ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Special Issue 9, May 2017

Software Testing: Reuse and Open-Source

Riddhiman Ghosh[1], Jai Pratap Dixit[2], Dr. P.K. Dwivedi, Alok Mishra

Assistant Professor, Department of Information Technology, Ambalika Institute of Management and Technology, Lucknow, Uttar Pradesh, India[1,2,4]

Professor, Department of Applied Science, Ambalika Institute of Management and Technology, Lucknow, Uttar Pradesh, India[3]

ABSTRACT: Software testing phase is considered as the most difficult and time consuming phase. Testing is the process to execute a system in order to identify any gaps, errors or missing requirements and also find that whether it satisfies the specified requirements or not. Reuse is to use an item again after it has been used. This includes conventional reuse where the item is used again for the same function and new-life reuse where it is used for a different function. Reuse is the concept that reduces implementation time and increase the reasonableness. Subroutines or functions are the simplest form of reuse. By applying reuse in the testing phase, it will reduce the total software development cost. Open source is a domain whose elements are available to the general public for use and/or modification from its original view.

Our papers is an analysis of software testing and reuse using open source and also prefer a model that minimize the testing cost of any software project.

KEYWORDS: Open Source, Reuse, Software Testing, Warehouse, Development Cost.

I. INTRODUCTION

Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Software testing can be stated as the process of validating and verifying that a computer program/application/product:

- meets the requirements that guided its design and development,
- works as expected,
- can be implemented with the same characteristics,
- and satisfies the needs of stakeholders.

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software. The difficulty in software testing stems from the complexity of software: we cannot completely test a program with moderate complexity. Testing is more than just debugging. The software productivity and quality can be improved by the systematic reuse of software. Reuse is good in nature. This paper has the brief information about the Reuse and Testing. The testing and reuse model is beneficial to develop a low cost project.

II. SOFTWARE REUSE

Software reuse is the process of creating software systems from existing software rather than building software systems from scratch. This simple yet powerful vision was introduced in 1968. Software reuse has, however, failed to become a standard software engineering practice. In an attempt to understand why, researchers have renewed their interest in software reuse and in the obstacles to implementing it.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Special Issue 9, May 2017

Software component reuse is the key to significant gains in productivity. However, to achieve its full potential, we need to focus our attention on development for reuse, which is a process of producing potentially reusable components. We know clearly the difficulties that are faced when trying to reuse a component that is not designed for reuse.
Therefore, the emphasis of the research described here is on development for reuse rather than development with reuse, which is a process of normal systems development (i.e., existing form of reuse). The process of developing potentially reusable components depends solely on defining their characteristics such as language features and domain abstractions. Reuse guidelines can represent such characteristics clearly. Therefore, we need to formulate objective and automatable reuse guidelines.
There have been previous studies on reuse guidelines (Booch 1987; Gautier and Wallis 1990; Braun and Goodenough 1985; Dennis 1987; Hooper and Chester 1991), but these authors emphasize on general advice including documentation and management issues; their guidelines are sometimes unrealizable and contradictory.
In this paper, we will explore the general area of development for reuse and discuss how we can formulate realizable and objective reuse guidelines. We will also review some of these existing guidelines and present our guidelines. Why do we need such objective and realizable reuse guidelines? They are important for:

- Assessing the reusability of software components against objective reuse guidelines.
- Providing reuse advice and analysis.
- Improving components for reuse which is the process of modifying and adding reusability attributes.

In our work, reuse guidelines fall into two classes:

1. Language-oriented reuse guidelines: Most existing programming languages including object-oriented languages provide features that support reuse. However, simply writing code in those languages doesn't promote reusability. Components must be designed for reusability using those features. Such features must be listed as a set of design techniques for reusability before design takes place.
2. Domain-oriented reuse guidelines: Guidelines that are relevant to a specific application domain. We discuss more on this in a later section of this paper.

The language we have chosen for study is Ada, and the application domain chosen is components of abstract data structures (ADS). The main reason for choosing Ada is because of its explicit technical support for reuse, features such as the packaging mechanism, generics, support for abstraction, exceptions, parameterization, building blocks, and information hiding. The reason for choosing ADS as the application domain is partly because, as computer scientists, we might be considered domain experts ourselves in this area and partly because it has been extensively studied and documented. These components are the fundamental building blocks for many applications.

III. CONCLUSION OF VARIOUS RESEARCH PAPERS ON SOFTWARE REUSE

Although reusability guidelines and module-oriented metrics provide an intuitive feel for the general reusability of a component, we need to work on proving that they actually reflect a component's reuse potential. Until researchers can agree on this issue, we will not develop a uniform metric. Existing techniques show a wide range of ways to address this problem, ranging from empirical to qualitative methods. So far, the results mostly indicate a need to first define a suitable scope for future research. This scope must include the affects of domain and environment when we talk about measuring the "reusability" of individual components.[1]
A reuse program must be planned, deliberate, and systematic if it is to give large payoffs. As organizations implement systematic software reuse programs in an effort to improve productivity and quality, they must be able to measure their progress and identify the most effective reuse strategies. In this article we surveyed metrics and models of software reuse. A metric is a quantitative indicator of an attribute. A model specifies relationships between metrics.[2]
With a specific end goal, which is to fabricate value reusable product and achieve the most incredibly addition from reuse, standard coding drills and code documentation should exist opposite group these benchmarks help understand each and every assets quickly. if proper procedures are conjured and the essential arranging move along at a comfortable pace declines plus partnered activity cost. All the aforementioned profits in the extended run can drastically expand profit in a conglomeration and reduction the on the whole hazard of undertaking advancement by supplying a robust group from which all resulting unit house parts are determined. Much has been finished, but there is still much to do when the vision of preferred framework rising via reuse and area building is totally accomplished.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Special Issue 9, May 2017

However most groups reuse segments to recovery the time and expense, reuse is never hazard unlimited. The effort required for reuse of existing software depends on several factors that must be well understood before a determination to reuse functionality is made.
Reuse involves three activities redesign, reimplementation and retesting. Research is needed to identify and validate measures of reusability including good ways to estimate the number of potential reuses. The biggest new technical challenge on a product line approach is the initial design of the software architecture for robustness towards potential future expansion and its subsequent maintenance to deal with technology change validating and extending the identified metrics with focus on incremental development of component based system can be the subject of future work. [3]
Reusability increases with increase in DIT and NOC but it decreases with increase in CBO. In this paper, we have proposed an approach to measure the reusability of object oriented program based upon CK metrics (DIT, NOC & CBO). Since reusability is an attribute of software quality, we can quantify software quality by measuring software reusability. Hence, this approach is important to measure reusability of class diagram. [4]

Many surveys on software testing and reuse have been carried out and it has been seen that reuse is not attempted in testing phase as in other phases of software development. Although some tools and test design patterns are being used in the field of software testing, but are not enough to say that reuse is being successfully applied in testing phase. In this paper, we have shown that there are various possibilities of reuse in testing process. Even, all the activities, carried out during software testing, have large potential for reuse. Although the whole testing process cannot be automated, testers can be aided through test design patterns, templates and test environments. We have highlighted the artifacts that can be reused during testing phase. These testing artifacts should be stored in reuse repository for access to testers. We have also explained the categorization scheme for different reusable test artifacts. This classification will help testers to locate and reuse these artifacts easily. [5]
All the above conclusions point the method to develop software by using the previously designed code. The reusability is the best method to develop a low cost project. There are various methods to apply Reuse, but it depends on project to project.

IV. SOFTWARE TESTING REUSE

In software Development Life Cycle, Reuse is frequently used. But the contribution of reuse in testing phase is very less. When software is going to test it allow various test cases. These test cases are designed by the testers at the time of testing. The requirement of the project is also considered.
By using the concept of database and data mining, previous test cases can be store at a place. The previous information is very helpful to design the test cases for future aspects. There are various conclusion of research papers as –

V. CONCLUSION OF VARIOUS RESEARCH PAPERS ON SOFTWARE TESTING REUSE

Software testing is a component of software quality control (SQC). SQC means control the quality of software engineering products, which is conducting using tests of the software system. These tests can be: unit tests (this testing checks each coded module for the presence of bugs), integration tests (interconnects sets of previously tested modules to ensure that the sets behave as well as they did as independently tested modules), or system tests (checks that the entire software system embedded in its actual hardware environment behaves according to the requirements. [6]
Software testing is an important technique for the improvement and measurement of a software system quality. But it is really not possible to find out all the errors in the program. So, the fundamental question arises, which strategy we would adopt to test. [7]
Except testing phase, Reuse is frequently used in all phases of software development life cycle. There are many tools and test design patterns are used in the field of software test. But it is not sufficient to apply reuse in testing phase. Our paper describes the various possibilities of reuse in testing process. During software testing, all the activities have large potential for reuse. Testing cannot be automated. But the database can be updated by inserting test design patterns, templates and testing environment. The artifacts are highlighted that can be reuse in testing. The classification is necessary for distinguish the class of test as well as help the testers to reuse and locate these artifacts. There is a

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Special Issue 9, May 2017

possibility to use this concept to develop a model that is used for testing the software projects and access is open source. So that it will very helpful to reduce the testing cost as well as software project cost. [8]
Reuse of applications and components reduce the cost of test. The savings occur from reductions in development, integration, documentation, maintenance and training. Reuse occurs with internally developed applications that are reused on other projects and commercially available applications that are internally deployed for maximum benefit. When developing for reuse, instigate phased releases to motivated users, incorporate feedback in the development process, design for expandability and keep users happy. Acceptance is as important to successful reuse as is the application's capability. If it seems to be hard to use, it won't be (except by force). Design a rich set of functionality with a minimal learning curve. Make it easy to incorporate into existing processes and procedures. Provide interfaces that encourage its integration into even more powerful systems. Designing and developing for reuse is possible. At last count there were over 15 groups or projects using the test executive. What is required is a vision of what is necessary and a structured process for achieving it. Strong internal support for the approach is necessary, along with a compelling reason for adoption. [9]
Hence, the Testing model and Reuse is the current topic to research. There are various mythologies to develop a Test Reuse concept. But if this concept is Integrated to the open source era. The Model will be n time effective and efficient.

VI. OPEN SOURCE MODEL FOR TESTING

This paper is also proposing a model and promotes the testing in open source environment. The concept of model is defined as-
Let us consider that, there are three software industries S1, S2 and S3. And they held various software projects. Project P11, P12, P13, P14 are governed by industry S1. Project P21, P22, P23, P24 are governed by industry S2. Project P31, P32, P33, P34 are governed by industry S3. To test the project P11, industry S1 has to design various test cases. And for project P12, Industry S1 has to design the test cases again.
For each project, new test cases are to be designed. Similarly Industry S2 and S3 have to design new test cases to check their respective projects. Various research paper focus their study on design a database for testing which help the industry as well as the new testers to test the projects. Previous test cases are stored in the database. And if a new tester wants to test the project, first it check the domain* of the project. If the domain is available in the database, testers fetch the information about the domain. And apply the checking and testing the project. This process may save a lot of time to design the tests. If the time consume by the project is less, the cost of the project is less.
But this model is the approach to design the open source testing by following the open source software development. Open source software development is open for the entire user, either they are technical or non technical. But Open Source Testing provides the access only for software industries. If the industry follows the open source testing model, then there is no time required to design the test case. If the testing time of a project minimum. Then the cost of whole project can be optimized.
In the database, there are various classes. A class is a similar kind of objects. For e.g. there are various classes such as Modules , Designs, Documents, Codes etc. There is a procedure to use this model.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Special Issue 9, May 2017

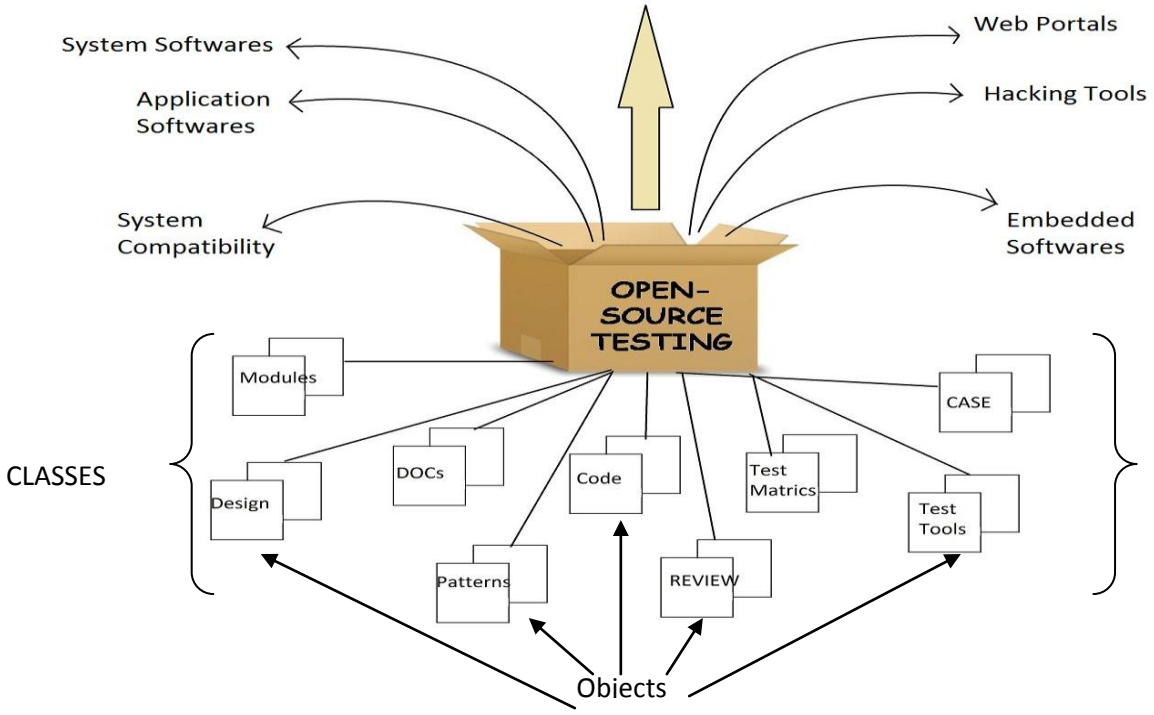

Fig- Open Source Testing Model

i. Classify the type of project. Similar type of projects are bound together. For e.g. all bill dispatching software for commercial use is to be classified in a same domain.
ii. The test cases for the different modules (SRS, Code, Design etc) are store at their particular object position.
iii. In this database, only genuine test cases will remain store.
iv. Now, If a new user wants to test the software. The test cases may access by recognizing the classification of his project.

VII. FUTURE WORK

If we integrate this concept to OLAP and OLTP model, we can access the information about the testing quickly. In future there are various aspects to discuss.

- How to insert and update the classes and cases effectively and efficiently.
- A mechanism to reduce the access time.
- As the model is open source, how to tackle with the fake information.
- What make the industry to share their resources to others.

VIII. CONCLUSION

The Paper is proposing a concept to develop the open source model for testing. Many develops says Testing and Maintenance is the largest phase. And this model is proposed to reduce the time consumed by this phase. As the model is developed and implemented various challenges and errors take place. Testing and reuse using open source can also helpful to develop a method by using that we are able to reduce the software development cost. As the cost is reduce the customer and industry both are able to establish a low cost project development and healthy relation with customers.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Special Issue 9, May 2017

REFERENCES

[1] Jeffrey S. Poulin ,1993 Measuring Software Reusability , poulinj@lfs.loral.com or MD 0124, Loral Federal Systems–Owego, New York, 13827.
[2] WILLIAM FRAKES Virginia Tech, ACM Computing Surveys, Vol. 28, No. 2, June 1996
[3] Mini Bali, Software Reusability promotes High productivity and Increased Quality, International Journal of IT, Engineering and Applied Sciences Research (IJIEASR) ISSN: 2319-4413 Volume 1, No. 1, October 2012
[4] Pradeep Kumar Bhatia, An Approach to Measure Software Reusability of OO Design Proceedings of 2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008) RIMT-IET, Mandi Gobindgarh, March 29, 2008
[5] Meeta Prakash, Possibility of Reuse in Software Testing, "6th annual International Software Testing Conference in India 2006",
[6] Jovanović, Irena, Software Testing Methods and Techniques, 2009 http://www.internetjournals.net/journals/tir/2009/January/Paper%2006.pdf
[7] Mohd. Ehmer Khan , IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 1, May 2010, ISSN (Online): 1694-0784.
[8] Boehm, B., Helping students learn requirements engineering, Proc. Software Engineering Education, 1996, pp. 96 -97
[9] Douglas D. Lonngren, Reducing the Cost of Test Through Reuse, http://www.serendipsys.com/ReduceCost.pdf