Robot Framework: A boon for Automation

Mandara Nagendra[1], C N Chinnaswamy[2], Dr. T H Sreenivas[3]

M.Tech student[1], Associate Professor[2], Professor[3]
Department of Information Science and Engineering[1,2]
Department of Computer Science and Engineering[3]
The National Institute of Engineering, Mysore, Karnataka, India[1,2]
VVCE, Mysore, Karnataka, India[3]

Abstract: **System-level test automation has gone through multiple generations, where each new generation has raised the level of abstraction used in the test design. As software is updated or changed, or reused on a modified target, emergence of new faults and/or re-emergence of old faults is quite common. Regression testing is an integral part of the extreme programming software development method. This paper talks about test automation, regression testing, types of regression testing, usage of Robot framework and its advantages.**

Keywords: System-level test automation, regression testing, Robot framework

I. INTRODUCTION

To integrate a component within a larger system, three major properties, the fitness, the correctness, and the robustness, must be tested. The fitness of a component for an application is in general treated as the compatibility of the provided interface of the component and the specification of the required interface of the application. The correctness of a component is its ability to return the correct output when provided with the correct input, while the robustness concerns the absence of a behaviour possibly jeopardizing the rest of the system, especially under wrong input. When lot of components is present, integration testing became quite complex and one of the software development improvement steps pertains to testing process improvements which can hardly be done without test automation.

System-level test automation has gone through multiple generations, where each new generation has raised the level of abstraction used in the test design. The state of the art test automation, the keyword-driven testing process, abstracts the implementation of tests behind high-level actions, i.e. keywords. In GUI testing, keywords typically depict basic user actions, such as pressing keys, typing or reading text. The tests are built as sequences of keywords, and keywords are automatically translated into concrete low-level scripts

There are various tools for test automation available – commercial and open source, but few are suitable for black box testing. The two main benefits of keyword abstraction are the reduced amount of maintenance work related to tests and the fact that the tests are easier to build and understand. When a detailed implementation of an action is in one script, the maintenance work needs to be performed only to that script. Understanding and creating tests do not require programming expertise since keywords are easy to understand, because they are not low-level system commands, but rather general commands familiar from everyday usage. A keyword script also is much shorter than a more traditional test script.

II. BACKGROUND OF REGRESSION TESTING

As software is updated or changed, or reused on a modified target, emergence of new faults and/or re-emergence of old faults is quite common. Sometimes re-emergence occurs because a fix gets lost through poor revision control practices (or simple human error in revision control). Often, a fix for a problem will be "fragile" in that it fixes the problem in the narrow case where it was first observed but not in more general cases which may arise over the lifetime of the software. Frequently, a fix for a problem in one area inadvertently causes a software bug in another area. Finally, it may happen that, when some feature is redesigned, some of the same mistakes that were made in the original implementation of the feature are made in the redesign.

Regression testing is an integral part of the extreme programming software development method. In this method, design documents are replaced by extensive, repeatable, and automated testing of the entire software package throughout each stage of the software development process. Regression testing is done after functional testing has concluded, to verify that the other functionalities are working.

III. TYPES OF REGRESSION TESTING

The various regression testing techniques are:

- **Retest all**

This technique checks all the test cases on the current program to check its integrity. Though it is expensive as it needs to re-run all the cases, it ensures that there are no errors because of the modified code.

- **Regression test selection**

Unlike Retest all, this technique runs a part of the test suite (owing to the cost of retest all) if the cost of selecting the part of the test suite is less than the Retest all technique.

- **Test case prioritization**

Prioritize the test cases to increase a test suite's rate of fault detection. Test case prioritization techniques schedule test cases so that the test cases that are higher in priority are executed before the test cases that have a lower priority.

- **Types of test case prioritization**
  - General prioritization – Prioritize test cases that will be beneficial on subsequent versions
  - Version-specific prioritization – Prioritize test cases with respect to a version of the software.
- **Hybrid**

This technique is a hybrid of regression test selection and test case prioritization.

Figure 1 shows the types of regression testing.

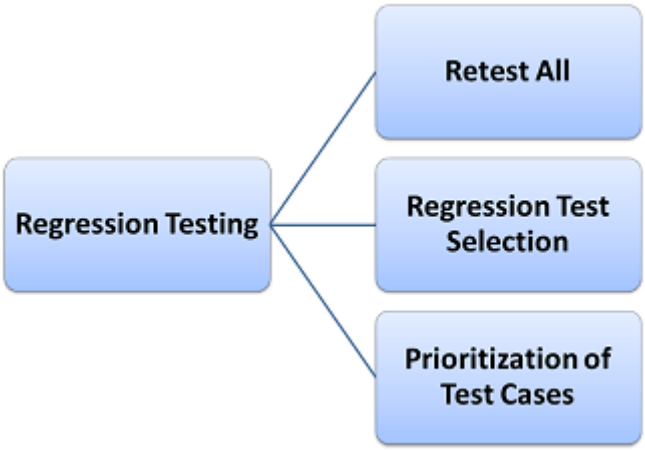

Fig. 1 Regression testing techniques

IV. ROBOT FRAMEWORK

The Robot framework is an open source test automation framework, solely developed by Nokia Siemens communication technology limited company. Robot framework is used for acceptance testing and acceptance test-driven development (ATDD). It has a tabular test data syntax and it uses the keyword-driven testing approach. It is a set of automated testing tools as shown in Figure 2.

Test Data- The test data comprises of test and data files and folders as well as the contents of those which are used for test execution.

Robot framework - It must be run on Python or Jython, the measured system must setup by Python or Jython, because Robot Framework source code is written by Python language.

Test Libraries - It is composed of two parts, with BuiltIn Library Robot framework, as well as the R&D personnel according to the test requirements.

System under test – the product to be tested.

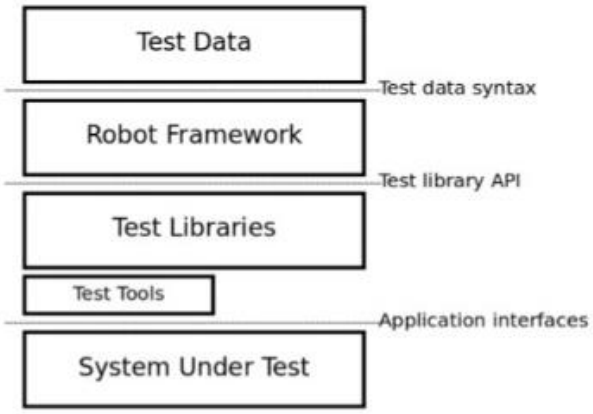

Fig. 2 Robot framework architecture

Robot Framework performs testing process by calling keywords, which are written by Python or Java. Robot itself brings many already realized keywords functions, which simply leads storage files during the test file.

V. ADVANTAGES OF ROBOT FRAMEWORK

Robot Framework is a keyword-driven test automation framework, written in Python. It empowers testers to automate and manage complex workflow scripts efficiently.

Keyword-driven testing or some call it table-driven testing are the notions widely applied to an application-independent automation. The tester needs to develop data tables with keywords, independent of the test automation framework or any other tool used to run them. Then it is required to code the test script that will, in its turn “drive" the tested application and the data.

All the features, mentioned below ensure that Robot Framework can be used to automate test cases in a quick and proficient fashion

- High-Level Architecture
- Simple Tabular Syntax
- Data-driven Test Cases
- Separate Test Data Editor
- Clear Reports
- Detailed logs
- Generic test libraries
- Webtesting, Swing, SWT, Windows GUIs, databases, SSH, Telnet,...
- Remote test libraries and other plugins for Jenkins/Hudson, Maven, Ant,..
- Text editor support: Emacs, Vim, TextMate

A. Testing Libraries

Once you’ve installed Robot Framework you get the core framework and a bunch of useful **Standard Test Libraries**. But you are not limited by those in any case. Apart from Standard Test Libraries there are many external ones that are mostly contributed by the generous Robot Framework community and serve different purposes.

B. Remote Libraries

To make Robot Framework “supreme” Framework we can use **Remote Libraries** to write our own Test Libraries in any programming language that supports the XML-RPC protocol and run them on different machines if that’s what we need. Remote Libraries are employed the same way as “normal” Test Libraries, except you’ll need to import them in Test Cases and/or Resource Files. Another huge advantage here is that **Robot IDE** also fully supports this functionality and can utilize the documentation from Remote Libraries.

C. Test Cases

Writing Test Cases - Moving on to the test cases, Robot has all its test cases in tables. You can save them either in HTML or table-separated value (TSV) files.

Managing Test Cases - There are several ways in which you can organize test cases in Robot Framework. Test cases are made into test suites, which are sets of test cases, taken either from single or multiple files.

Running Test Cases - Once we've finished writing the test cases we need, those can be executed using provided Python scripts. Since our lovely Robot Framework is system- and platform- independent - we can run our tests any platform with Python: be it Windows, Linux, Unix or Mac.

The execution of test cases is done in the following steps:

- Collecting test cases, reading and setting variables.
- Running all the steps in every test case
- Providing the execution statistics (which test cases have passed/failed)
- Writing the detailed log in xml format
- Generating the report and log in html format.

The tester can set numerous options from the command line: select the test cases by name or tag, set variables, even configure the order in which those tests will be performed. The last option is really useful. You can run tests randomly, thus increasing the probability to find bugs.

VI. CONCLUSION

This paper analyses Robot Framework which is based on automated testing framework. It is also a keyword driven automated test table frame and it will make the process and management more standardized. Robot Framework is a very easy-to-use, yet still powerful enough to be an acceptance-level test automation framework. It is quite flexible, and you can extend its functionality through Python and Java modules. You don't need much programming experience to write test cases. Writing them in tables makes your tests well-structured and ultra-readable. The logs and reports are also a good information resource. Hands down Robot Framework is our go-to web automation tool.

REFERENCES

[1] Nokia Solutions and Networks. Robot Framework User Guide. [online].available: http://robotframework.org/robotframework/latest/RobotFrameworkUser-Guide.html.

[2] Laukkanen P, "Data-Driven and Keyword-Driven Test Automation Frameworks", Master's Thesis, Helsinki University of Technology Aalto University, 2006.

[3] Model-Based Testing with a General-Purpose Keyword-Driven Test Automation Framework by Tuomas Pajunen, Tommi Takala, and Mika Katara, Department of Software Systems, Tampere University of Technology, Finland

[4] Usage of Robot Framework in Automation of Functional Test Regression by Stanislav Stresnjak and Zeljko Hocenski in ICSEA 2011: The Sixth International Conference on Software Engineering Advances

[5] Application Analysis of Automated Testing Framework Based on Robot by Liu Jian-Ping, Liu Juan-Juan, Wang Dong-Long, School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China

[6] Software Test Automation with Robot Framework by Harsha T and B A Sujatha Kumari in International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 5 Issue: 6