Towards Reduction of Cost of Software Quality by Implementing Regression Automation

Gopinath Ganapathy and Deepa Vijay
Bharathidasan University, Trichy, India
Email: {gGANAPATHY, deepa.vijay1}@gmail.com

Abstract—The cost of Software quality is very high due to various hidden costs, besides it brings huge losses to the client. The rework cost per bug is very high in post production. The idea of this paper is to bring down the cost of quality without compromising the quality by widening the test coverage and detecting bugs at earlier stages of testing cycle by implementing regression automation approach.

Index Terms—cost of software quality, regression testing, regression automation, component testing

I. INTRODUCTION

Software is one the most important and yet one of the most economically challenging technologies of the current era. Even though many successful software products and systems exist in the world today, an overall lack of attention to quality, skipping regression testing process due to tight delivery schedule. Regression Testing is methodology in which testing is performed on modified program to verify that the changes are correct and to ensure that the changes has not adversely affected the position of program. Small changes in one part of a program may have subtle undesired effects in other seemingly unrelated parts of the program. Full Regression testing consumes lot of time and effort which increases maintenance cost.

Studies shows that due to insufficient of time, regression testing is skipped or poorly executed which resulted poor quality [1]

Regression testing is very mandatory and it should be followed for a defect free release [2]. Many IT firms follow regression testing manually which resulted in increase in testing cycle time. Test cycle time can be reduced by automating the regression test suite; it also reduces failure cost by early deduction of bugs in testing cycle.

II. RELATED WORK AND LITERATURE SURVEY

Researchers and Practitioners proposed various techniques on regression test suite optimization, test case prioritization and regression test selection for improving the cost effectiveness of the regression testing.

David Binkley at [3] described techniques for reducing cost of regression testing by reusing old test cases and old test results using language semantics.

Other methods include root cause analysis [4], software process improvements [5], enhancing uniform coding standards, reducing the regression test cases by Regression Test Selection Technique (RTST) [6]. The proposed method provides the approach for reducing the cost of poor software quality by automating the regression test suite at different phases of testing life cycle there by reducing the regression cycle time, feature certification time, early detection of issues and improves the quality by wider test coverage. Even though the organization has to spend appraisal cost towards automating test cases at component level, integration and UI level considerable amount of savings is expected as the automation tests are reused for all the releases throughout the years.

III. PROPOSED WORK

This paper mainly focuses of two distinct components of Cost of Quality: Appraisal Costs and Failure Costs [7]-[8].

Appraisal Costs: The costs of verifying, checking or evaluating a product or service at the various stages during the delivery process of that product or service to the customer. Appraisal Costs of Quality are the ones we tend to focus on because we have direct control over them.

Failure Costs: The costs incurred by a company because the product or service did not meet the requirements and the product had to be fixed or replaced or the service had to be repeated.

Software firms follow different types of Regression testing approaches like Project with No Regression testing, Projects with Full Regression testing and Optimized Regression Testing.

A. Impacts of Project with No Regression

Regression testing is skipped in most of the projects follows agile methodology, due to insufficient of time, more adhoc requests and tight delivery schedules. New code often reveals more defects than previously tested old code. Studies show that errors are 20% more likely to appear in code that has been just touched to repair a defect [9]. One-fifth of the code repairs introduce new errors. Without regression testing it is difficult to find the
bugs introduced due to defect fix and implementation of new features. Regression testing can be used not only for testing the correctness of a code and also for tracking the quality of its output [10].

As there are more chances of leaking critical bugs into production without proper impact analysis of regression testing, the following are the impacts observed due to skipped Regression testing:

a) Slipped Schedule
b) Loss of Reputation in the market
c) Loss of customer good will

In this approach, Appraisal Cost is reduced but Failure Cost has increased which in turn results in increase in overall cost of quality.

This paper proposes Optimized Regression automation approach which overcomes the above mentioned flaw.

B. Project with Full Regression

Full regression testing involves execution of all the tests in the existing test suite. So the full regression technique [11] is very expensive as compared to other regression techniques. Regression test suites are costly to execute in full as it require more time and budget.

C. Optimized Regression Approach

Due to expensive nature of “Full Regression” technique, Optimized Regression approach is proposed. In this approach instead of rerunning the complete test suite we are selecting part of the test suite. The first step is selecting the test suite based on Critical Path or Risky path scenarios. Then the selected path scenarios should be automated to increase the test coverage and also to reduce regression cycle. IT firm should invest in automation of regression test cases as part of appraisal cost. Once implemented could be reused across all regression testing cycles thereby minimizing manual effort and obtaining the quality product.

D. Optimized Regression Automation

Regression automation is categorized into 3 phases namely Component Level automation, Integration Level Automation and Usability testing.

![Diagram: Proposed Approach for Regression Automation](image)

The inputs required to automate the application:

a) System Requirement Specification – To understand the basic functionality of the software Components.

b) Software Design Document – To understand how many Software Components exists in the System and how they collaborate with each other as a system.

Component Interfaces – To understand the sequence of messages and/or events through which the components collaborate with each other for a given functionality.

1) Component automation testing

What is a software component?

A software component is an independently deliverable piece of functionality providing access to its services through interfaces [12].

2) Role of software component test automation

The role of component testing is illustrated briefly with the following scenario.

Components are unit tested during development cycle and system tested at the end of the iteration by Quality Engineers. In the process of unit testing, newly developed functionality in the module has been tested by development team. The bugs are not detected until system testing is done. The bugs detected in the process of system testing introduce delay in software release.

This flaw could be eliminated by implementing software component testing as and when the software Components or modules are developed. The software components are tested in isolation as well as in conjunction with other components, as integration tests.

Component tests once automated could be executed in different environments (Development, Pre-Production and Production). Components can be tested independently without considering the availability of upper and lower components in Service Level testing.

The activities for Service Level automation include:

a) Preparation of the test environment
b) Evaluation of a possible test automation tool to satisfy the need for test automation.
c) Evaluation of a need of a library / framework to act as middleware between the component /host under test and the simulation environment.
d) Identification of reusable test procedure library
e) Design of test scripts.
f) Creation of report generation libraries
g) Execution of test scripts

It is always suggested to execute all the service level test cases (Regression level) whenever any functionality changes are taking place in any component. This will detect the bugs in services implemented in that component at earlier stages of testing cycle. Rework and retesting the bugs are cheaper in Development environment than testing and Production environments. This reduces the Cost of Software Quality.

3) Integration Automation testing (End to End testing)

Similar to system testing, Integration testing involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, or interacting with other hosts, applications, or systems. End to end testing is performed in an integrated code base of all components.

Advantages of regression automation:
• Lowers testing cycle times and costs and significantly reduce or eliminate manual testing efforts.
• Expand test scenarios to catch more bugs earlier and improve quality.
• Improve productivity by reusing test assets and expanding collaborative testing across the lifecycle.
• Dramatically reduce labor costs and improve time-to-market.

4) Usability Testing

Usability testing is a technique for ensuring that the intended users of a system can carry out the intended tasks efficiently, effectively and satisfactorily. Usability Testing includes the following which affect user's experience:

• How easy is it to navigate through the web application?
• Is it obvious to the user which actions are available to him or her?
• Is the look-and-feel of the web application consistent from page to page, including font sizes and colors?
• Content should be logical and easy to understand.
• Images should be placed properly with proper sizes

Usability testing should be carried out before production release, so that any significant issues identified are addressed. Usability testing should be done manually with respect to end users point of view.

IV. IMPLEMENTATION OF COMPONENT AUTOMATION IN WEB APPLICATION

The proposed Regression Automation test strategy has been implemented in Travel & Hospitality domain project in an IT organization and below model represents piloted project application flow for the period of 6 months. This has been implemented using Selenium and TestNG framework.

This module has many components and each component interacts with each other through service calls. Every component exposes and consumes services.

For example:

Service Orchestration Layer Component:

It exposes services - searchService, repriceService, reservationService, loadReservation, cancelService. It consumes corresponding vertical services. These consumed services will be called from their exposed services.

There are 2 components in Fig. 2. Component A is a UI layer and component B is Service Orchestration layer. Component B exposes different services and consumes 3 vertical services as shown in the architecture diagram. Component B’s services are automated. Component B is tested individually by mocking the input and output services. Component tests are executed in development environment along with impacted component regression tests and integration tested in QA environment. Manual test cases are minimized by automating component and integration level tests, thus reduces the time taken to certify the feature.

Before implementation of this method, defects were encountered during end to end testing which resulted in increase of test cycle time and delayed product release due to rework and retesting the bug fixes. After implementing this approach, defects are found at the earlier stages of testing cycle. It helps the developers to isolate the issue and bugs can be fixed quickly. This approach helps us in increasing the quality of the product in turn reducing the failure cost.

Take away from component automation:

• This implementation process will help us in identifying defects well in advance in testing cycle and isolate the issue with respect to component. This reduces turnaround time and helps developers to fix the defects in earlier stage. This process saves time and cost.
• Test coverage is wider in component testing hence covering more defects and eliminating defect leakage.
• Isolated Validation – There is no dependency on live services. Tests can be executed by using mock data. This reduces the latency time.

Drawbacks of not implementing Component Automation Testing:

Analyzing the issue takes longer time; it could be a day or a week because issue to be identified at different component levels and has to be isolated. This increases the testing cycle time and hence increasing in the cost.

V. RESULTS

After implementing regression automation strategy, it was found that the test cycle time has been reduced and able to deliver defect free product. Since the bugs were detected at earlier stages, cost of fixing the bugs is cheaper resulting in reducing the cost of software quality. As per the general rule of defect fixing cost, the defect fixing cost in various stages is 1:10:100. The defect costs 1 unit to fix in requirements and design; it costs 10 units in system testing and 100 units in production.

Real time defect leakage data were gathered over the period of 6 months as furnished in the below table.
Table I. shows the data captured in real time environment of 5 different teams in Travel and Hospitality domain project before implementation of the proposed approach. The real time data clearly shows that the defect leakage percentage is very high and cost of fixing the defects is also high.

Table II. shows the data captured in the same project post implementation of the proposed approach. The table clearly shows that the defect leakage count is reduced which results in reducing the cost of fixing the defect.

The total failure cost is reduced from $5000 to $2200 for a period of 6 months. Total cost saved is $2800.

Table III. shows the percentage of defects detected in component level testing. The cost of fixing the defect in development environment is much lower than QA and production environments. Cost of fixing the defects detected in component level testing was $300. If the same defects are detected in production environment then the approximate cost of fixing the defect would be $3000.

By detecting more defects earlier in testing life cycle reduces turnaround time for fixing the defects, reduces feature certification time and minimizes rework which in turn increases the quality of the product and reduces failure cost. By implementing the proposed method in all the modules of the project, defect leakage count can be reduced which in turn reduces the cost of poor quality.

VI. CONCLUSION AND FUTURE WORK

An approach has been suggested for reducing the cost of software quality in testing. With the proposed method, it is proved that the cost of software quality could have been reduced by approximately 50%. This experimentation is carried out with one of the module in a big project. If this proposed method is extended across all the modules, then the huge reduction of failure cost could be seen which results in overall reduction of Cost of Software Quality.

REFERENCES


Gopinath Ganapathy, Trichy, India. B.Sc. Computer Science St.Joseph’s College Bharathidasan University,Trichy,India, Masters in Computer Applications, St.Joseph’s College Autonomus, Trichy, India, Ph.D, Madurai Kamaraj University, India.
He is currently the Chair and Head School of Computer Science & Engineering, Bharathidasan University, Trichy, India.
Dr. Gopinath Ganapathy is a Professional Member in IEEE, Professional Member in ACM (USA) and Life member in Computer Society of India (CSI).

Deepa Vijay, Bangalore, India. B.E.S. Electronics Science, Madras University, India, M.Sc. Computer Technology, Periyar University, Salem, India, M.Phil. Computer Science, Alagappa University, India.
Ph.D scholar in Department of Computer Science, Bharathidasan University, Trichy, India.