Eliminating Software Development Constraints through Software Testing and Integration

D. O. Egete[1], Ele, Sylvester I.[2] and B. I. Ele[3]
[1, 2, 3]*Department of Computer Science, University of Calabar, Nigeria.*

***Abstract*:-The elimination of software development constraints through software testing and integration is very significant in order to completely differentiate the constraints that we experience during the testing and integration of software development, because the procedures are inter-woven. However, it is advisable to separate the procedure of testing and integration method into different modules or sub-routines before combining it together as to ensure error-free software application that will be easier to work with and attest to its meeting the client specifications.**

***Keywords*: Software, Testing, Integration, Modules, Sub-routine, Procedure**

I. INTRODUCTION

Software testing is an investigation conducted to provide the user with information about the quality of the application or service under test. Software testing can provide objective independent result that allows the client to appreciate and understand the risks to develop software application and implementation (Kolawa, and Dorota, 2007). The process of evaluating software application it's to detect the differences between given input and expected output; also to assess the quality rate of the software item. Software testing is a process that should be done during the development process. In other words, software testing is a verification and validation procedure within the development process (Myers, 2009)

Testing will not completely identify all the defects within software. Instead, it furnishes a criticism or comparison against the state and behavior of an application package based on the process that will allowed the user to identify the error. These processes include specifications, comparable products, past versions of the same product; inferences about the expected customer, required standards (Cai. 2008). In 1956 the software application are integrated together without testing the application package module by module, this process does not allowed the tester or user to carried out debugging exercise freely. However; from 1957 -1978, the bulk debugging exercise of the software application was demonstrations again and its fail to debug effectively, which has make the software application package not to meets the client specification. (Hetzel, 2012).

II. ANALYSIS AND METHODOLOGY

The constraint experience by the programmer during the development of software application, is cause by integrating all the program source code of each modules together before commencing testing to detect errors and it become difficult to identify which modules the errors are coming from and how to debug it, as a result of bulk testing method used during the execution process that end up the programmer not meeting the client specification.

Hence; the primary purpose of testing software program is to know if the program is error free or not, if errors are detected then it should be debugged or corrected. Program software testing consists of examining the behavior of the program source code during execution process toward meeting the specifications (Littman, Szepesvari, 2007). Since the software development is a team work and each module of the software application can be develop by one person or an organization, before integrating it together in other to achieve the set goal. Therefore; there is a need for each developer to test its module of software program to confirm the source code and to ensure that any error detected during the process be debug or corrected, to make the software program error free and meet it specification (Falk and Hung 2011).

There are criteria to be considered when carrying out the debugging procedure to achieve quality computer software, these include: correctness, completeness and security of the program source code as well as technical requirements such as capability, reliability, efficiency, portability, maintainability and usability (Beizer, 2008). The dynamic analysis of the software source code, is the data use to test the software program and to confirm if it has meet the required specification before it being integrated. The emphasis is to achieve the error free testing capacity of the Software source code to verification and validation (Leitner, Ciupa, Oriol, Meyer and Fiva.2009).

Verification: Is the operation of testing the software application to see whether it truly expresses it consistency and meet its specification.

Validation: Is the process to confirm the software application if it has meet the specified requirement by the user for effective implementation.

Bottom-Up Testing: This process is used to integrate the module by module procedures testing of the program source code before compiling it together for the development of a complete computer software application that are errors free and meet the client specification. It enhances the degree of satisfaction and efficiency of actualizing the easier method use to eliminate the constraints face during testing and integration of program software.

Integration Testing: this processes deal with the combination of the interfaces and the interaction of the integrals components model of the software element and the architectural design that are integrated to confirm if the software has met the user/client specification .Therefore; every software application product has a target audience; example, the audience for video game software application are different from the audience of the financial institution software application. The software application develop can be accessed through the end user to determine the acceptability of the product and to confirm its errors free and easier access to it operational technique. One may need to determine if it's user friendly before recommending the software application to the general society for patronage.

Redundant- test detector: This testing tool generates a large number of test inputs that demonstrate the different sequences of the method use in the interface during module by module test. However a large portion of these difference sequences method are use to execute input values that argued the object states of the receiver. This process is to confirm the test being carried out on others module input values for the development of quality software product.

Non- redundant- test generator. This process is to avoid generating redundant-tests that will not be used to explore the actual symbolic object state of receiver space for normal program software execution. One should check the module testing tool that will continue to its concrete state. Furthermore; help in controlling the complex data structures test for effective execution.

Visual testing: The aim of visual testing is to provide developers with the ability to examine the software and determine the feature success or failure of the giving data test if it has met it specification. Visual testing methods require greater communication between testers and developers because the test can be used by many individuals involved in the development process to provide detailed reports and feedback record on screen for user actions.

III. METHOD OF ELIMINATING THE CONSTRAINTS FOR TESTING AND INTEGRATION

The best method used to eliminate the constraints experiences during testing and integration is sub routines programming format, to carry out the test and to avoid the traditional way of testing and integration technique. Therefore; sub-routines test used is to ensure error-free software which is easier to work with and meet the client specifications. Furthermore; it also allows the checking of the software program line by line in other to detect errors that will flag off or get displayed on screen. The line code where the error occurs allows the debugging/correction of that error to take place effectively without necessary starting all over the processes again. However; the combination of module by module method with code coverage method allow the update quality of debugging program source code to view the errors on screen during the execution processes and to accept the commands from keyboard to debugged the errors line by line until its complete the lifecycle of the system. As shown in the figure below:

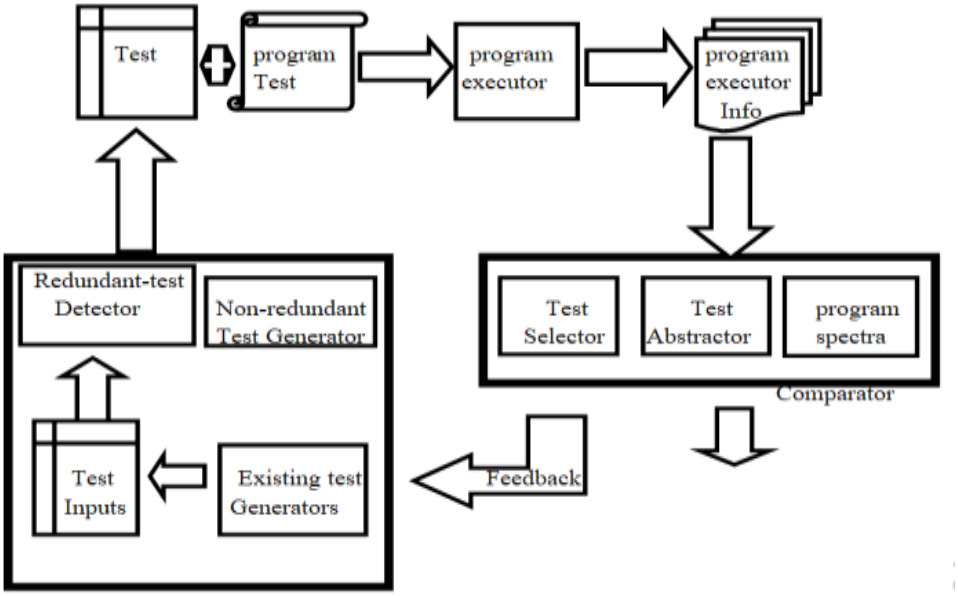

Figure: the structure used to eliminate the constraints face with Testing and Integration

Test adequacy provides stopping rules for the tester to determine the correct step to follow in satisfying the testing requirement before carried out the test using module by module method to achieve quality output that will meet the specification of the software. Therefore; after successfully debugging the errors generated by input variable through the tools and the rules provided by test adequacy, it will now be easier to integrate all the modules together to make it complete software application with errors free that satisfy the client or user.

Result/Discussion: The test engineer took the code coverage test result as an input variable to configure the process use in executing the program software line by line in order to detect the errors and generating report for proper documentation.

Test abstractor: It's received the report generated from the test engineer during the testing of the software program and to know the actual technique use to interface and abstract the errors for easier debugging.

Program-spectra comparator: This method compare the difference procedure use in testing the program source code that will actualized the errors free outputs, before integrating the program together to develop a complete software application that will meet the specification for accuracy and simplicity throughout the execution processes. The tool enhances the programmer to edit its program effectively.

Example: write a computer program to solve for the valve y from the equation below

$$Y = \sqrt{x^2 + 4x - 7,} \quad \text{where x is 1, 2}$$

1. Solution: A

```
05 Rem: program source code
10 Rem: To compute the valve of Y
15 Rem: Y= 0 initialization of Y
20 for I = 1 to 2
25 Z = sqr ( x(I)^2 ) + (4*x(I) – 7 )
30 Y = sqr(Z)
35 print "Y =" Y
40 next I
45 STOP
50 END.
```

The computations in A solution above cannot be easily use to debug an error freely when its flag off errors because the software program is not written in module format.

2. Solution: B

```
10 Rem: program source code
20 Rem: To compute for the valve of Y
30 Rem: initialization Y = sum
40 Rem: assign a, b, c to each module
50 for I = 1 to 2
60 a = (x(I)^2)
70 b = (4* x(I))
80 c = a + b - 7
90 sum = sqr(c)
110 Y = sum
120 print "Y =" Y
130 next I
140 STOP
150 END.
```

The computation in solution B above it's easier to identify and debug errors freely when its flag off errors because the software program is written in module by module format and errors are being debug line by line to made its user friendly.

IV. CONCLUSION

The programmer should always give priority to testing of the software program source code in module by module format before integrating the entire program modules together for easier development of an error free software application package that will meet the client specification; this procedure reduces the constraints faces by the programmer during testing and integration for error free debugging exercises.

Therefore; using sub-routines test is to ensure error-free software application also allows the checking of the software program line by line in other to detect errors that will flag off or get displayed on screen; the line code where the error occurs allows the debugging/correction of that error to take place effectively without necessary starting all over the processes again. the combination of module by module method with code coverage method allow the update quality of debugging program source code to view the errors on screen during the execution processes and to accept the commands from keyboard to debugged the errors line by line until its complete the lifecycle. Therefore; the difference between existing testing and integration software method is the independently technique use in detecting error throughout debugging exercises of software program line by line in each module for the period of the testing processes that gives error free software application before compilation and always meet our client specification without unnecessary delay.

REFERENCES

[1]. Beizer, B. (2008). *Software Testing Techniques* (Second ed.). pp. 21, 430 Van Nostrand Reinhold , New York.

[2]. Cai, W.: Extension Set and Incompatible Problems. Science Exploration 3(1), 83–97 (2008)
[3]. Hetzel,H (2012) The complete guide to software testing. Massachusetts, Mit Press.US
[4]. Kaner, C; Falk, J and Hung,N (2011). *Testing Computer Software, 2nd Ed*. pp. 480 pages. John Wiley and Sons, Inc .New York.
[5]. Kolawa, A; and Dorota,H (2010). *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press. pp. 41–43.
[6]. Leitner, A., Ciupa, I., Oriol, M., Meyer, B., Fiva, A.(2007) "Contract Driven Development Test Driven Development- Writing Test Cases", Proceedings of ESEC/FSE'07: European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering,(Dubrovnik,Croatia), September 2007
[7]. Littman, M., Szepesvari, C.: A Generalized Reinforcement Learning Model: Convergence and applications. In: 13th International Conference Machine Learning, pp. 310–318. IEEE Press, New York (2007)
[8]. Myers,G.J.(2003) Debugging and testing of. halstin: halted press.USA
[9]. Wang, M.H., Hung, C.P.: Extension Neural Network. In: Proceedings of the International Joint Conference on Neural Networks, vol. 1, pp. 399–403 (2010)