ISSN: 2348 - 2117

International Journal of Engineering Technology and Computer Research (IJETCR)
Available Online at www.ijetcr.org
Volume 3; Issue 3; May-June-2015; Page No. 141-144

Object Oriented Testing with Automation in Matlab using MLUnit Tool

Mamta, Rajiv Munjal

Computer Science and Engineering, MDU University, CBS Group of Institution (CBSGI), Jhajjar, India

mamta.yadav.2817@gmail.com

Computer Science and Engineering, MDU University, CBS Group of Institution (CBSGI), Jhajjar, India

rajiv.munjal@rediffmail.com.com

Abstract:
Validation and verification of the code is needed due to ever-increasing complexity of embedded software applications, and the emergence of safety critical applications. Many embedded software development groups are using models and doing upfront engineering before testing on the final product to address this need,. Use of old style of testing late in the development cycle resulted in very expensive release cycles. Object-orientation enforces many important programming principles, such as modularity, encapsulation, and information hiding; it is not enough to guarantee the quality of software products. Numerous analysis as well as design methodologies state the item a great well-designed object-oriented system would single need minimal testing. In this implementation of Class based testing in Matlab using ml unit tool.

Keywords: Object oriented testing, Automation testing, class based test, matlab, mlunit tool

1. Introduction

Object-oriented technology has become more and more popular in several various contexts. The Object-oriented paradigm is applied in the areas of programming languages, user interfaces, databases, design and specification methodologies. OOPS based languages are widely applied in industry, and several commercial applications are developed and designed and with object oriented technology.

As a consequence, the attitude towards object-oriented software quality has undergone a rapid change during the last years. Numerous analysis as well as design methodologies [75, 11, 23] state the item a great well-designed object-oriented system would single need minimal testing.

The object oriented paradigm has been considered powerful enough to assure software quality without any additional effort. Unfortunately, although object-orientation enforces many important programming principles, such as modularity, encapsulation, and information hiding, it is not enough to guarantee the quality of software products.

Object oriented software contains errors just like traditional code it is known to both practitioners and researchers. Due to their peculiarities object oriented systems present new and different problems with respect to traditional programs.

2. RESEARCH ADDRESSING QUALITY ASSESSMENT

Research addressing quality assessment lead to the definition of specific object-oriented metrics. These metrics provide quality indicators for identifying parts of the system which are more likely to be error-prone. Quality of object-oriented software has been addressed from two different viewpoints, namely, quality assessment and quality achievement in the last years.

When the level of quality of a class, a cluster of classes, or a system is inadequate, we need a way of improving it, Quality assessment methods are complementary to quality achieving techniques. As far as quality achievement is concerned, it is possible to identify two main approaches:

A. Methodology based: These methodologies pay little attention to verification of the developed system, according to the underlying hypothesis that a suitable application of the methodology should lead to well designed systems, which are easy to maintain. This methodology involves using techniques and methodologies that aim at improving the software development process and specifically address the analysis, design, and development of object-oriented systems.

B. Verification based: using static or dynamic analysis techniques that targets revealing faults. The underlying idea is that, despite the effectiveness of the process, human beings are error-prone and program will always

Corresponding author: Mamta

contain faults. Examples of static analysis techniques are formal proofs of correctness and code inspections and testing techniques are examples of dynamic techniques.

3. FOCUS AND CONTRIBUTION OF OOPS

While sharing several commonalities throughout regular procedural 'languages', kinds object-oriented paradigm highlights novel elements that be requested being proper resolved.

Inheritance, encapsulation and data hiding raise visibility problems imply incremental testing concerns, and polymorphism and dynamic binding introduce undesirability related issues. The structure of object-oriented software is different from that of traditional codes. Moreover, the structure involving object-oriented software is actually quite other through That associated with traditional programs.

Throughout object-oriented unique codes, procedures (methods) usually are small and also effectively grasped. Your difficulty will move coming from inside program code modules towards the interfaces between them. Testing at the unit level tends to be less complex in the object-oriented case than for traditional procedural systems.

4. AUTOMATED TESTING: PROCESS, PLANNING, SELECTION OF TOOLS [2]

Manual testing is performed by a human in front of a computer carefully executing the test steps. Using an automation tool to execute your test case suite is Automation Testing.

The actual automation computer software can also enter test out files in to the System under Analyze, examine anticipated in addition to actual effects in addition to crank out detailed test out studies. Test Automation demands considerable investments of money and resources. Effective improvement series requires setup connected with exact same test suit regularly. Using a test automation tool it's possible to record this test suite and re-play it as required. No human intervention is required once the test suite is automated. This improved ROI of Test Automation.

Purpose of Automation is to reduce number of test cases to be run manually and not eliminate manual testing all together.

Benefits of Automated Testing

Automated testing is essential due to following reasons:

- Manual Testing is time as well as cost consuming
- It's difficult to test for multi lingual sites manually
- Automation does not need Human intervention. You can run automated test unattended (overnight)
- Automation boosts speed of test execution
- Automation helps boosting Test Coverage
- Manual Testing can become boring and error prone.

Test Cases to Automate

Test cases to be automated can be selected to increase the automation ROI

- **High Risk** - Business test cases
- Test cases that are executed again and again
- Test Cases that are very difficult to perform manually
- Test Cases are time consuming

The following category of test cases are not good for automation:

- Test Cases that are newly designed and not executed manually already at least once
- Test Cases regarding that the Needs are generally changing quickly..
- Test cases in which executed with the ad-hoc basis.

Automation Process [2]

Steps are followed in an Automation process

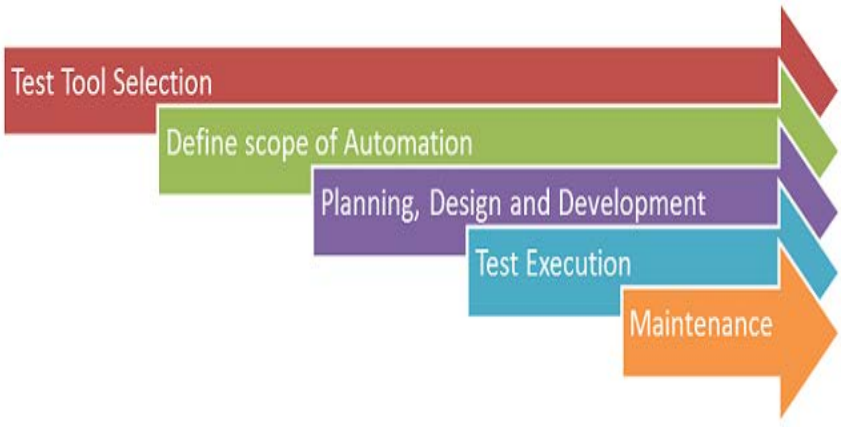

Fig 1: Test tool selection

Analyze tool selection mainly is determined by the particular engineering the appliance under Analyze is built about. For instance QTP does not support Informatics

Define the scope of Automation

Scope involving automation may be the place of a form under Test that will be automated.

- Feature which can be ticks due to the business
- scenarios of which have large quantity regarding data
- Common benefits across applications

Planning, Design and Development

Through the particular phase an individual build Automation strategy & plan, of which includes immediately after details-

- Automation tools selected
- Framework design AND it's features
- In-Scope along with Out-of-scope item involving automation
- Schedule and Timeline regarding scripting in addition to execution
- Deliverables involving automation testing.

Test Execution

Automation Scripts usually are executed during your phase. The scripts need input test info sooner There are set for you to run. Soon after executed they required comprehensive test reports

Maintenance

In the same way new functionalities are generally excess towards the program Under Test inside successive cycles, Automation Scripts need to help always be added, reviewed as well as held pertaining to each release cycle.

5. IMPLEMENTATION OF CLASS IN MATLAB [5]

Object-oriented programming (OO) applies to help software development your current standard science along with engineering practice connected with identifying patterns along with defining a classification method describing the person patterns.

Classification systems along with design patterns enable engineers as well as scientists to make sense involving complex systems along with for you to reuse efforts by others.

By employing classification systems and also design patterns to help programming, your OO approach improves ones ability to be able to manage software complexity—particularly ticks As soon as developing along with maintaining large applications as well as facts structures.

Class [3]

classdef Syntax

Class definitions are blocks of code that are denoted by the classdef keyword at the beginning and the end keyword at the end. Files can contain only one class definition.

The following diagram shows the syntax of a classdef block. Only comments and blank lines can precede the classdef key word.

Sample code to define class

```
classdef clas1
  properties
    x
  end
  methods
    function p=sq(obj)
      p= obj.x*obj.x
    end
   end
  end
```

when we run above code then result is as follow

Create object of class

```
>> y=clas1
y =
        class1
```

Assign value of property

```
>> y.x=9
y =
        class1
```

accessing member function of class and passing object as parameter

```
>> sq(y)
p =
  81
ans =
  81
```

Testing using assert keyword

```
assert_equals(81,sq(y))
p =
  81
```

Testing by passing wrong value

```
assert_equals(82,sq(y))
p =
  81
??? Error using ==> mlunit_fail at 34
Data not equal:
 Expected : 82
 Actual   : 81
Error in ==> abstract_assert_equals at 115
  mlunit_fail(msg);
Error in ==> assert_equals at 42
abstract_assert_equals(true, expected, actual,
varargin{:});
```

Creating Test Case for MLUnit [4]

test_cl1.m

```
function self = test_cl1(name)
%test_cl1 constructor.
%
% Class Info / Example
% ====================
% The class test_cl1 is the fixture for all tests of test-driven
% cl1. The constructor shall not be called , but through
% a test runner.
tc = test_case(name);
self = class(struct([]), 'test_cl1', tc);
```

test_v1

```
function self = test_v1(self)
y=clas1;
y.x=9;
assert_equals(81,sq(y))
assert_equals(80,sq(y))
```

Output :

```
Running suite @test_cl1


p =

    81


p =

    81

  test_v1 FAIL:
    Data not equal:
      Expected : 80
      Actual   : 81
    In test_v1.m at line 6
```

Figure 2:

6. CONCLUSION:

There is a substantial requirement of more upfront executive with today's embedded application layout method. Inside the automotive area, very little upfront examining has become performed. Using the introduction involving executable modeling resources for example MLUnit this specific upfront examining is actually more feasible. It is the task of the tool vendors to make this specific examining technological innovation obtainable along with useful for the end user.

7. REFERENCES:

1. Object oriented software testing by Devid C. Kung http://www.ecs.csun.edu/~rlingard/COMP595VAV/OOSWTesting.pdf
2. Automated Testing tools http://www.guru99.com/automation-testing.html
3. Matlab Documentationhttp://in.mathworks.com/help/matlab/matlab_oop/getting-familiar-with-classes.html
4. ML-Unit Matlab unit Test Framework http://sourceforge.net/p/mlunit/mlunit/HEAD/tree/trunk/
5. Object oriented programming in Matlab http://www.ce.berkeley.edu/~sanjay/e7/oop.pdf
6. Artem, M., Abrahamsson, P., & Ihme, T. (2009). Long-Term Effects of Test-Driven Development A case study. In: *Agile Processes in Software Engineering and Extreme Programming, 10th International Conference, XP 2009,. 31*, pp. 13-22. Pula, Sardinia, Italy: Springer.
7. Bach, J. (2000, November). Session based test management. *Software testing and quality engineering magzine(11/2000)*,(http://www.satisfice.com/articles/sbtm.pdf).
8. Bach, J. (2003). Exploratory Testing Explained, The Test Practitioner 2002,(http://www.satisfice.com/articles/et-article.pdf).
9. Bach, J. (2006). *How to manage and measure exploratory testing.* Quardev Inc.,(http://www.quardev.com/content/whitepapers/how_measure_exploratory_testing.pdf).
10. Basilli, V., & Selby, R. (1987). Comparing the effectiveness of software testing strategies.*IEEE Trans. Software Eng., 13*(12), 1278-1296.
11. Berg, B. L. (2009). *Qualitative Research Methods for the Social Sciences (7th International Edition)* (7th ed.). Boston: Pearson Education.
12. Bernat, G., Gaundel, M. C., & Merre, B. (2007). Software testing based on formal specifications: a theory and tool. In:*Testing Techniques in Software Engineering, Second Pernambuco Summer School on Software Engineering. 6153*, pp. 215-242. Recife: Springer.