A CRITICAL STUDY AND SURVEY ON REUSABILITY TESTING OF SOFTWARE

Devendra Kumar

Assistant Professor, Department of Computer Science & Engineering,

Ambalika Institute of Management & Technology, Lucknow.

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. Our paper represents the analysis and study of various research papers which are based on reuse and software testing.

Keyword: ***Reuse, Software Testing, Test Reuse Database.***

I INTRODUCTION

The software productivity and quality can be improved by the systematic reuse of software. Reusable modules and classes reduce implementation time, increase the reasonableness that prior testing and use has eliminated bugs and localized the code modification when a change in implementation is required. Subroutines or functions are the simplest form of reuse. For example, Lawyers nowadays rarely draft wills from scratch. Instead, they use a word processor to store wills they have previously drafted, and then make appropriate changes to an existing will. Other legal documents, like contracts, are usually drafted in the same way from existing documents.

This paper presents an empirical study of the software reuse activity by expert designers in the context of object-oriented design. Our study focuses on the three following aspects of reusability.

1) The interaction between some design processes, e.g. constructing a problem representation, and reuse processes, i.e. retrieving and using previous solutions.
2) The mental processes involved in reuse, e.g. example based retrieval or bottom-up versus top-down expanding of the solution.
3) The mental representations constructed throughout the reuse activity, e.g. dynamic versus static representation.

Code reuse is the idea that a partial or complete computer program written at one time can be, should be, or is being used in another program written at a later time. The reuse of programming code is a common technique which attempts to save time and energy by reducing redundant work. Reused software is more accurate than new software because already it has been tried and tested in working system [1]. Testing process includes taking test requirements

as input and preparing data for test execution, performing test execution, and evaluating test results. Testing normally requires 50% to 60% effort of total software development. Since testing is a difficult and time consuming activity of the software development there is a need to use, with other things, like the integrated and automated testing tools. Reuse has been considered as a technology that improves productivity and quality.

Initially reuse was considered and used only at coding phase. But later it was found that only reuse at coding phase cannot solve the software crisis. And now reuse is being applied at each phase of software development. These steps are as following:

1) Developing a reuse plan or strategy after studying the problem and available solutions to the problem.
2) Identifying a solution structure for the problem following the reuse plan or strategy.
3) Reconfiguring the solution structure to improve the possibility of using predefined components available at the next phase.
4) Acquiring, instantiating, and modifying predefined components.
5) Integrating the components into the products for this phase, and. evaluating the products. [2]

This paper is an attempt to conclude the testing reuse and its result.

II SOFTWARE PROJECT MANAGEMENT

Software project management is the art and science of planning and leading software projects [3]. It is a sub-discipline of project management in which software projects are planned, implemented, monitored and controlled. A software development process is concerned primarily with the production aspect of software development, as opposed to the technical aspect, such as software tools. These processes exist primarily for supporting the management of software development, and are generally skewed toward addressing business concerns. Many software development processes can be run in a similar way to general project management processes. As a sub discipline of project management, some regard the management of software development akin to the management of manufacturing, which can be performed by someone with management skills, but no programming skills [4].

III SOFTWARE DEVELOPMENT

Software development (also known as application development, software design, designing software, software application development, enterprise application development, or platform development) [5] Software can be developed for a variety of purposes, the three most common being to meet specific needs of a specific client/business (the case with custom software), to meet a perceived need of some set of potential users (the case with commercial and open source software), or for personal use (e.g. a scientist may write software to automate a mundane task). Embedded software development, that is, the development of embedded software such as used for controlling consumer products, requires the development process to be integrated with the development of the controlled physical product. [6]

To develop software, first we have to elucidate the exact requirements of the customers and prepare a document which contained all requirements (what client wants). The documentation is known as SRS. According to SRS developers and designers complete the project. After the development it is checked by the Testing Team.

IV SOFTWARE TESTING

It is expected to test the product in the industry before delivery. Due to testing and fixing faults, failure intensity will come down initially and may stabilize after certain time. Testing comprises the efforts to find defects. Testing does not include efforts associated with tracking down bugs and fixing them. In other words, testing does not include the debugging or repair of bugs. Testing is a procedure of finding faults, defects in the software. While debugging is to rectify the faults, defects find during testing in the software. There exist a vast number of testing techniques and methodologies that help to achieve cost effectiveness for the testing phase. Two basic approaches to testing are: functional and structural. In functional testing the structure of the program is not considered. The test cases are decided solely on the basis of the requirements or specification of the program or module, while in structural approach test cases are generated based on the actual code of the program or modules to be tested. Both testing approaches are complementary to each other .That is both must be carried out to test a system satisfactorily. The intent of structural testing is not to exercise all the different input or output conditions but to exercise the different programming structures and data structures used in the program. Control flow-based criteria, data flow-based criteria, mutation testing etc. are examples of structural testing approach. There are a lots of level at which reusability testing is considered by various researcher scholars.

V SOFTWARE REUSE

Reuse refers to using components of one product to facilitate the development of a different product with a different functionality. A reusable component need not necessarily be a module or a code fragment—it could be a design, a part of a manual, a set of test data, or duration and cost estimate. Code reuse, also called software reuse, is the use of existing software, or software knowledge, to build new software. [7] The term 'reuse' was proposed at the NATO software Engineering Conference in 1968. Code reuse is the idea that a partial computer program written at one time can be, should be, or is being used in another program written at a later time. The reuse of programming code is a common technique which attempts to save time and energy by reducing redundant work [8]. The Assets might be product parts, programming prerequisite dissection manuals, and plan models, database composition, protests, code documentation, area construction modeling, norms, test situations, and arrangements We infer that reusing of product is developing engineering, which recoveries the processing expense, enhancing the inventive mechanics from the more advanced in years one [9]. The software library is a good example of code reuse. Programmers may decide to create internal abstractions so that certain parts of their program can be reused, or may create custom libraries for their own use. Some characteristics that make software more easily reusable are modularity, loose coupling, high cohesion, information hiding and separation of concerns [10].

If the emphasis is on quality, then it will be advantageous if one has access to well-designed, well-tested, and well documented code that has a proven correct and well-understood behavior. To use the components in future, there is a strategy of a team to design the components in a specific manner. This term is known as Planned Reuse. Generally, Opportunistic reuse has two types:

5.1. Internal reuse

A software industry or a team reuses its own developed module. These develop modules are stored in a record for further reuse.

5.2. External reuse

A software industry or a team hire third party components and licensed them. The cost of licensing the third party is up to 20 percent of the internal development cost. The team require extra time to learn, find and integrate the modules.[10]

VI POSSIBILY OF REUSE IN TESTING PROCESS

Reuse can provide maximum benefit if it was implemented on all phases of Software Development Life Cycle (SDLC). But in practice reuse is frequently use in coding. It is also true that testing process cannot be fully automated. Human involvement is necessary during testing process. All the test cases cannot be applied on all the software projects. If the test cases apply, it is not compulsory that test case provide the positive output. Test cases may vary project to project. Debugging and test analysis also requires human involvement.

VII TEST PLAN GENERATION AND REUSE

It is the first activity in software testing when the implementing process is completed. Testing should be plan at the beginning. Test plan Documents define:-

a. The Scope
b. The Approaches to be taken.
c. The Schedule of Testing.
d. Test Items for Entire Testing Process
e. Personnel Responsibilities.

Testing cannot be fully automated but some module of testing can be automated like-

a. Checking consistency between testing schedule and schedule.
b. Selection of test units from system design and detailed design.

The software projects and their tests are stored in database and accessible in future. This record is very useful for new tester because it will work as experience to the new tester. If a Complex module is to be designed by team and similar type of module is also developed previously. Then the team used to reuse the design and coding of previous module. Similarly, previous test case also applicable to this new module. The Applications belonging to the same domain are stored in database. And testers can apply that information during testing and save a lot of time. The stored information is very useful to design the test plans. And these test plans can be reuse over and over.

VIII TEST CASE GENERATION AND REUSE

The goal of test case design is to create a set of test that are effective in testing. To decide the set of test cases propitiate the criteria or not is very arduous. Server tools are providing feedback in structural testing. It is similar to white box testing. In structural testing derivation of test cases is according to the program structure. Hence the knowledge of program is used to identify the additional test cases. To reuse test cases (by using patterns) is one of the best methods. One of the best examples of testing where test cases are reused, is regression testing.

If any new bug occurs with the test case, it should also store in the database. UML is the best method to describe the test cases. OMG (Object Management Group) standardize UML for industry for standard analysis and design notations. Test cases should be designed specifically for a particular domain, so that one can reuse those test cases. Now the time consumption is only for those phases, which are not developed previously means for new modules.

IX TEST CASE EXECUTION AND REUSE

After designing the test cases, the next step is to execute the time cases. It may require some construction of derive modules. It is a dummy routine that simulates a module. Derivers are needed to set up the appropriate environment and invoke the module, which has no subordinate. These stubs and derivers can also be developed already by identifying them by analyzing the domain and can be reused. For analysis of the execution results data is to be collected. Sometimes, test procedure specification document is used to execute the test cases [11]. This document specifies any special requirements that exist for setting the test environment, the methods and formats for reporting the results of testing and measurements, if needed, along with how to obtain them.

Here is again the possibility of reuse of this specification document among applications belonging to the same domain. Data collection forms and software can be developed in a particular domain and reused. Different types of preprocessors/postprocessors can be constructed based on different kinds of probes: probes that are inserted where a path splits into multiple paths and where multiple paths merge or a different probe for each path. According to need, users may select the one among all. Test oracles are needed to test any program. A test oracle is a mechanism that can be used to check the correctness of the output of the program for the test cases. The output of the two is compared to determine if the program behaved correctly for the test cases.

Test oracles should be automated to give always a correct answer. But it is very difficult to automate test oracles. Systems for which oracles cannot be “automated”, oracles created during testing of previous systems may be used for generating oracles for systems of similar types. Test oracles should be developed as components and put in repositories. These test oracles can be reused either “as-is” or after customization, if needed.

X REUSE IN “ERROR CAUSES” DETECTION AND THEIR REMOVAL

During execution errors are detected. The next step is to locate the cause of errors and correct them. Here also, we have a possibility of reuse of the test summary report generated for a domain during testing of the application belonging to that domain. Test summary report stores the summary of the entire test case execution. The summary

gives the total number of test cases executed, the number and nature of errors found, and a summary of any metrics data (e.g. effort) collected. This report can be reused either at project level or within domain.

XI REUSE IN EVALUATION/REVIEW OF TESTING PROCESS

The last step in testing process is to evaluate/review the testing carried out to determine its quality. The evaluation of testing process is necessary to see whether it has been carried out satisfactorily i.e. carried out thoroughly or not. As with many other verification methods, evaluation of quality of testing is done through "test review". And for any review, a formal document or work product is needed [11]. All the test deliverables are reviewed, using a formal review process, to make sure that the testing has been carried out with the policy specified in the test plan, test cases specification is generated for regression testing etc. In this activity, the formal review process is being reused. If the review will be done in an ad hoc way, that is, if there will not be a formal (repeatable) process, testing evaluation may not be done successfully because testers would always try to show that testing is carried out successfully. The quantitative(s)/quantitative(s) that can be used to measure the quality of the scripts that are developed by Engineers can also be reused in a particular domain. The traditional qualitative(s)/quantitative(s) would be: Religious practice of coding guidelines, Duration taken for a test run, Quality of documentation (Test script comments, Test script description document), Classifying the Review Comments in to different levels (e.g. minor, major, trivial etc.) based on the impact of the same on the script execution, And number of times the script is run on the test harness before the same is run successfully etc.[12]

Thus we have seen that there is much possibility of reuse in testing process. For each activity carried out during testing, test design patterns can be created to store the decisions that should be taken during different situations. Test environment can be developed to aid testers. The test design patterns schema is described as follows (14):

XII SOFTWARE REUSE BENEFITS

1) **Increased dependability** Reused software that has been tried and tested in working systems, should be more dependable than new software. The initial use of the software reveals any design and implementation faults. These are then fixed, thus reducing the number of failures when the software is reused.
2) **Reduced process risk** If software exists, there is less uncertainty in the costs of reusing that software than in the costs of development. This is an important factor for project management as it reduces the margin of error in project cost estimation. This is particularly true when relatively large software components such as sub-systems are reused.
3) **Effective use of specialists** Instead of application specialists doing the same work on different projects these specialists can develop reusable software that encapsulates their knowledge.
4) **Standards compliance** Some standards, such as user interface standards, can be implemented as a set of standard reusable components.
5) **Accelerated development** Bringing a system to market as early as possible is often more important than overall development costs. Reusing software can speed up system production because both development and validation time should be reduced.

6) **Reliability and Safety** Better system reliability is one of the goals of software reuse. It is argued that reusable components, because of more careful design and testing and broader and more extensive usage, can be more reliable that one use equivalents. If so, then it is further argued that using these more reliable components in system architecture can increase the reliability of the system as a whole.

XIII REUSE APPROACHES

1) **Design patterns** Generic abstractions that occur across applications are represented as design patterns that show abstract and concrete objects and interactions.
2) **Component-based development** Systems are developed by integrating components (collections of objects) that conform to component-model standards.
3) **Application frameworks** Collections of abstract and concrete classes that can be adapted and extended to create application systems.
4) **Legacy system wrapping** Legacy systems that can be "wrapped by defining a set of interfaces and providing access to these legacy systems through these interfaces (13).
5) **Service-oriented systems** S-O systems are developed by linking shared services that may be externally provided.
6) **Application product lines** An application type is generalized around a common architecture so that it can be adapted in different ways for different customers.
7) **COTS integration** Systems are developed by integrating existing application systems.
8) **Configurable vertical applications** A generic system is designed so that it can be configured to the needs of specific system customers.
9) **Program libraries** Class and function libraries implementing commonly used abstractions are available for reuse.

XIV TEST REUSE DATABASE AND TEST ARTIFACTS-

We have seen there are different artifacts that can be reused during testing. Two basic types of test knowledge are recognized: generic test knowledge, which applies to all projects, and project specific test knowledge. The arrows in the figure 2 represent the possible kinds of test knowledge that may be placed in the test database and have been already explained above.

Prieto-Diaz and Freeman encouraged white-box reuse and identified five program attributes for evaluating reusability [15]. The attributes used are: Program Size, Program Structure, Program Documentation, Programming Language, and Reuse Experience.

These artifacts must be stored in some reuse library in such a way that they can be accessible to users easily. Artifacts in test reuse library should be categorized in such a way that they can be accessed easily. Categorization of different testing artifacts is explained in figure 3. Some artifacts can be categorized based on domain while others may not. It is shown in column one.

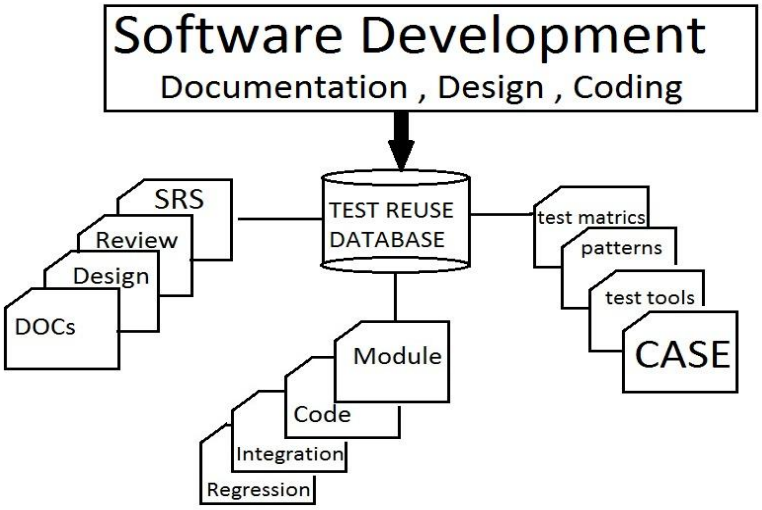

Figure 1: The Test Database

Other criteria for classifications are shown in column second. Second column for some artifacts is empty because they cannot be categorized except based on domain.

Figure 2: Kind of test artifacts [16]

Testing artifacts	**Categorization**	**Examples**
Preprocessors/ postprocessors (According to domain)	According to the type of probes	1. A different probe is inserted for each path. 2. Inset a probe where a path splits into multiple paths and where multiple paths merge. 3. Inset a probe after each line of code etc.
Test cases (According to domain)	1. According to level of testing 2. According to "as-is" reusable versus customizable:	1. Unit testing, integration testing, system testing 2. Test cases (project level reuse), templates, test design patterns.
Testing tools	1. According to activities 2. According to particular testing	1. Test case evaluation tool, probes instrumentation tool etc. 2. Tools for data flow-based testing, mutation testing etc.
Test environments (According to domain)		Environment for unit testing, environment for test plan generation etc.
Test oracles (According to domain)	According to "as-is" reusable versus customizable:	Automated test oracles (test oracle components), test oracle templates
Testing metrics	According to activities	Defect removal efficiency, testability etc.
Testing deliverables (According to domain)		Test case specification report, test summary report, test log etc.

XV CONCLUSIONS

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 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.

REFERENCES

[1] Ajay Kumar (2012) “measuring software reusability using svm based classifier approach”, International Journal of Information Technology and Knowledge Management January-June 2012, Volume 5, No. 1, pp. 205-209.

[2] Anupama Kaur1 , Performance Evaluation of Reusable Software Components, International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 4, April 2012)

[3] Stellman, Andrew; Greene, Jennifer (2005). Applied Software Project Management. O'Reilly Media. ISBN 978-0-596-00948-9.

[4] http://en.wikipedia.org/wiki/Software_project_management

[5] "Application Development White Papers (Development of Software, Software Design, Designing Software, Software Engineering, Software Application Development, Enterprise Application Development, Platform Development, Software Development, Applications Development, Development) Software Downloads Definition and Webcasts". Bitpipe. Retrieved 2012-08-05.

[6] http://en.wikipedia.org/wiki/Software_development

[7] Frakes, W.B. and Kyo Kang, (2005), "Software Reuse Research: Status and Future", IEEE Transactions on Software Engineering, 31(7), July, pp. 529-536.

[8] http://en.wikipedia.org/wiki/Code_reuse

[9] Mini Bali, Software Reusability promotes High productivity and Increased Quality International Journal of IT, Engineering and Applied Sciences Research (IJIEASR) ISSN: 2319-4413.

[10] Neha Budhija and Satinder Pal Ahuja, Review of Software Reusability International Conference on Computer Science and Information Technology (ICCSIT'2011) Pattaya Dec. 2011

[11] Jalote P., “An Integrated Approach to SOFTWARE ENGINEERING”, ISBN 81-7319-271-5, Second Edition, Narosa Publishing House, 2000.

[12] Manjari Gupta, Possibility of Reuse in Software Testing"6th annual International Software Testing Conference in India 2006"

[13] Gotterbam and Rogerson 1998, "The Ethics of Software Project Management", in Ethics and Information Technology, ed. G&an Collste, New Academic Publisher.

[14] W. Lim, Effects of Reuse on Quality, Productivity, and Economics" IEEE Software, 11(5, Oct. 1994), 23-30.

[15] Mayobre, G., 1991 "Using Code Reusability Analysis to Identify Reusable Components from Software Related to an Application Domain", Proceedings 4th Workshop on Software Reuse, Reston. VA.

[16] Manjari Gupta, *Possibility of Reuse in Software Testing*, 6th annual International Software Testing Conference in India 2006.