, the automated test will no longer be able to find the object when it runs and will fail. To run the automated test successfully, you may need to replace old names with ones in the entire project, before running the test against the new version of the application. However, if you provide unique names for your controls, it makes your automated tests resistant to these UI changes and ensures that tour automated tests work without having to make changes to the test itself. This best practice also prevents the automated testing tools from relying on location coordinates to find he control which is less stable and breaks easily.

IJERT

automated testing with test complete

the best practices described in this article will help you successfully implement test automation. our own automated testing tool-automatedQA test complete- includes a number of features that make it easy for you to follow these best practices:

with test complete you can perform a different type of software testing

.functional testing
.unit testing
.load testing
.keyword-driven testing
.data-driven testing
.regression testing
.distributed testing
.coverage testing
.Object-driven testing
.manual testing

Test complete allow you to divide each test into individuals test parts, called test items and organize them in a tree-like structure. it lets you repeatedly use individual test and run them in a certain order.

.test complete supports keyword-driven testing. these automated tests can be easily created by in experiences test complete users, and they are a good option when a simple test needs to be created quickly.

.test complete supports five scripting languages that can used for creating automated test scripts: VBScript, Jscript, Delphi script c++script and c# script.

.when test complete, QA engineers can share a test project with their team.

.test complete offers a name mapping feature that allows you to create unique names for processes, windows, controls and other objects. It makes your names and test cleares and easier to understand, as well as independent of all objects properties and less prone to errors to understand if the UI changes. This feature allows you to test your application successfully even in the early stages of the application life cycle when the GUI changes often.

.Test complete provides many other features that help you get started quickly with you automated testing.

Test complete addresses a full range of software testing challenges facing corporate IT documents

Product developers, QA engineers, and consultants. The software enhances the software testing process by increasing efficiency, removing complexity and covering costs

Viable automated testing methodologies

Now that we've eliminated record/playback as a reasonable long-term automated testing strategy, lets discuss some methodologies that I(as well as others) have found to be effective for automating functional

or system testing for most business applications

The "Functional Decomposition" Method

The main concept behind the "functional decomposition" script development methodology is to reduce all test cases to their most fundamental tasks, and write User-Defined functions. Business function scripts, and "sub-routine" or "utility" scripts which perform these tasks independently of one another. In general, these fundamental area include:

1. navigation(eg. "access payment screen from main menu")

2.specific(business) function(e.g. "post a payment")

3.data verification(e.g. "verify payment updates current balance")

4.return navigation(e.g. "return to main menu")

In order to accomplish this, it is necessary to separate data from function. this allows an automated test script to be written for a business function, using data-files to provide the both input and the expected-results verification. A hierarchical architecture is employed, using a structured or modular design.

2.Conclusion

Test completes have many features that will help you build a solid foundation for your automated testing process that promotes high software quality. you will be able to run test faster, test more code improve the accuracy of your tests, and focus your testing teams attention on more important tasks that take advantage of their human capabilities.

IJERT

3.References

[1]Automated software testing: introduction, management, and performance by E.Dustin, et al(1999)

[2]automated testing handbook, by L.hayes(1995)

[3]software testing automation:effective use of test execution tools by D.Graham et al(1999)

[4]just enough software test automation, by D.mosley, et al(2002)

[5]integrated test design and automation:using the test frame method by H.buwalda,et al(2001)

Kolawa, Adam; Huizinga, Dorota (2007). *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press. p. 74. ISBN 0-470-04212-5.

.Brian Marick. "When Should a Test Be Automated?". StickyMinds.com. Retrieved 2009-08-20. Elfriede Dustin, et al. (1999). *Automated Software Testing*. Addison Wesley. ISBN ISBN 0-201-43287-0.

Elfriede Dustin, et al.. *Implementing Automated Software Testing*. Addison Wesley. ISBN ISBN 978-0-321-58051-1.

Mark Fewster & Dorothy Graham (1999). *Software Test Automation*. ACM Press/Addison-Wesley. ISBN 978-0-201-33140-0.

Roman Savenkov: *How to Become a Software Tester*. Roman Savenkov Consulting, 2008, ISBN 978-0-615-23372-7

Hong Zhu et al. (2008). *AST '08: Proceedings of the 3rd International Workshop on Automation of Software Test*. ACM Press. ISBN 978-1-60558-030-2.

Mosley, Daniel J.; Posey, Bruce. *Just Enough Software Test Automation*. ISBN 0130084689..