International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 7 4463 - 4466

Comparative study of Object oriented and Functional testing using ML-Unit Tool

Mamta
Computer Science and Engineering(MDU)
CBS Group of Institution(CBSGI)
Jhajjar, India
e-mail: mamta.yadav.2817@gmail.com

Upasna
Computer Science and Engineering(MDU)
Ganga Institute of technology and management(GITM)
Jhajjar, Kablana
e-mail: upasnasetia087@gmail.com

***Abstract*—:** Object-oriented languages tend to be good and anyone make application for a fixed set of operations with things in addition to in the same way ones program code evolves, people primarily fill out new things. That is accomplished from adding new classes of which implement existing methods, plus the existing classes usually are left alone. Function languages are generally good as soon as people make application for a fixed set of things, and equally your current program code evolves, a person primarily add new operations in existing things. This is done by adding new function in which compute with existing information types, along with the existing function are generally left are done. Here in our paper we will compare object oriented with functional testing.

Keywords*-*Automation Testing, Functional Testing, Object-Oriented Testing, MATLAB, ML-Unit Tool

*****

I. INTRODUCTION

- Functional programming approach

Throughout computer science, functional programming is a programming paradigm—a style associated with building your own structure as well as elements of computer programs—that treats computation in the same way your evaluation regarding mathematical functions and avoids changing-state and also mutable data. It is a declarative programming paradigm, that means programming is completed inside expressions. In Functional code, your output value of a function depends only towards arguments which are input for the function, therefore calling a function f twice through the same value for an argument x can produce the same result f(x) each time. Eliminating side effects, i.e. changes with state that do not depend towards the function inputs, can make it simpler to help understand as well as predict your behavior of the program, which is to be which is one of the key motivations to its development with functional programming[1].

Many functional programming languages is actually viewed as elaborations with the lambda calculus. Another declarative programming paradigm, logic programming, is based with relations[8]. .

- Object Oriented programming Approach

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which are data structures that contain data, in the form of fields, often known as attributes*;* and code, in the form of procedures, often known as methods[4].

A great distinguishing feature associated with objects can be access often modify your own data fields of an object with which these are generally associated. In OO programming, computer programs are formulated by generating them out associated with objects that idea interact throughout object another.

There is actually important diversity in object-oriented programming, but all popular languages tend to be class-based, meaning that the idea objects are generally instances involving classes, that also determines their type.

Many of an almost all widely used programming languages are generally multi-paradigm programming language this assist object-oriented programming in order to a good far better or even lesser degree, typically inside combination within imperative, procedural programming. Essential object-oriented languages include Python, C++, Objective-C, Smalltalk, Delphi, Java, Swift, C#, Perl, Ruby in addition to PHP.

II. REAL-WORLD MODELING AND RELATIONSHIPS

OOP can be used in order to connect real-world objects and processes inside digital counterparts[2]. However, not anybody agrees it OOP facilitates directly real-world mapping or maybe it real-world mapping is usually even a good worthy goal; Bertrand Meyer argues in Object-Oriented Software Construction This is a method will be not a good model of your world but a good model connected with a series of area of the world; "Reality is often a cousin twice removed". on the same time, a number of principal limitations of OOP had been noted. For example, ones circle-ellipse problem is actually challenging to help handle employing OOP's title of inheritance. However, Niklaus Wirth (who popularized the adage currently known just like Wirth's law: "Software is actually finding slower more rapidly in comparison with hardware becomes faster") said of OOP in his paper, "Good Ideas because of the Looking Glass", "This paradigm closely reflects ones structure involving systems 'in your current real world', and also this can be therefore properly ideal to model complex systems with complex behaviours"[8].

III. THE TEST MODEL AND ITS CAPABILITIES

The tools regarding automated testing based on top of certain machines regarding software/programs and also algorithms [5].

his mathematically defined test model contains following kinds associated with diagrams[10]:

3.1 Class Diagram: A class diagram or maybe a good object relation diagram (ORD) represents ones relationships between the numerous classes among with it is type. Kinds of relationships are usually mainly: inheritance, aggregation, in addition to association. Inside object oriented programs You will find three different relationships between classes they are inheritance, aggregation addition to association[10].

3.2 Control Flow Graph: A control flow graph represents your control structure of an member function and the interface for you to other member is functional so tester may know which taken and/or updated along with which some other is tend to be function invoked by the member function[10].

3.3 State Transition Diagram: A STD or even a Object State Diagram (OSD) represents your state behavior of your object class. Now the state of the class can be embodied in their member variables that happen to be shared among the methods. The OSD shows the various states of a class (various member variable values), and also transitions between them (method invocations)[10].

IV. Object Oriented Unit Testing

- smallest testable unit could be the encapsulated class as well as object[1]
- similar to system testing of conventional software
- do not test operations in isolation from one another[4]
- driven by class operations and state behavior, not algorithmic detail or data flow across module interface
- focuses in groups of classes that collaborate or communicate with number of manner[4]
- integration of operations one at a time into classes is often meaningless
- regression testing is important just like each thread, cluster, or subsystem can be added to the system
- thread-based testing
    - testing all classes needed to respond to help one system input or event
- use-based testing
    - test independent classes first
    - test dependent classes making using them next
- cluster testing
    - groups associated with collaborating classes are tested for interaction errors

Object Oriented Validation Testing

- focuses on visible user action as well as user recognizable outputs from the system[9]
- validation tests are usually based on OOA
    - use-case scenarios
    - object-behavior model
    - event flow diagram
- conventional black-box testing methods can be used to drive the validation tests

OO Test Case Design

- Each test case should become uniquely identified and be explicitly associated with a class to be tested[4]
- State your purpose with each test
- List the testing details for each test

OO Test Case Detail

- states to examine for each object involved
- messages and operations to help exercised like a consequence of the test[4]
- exceptions that may occur anytime when our object is tested
- external conditions required to be changed for the test
- supplementary information necessary to understand or even implement the test

OO Test Design Issues

- White-box testing methods is usually applied to applied to testing the code used to implement class operations, but not much else[4]
- Black-box testing methods are appropriate for testing OO systems

OOP Testing Concerns

- classes may contain operations that happen to be inherited from super classes[1]
- subclasses may certainly operations that were redefined rather than inherited
- all classes derived from a before verified tested base class need to be thoroughly tested

Interclass Test Case Design Multiple Class Testing

- for each client class operate the list of class operators to develop random test sequences that send messages to other server classes

 ISSN: 2321-8169
4463 - 4466

- for each message generated determine your current collaborator class and the corresponding server object operator
- for each server class operator (invoked coming from client object message) determine the message it transmits
- for each message, determine the next level of operators that are invoked as well as incorporate them into the test sequence

Interclass Test Case Design Behavior Model Testing

- test cases must cover many states in the state transition diagram[8].
- breadth first traversal of the state model can be used (test one transition at a time and only make use of previously verified transitions when testing a whole new transition)
- test cases may also be derived to ensure that all behaviors for its class have been adequately exercised

V. IMPLEMENTATION OF AUTOMATED TESTING IN MATLAB USING MLUNIT

mlunit originally began as an update to mlUnit (http://sourceforge.net/projects/mlunit/), also available from MATLAB Central file exchange[7].

The purpose was to add support for the new "classdef" style classes in MATLAB 2008a[6]. Creating tests involves subclassing a class named TestCase, then adding methods whose names begin with "test". Inside each method you can used validation methods (assert, assertEquals, assertNotEquals) to check for success or failure. All tests are run automatically and their results recorded and reported after the run[7].

(1.) Function based testing[7]

Code of Prime.m[3]

```
function primalitytest = prime(n)
m = 2; % initialise factor to test
flag=0;
for m = 2:floor(sqrt(n))
   if mod(n,m) == 0 %m is a factor of n
      flag=1;
   end
end;
if(flag==1)
primalitytest='No';
else
primalitytest='Yes';
end
```

```
>> assert_equals('Yes',prime(3))
>> assert_equals('Yes',prime(4))
??? Error using ==> mlunit_fail at 34
Data not equal:
  Expected : 'Yes'
  Actual   : 'No'
  Changes  :  ^^^

Error in ==> abstract_assert_equals at 115
   mlunit_fail(msg);

Error in ==> assert_equals at 42
abstract_assert_equals(true, expected, actual, varargin{:});
```

(2.)Class based Testing[7]
Testing of BasicClass.m

```
classdef BasicClass
  properties
    Value
  end
  methods
    function r1 = roundOff(obj)
      r1 = round([obj.Value],2);
    end
    function r = multiplyBy(obj,n)
      r = [obj.Value] * n;
    end
  end
end
```

Accessing method from class using object :
Step1: create object of class.
Step2: assign value to instance variable of class using object.
Step3: access method from class and pass object as an argument[7].

```
>> x=BasicClass

x =

    BasicClass

>> x.Value=8

x =

    BasicClass

>> multiplyBy(x,4)

ans =

    32
```

 ISSN: 2321-8169
4463 - 4466

Using assertion on object of class

```
>> assert_equals(32,multiplyBy(x,4))
>> assert_equals(22,multiplyBy(x,4))
??? Error using ==> mlunit_fail at 34
Data not equal:
  Expected : 22
  Actual   : 32

Error in ==> abstract_assert_equals at 115
   mlunit_fail(msg);

Error in ==> assert_equals at 42
abstract_assert_equals(true, expected, actual, varargin{:});
```

VI. FUTURE SCOPE

- Automation will not involve People intervention.
- You possibly can work programmed examination unattended (overnight).
- Automation increases rate involving examination setup[5].
- Automation allows enhance Test Coverage.
- Manual Testing Examining can be uninteresting and therefore error prone[5]

VII. CONCLUSION

By the benefits involving executable modeling tools this actual upfront testing will be more feasible. That is the work of your tool services for you to make the particular testing technology available along with convenient or practical to the user. Within Object Oriented environment ones main troublemakers[8].
That cause problem for testing are Inheritance, Polymorphism and also Encapsulation. My spouse and i studied the problem that are developed from these elements. A detailed thorough study of the testing strategies available to test on programs designed under OO environment continues to be made[4]. Several of a specialized techniques available to confirm OO software have also been discussed. And ML-Unit play a major role to do these type of testing[7].

REFERENCES

[1] Basilli, V., & Selby, R. (1987). Comparing the effectiveness of software testing strategies. *IEEE Trans. Software Eng., 13*(12), 1278-1296.
[2] Berg, B. L. (2009). *Qualitative Research Methods for the Social Sciences (7th International Edition)* (7th ed.). Boston: Pearson Education.
[3] 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.
[4] Object Oriented software testing by Devid C. Kung http://www.ecs.csun.edu/~rlingard/COMP595VAV/OOSW Testing.
[5] Automated Testing tools http://www.guru99.com/automation-testing.html
[6] Matlab Documentation http://in.mathworks.com/help/matlab/matlab_oop/getting-familiar-with-classes.html
[7] ML-Unit Matlab unit Test Framework http://sourceforge.net/p/mlunit/mlunit/HEAD/tree/trunk/
[8] Object Oriented programming in Matlab http://www.ce.berkeley.edu/~sanjay/e7/
[9] Bach, J. (2000, November). Session based test management. *Software testing and quality engineering magzine(11/2000)*.
[10] Basilli, V., & Selby, R. (1987). Comparing the effectiveness of software testing strategies. *IEEE Trans. Software Eng., 13*(12), 1278-1296.