Functional Testing of Requirements and Code Base

As a means to:

Reduce Cost of Quality and
Increase Return on Investment.

Author:
Aakash Vakil, CSTE

Table of Contents

ABSTRACT

Traditionally we relate 'Defect Prevention' to the coding phase of development life cycle. We are aware of the benefits of preventing coding defects and the advantage it gives in terms of reduced cost and schedule variance.

Highly competitive market along with mature customers, have made the need felt for reducing the product delivery cycle, which would in turn increase the *Return on Investment* (RoI) for clients and reduce the *Cost of Quality* (CoQ) for the organization.

Reduction of delivery cycle would be possible only if we cut back on efforts that organizations spend on re-working and re-visiting requirements that were found to be faulty in coding or testing phase.

We have started to realize the magnitude of impact that incomplete & missed requirements have on the entire product. We have also realized that if defects buried within requirements are captured at the stage of injection – when requirements analysis is carried out, organizations can save lot of budget on the quality control activities that is normally needed.

With this white paper, we are making an attempt to highlight the advantages of sound testing of requirements and the issues which faulty or ambiguous requirements can create, if ignored, during coding and testing phase.

We will also see how a functional approach to unit testing can be utilized to catch functional issues early on in development cycle.

This paper will help organizations become more aware of the situation and take charge of the problem during Requirements Phase & Unit Testing Phase before it develops into a death trap – a point of no return.

Identifying the risks associated with ambiguous requirements

Wikipedia defines Risk as: "A concept which relates to human expectations. It denotes a potential negative impact to an asset or some characteristic of value that may arise from some present process or from some future event."

However, in everyday usage, "risk" is often used synonymously with "probability" of a loss or threat. In categorizing risk, we are actually determining the probability of any negative event (threat) of actually happening.

Risk is present everywhere and in all activities; software is no exception – be it a simple software to scribble notes to complex software's that perform millions of calculations a second.

Now that we know what risk is, let us look at some of the risks that are faced when requirements are ambiguous.

Risk of variance between expectations

This is the variance in what is 'desired' by user and what is 'understood' by system designer and then what is 'developed' by developer.

Approximately 85% projects never get completed. The ones that do get completed do not meet the client requirements 6 out of 10 times.

This is because of the two 'Gaps' that are present in the solution development and implementation process.

First gap is created between what client wants or desires and what the system designer thinks that the client wants. This is the 'Understanding Gap'.

Second gap is between the system designer and system implementer. This is what designer thought as what client wanted and what the implementer understood based on designers communication. This is the 'Technical Gap'

The first gap results in maximum losses for projects in terms of schedule and cost. This gap needs to be closed as early as possible.

Degradation of team morale

Team morale is at stake when defects that could have been eliminated in the first place are identified at later stages. The issue here is that no one is ready to take the ownership for defect injection.

It is normal to hear that developer coded the system as per specification, but the design specifications were faulty. Many times this turns into a blame game where every one is blaming every one. This kind of environment is a sure fire way to create 'Us versus Them' mentality.

Impact on organizations' goodwill and project pipeline.

As they say in consulting industry – deals are made at the golf course. This is something that should not be taken lightly. Good things and bad things about a company are shared across senior management from different companies in an informal manner. Though these are not formal press releases, they do make an impact on the decision.

Goodwill is affected and because of that a dent is created in project pipeline as well. If a project fails when client believes that your team has done a bad job at meeting their requirements, that's negative publicity – and it travels faster than you would imagine.

Impact on cost and schedule

This is the risk that hits directly at companies bottom line. Let me explain this with a classic text book example.

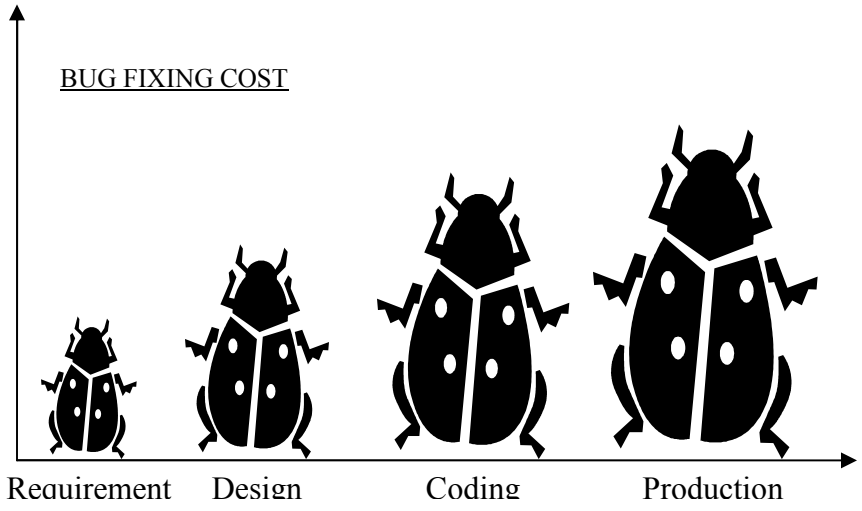

Fig 1: Bug Fixing Cost per development stage

We all have seen this diagram right from the time we picked up our first text book on testing. The later a bug is found, longer it takes to fix it as complexity increases, dependencies multiply and last but not the least – it takes more time to test the fix. Therefore, cost also increases along with all other things.

Quality Techniques that aid in reducing quality gap

80-20 Quality Probe through JAD sessions

Using this technique, the team identifies those 20% of requirements that are most critical to business and cover almost 80% of daily usage. The roots of this technique lie in the Pareto Analysis and then famous 80-20 rule used in defect prevention.

Joint Application Development (JAD) sessions are held where the client team, development team and testing team – all come together and brain storm on those 20% of requirements. The intent here is to filter out any ambiguities that may be present in requirements that would potentially be a defect candidate in later stages.

Failure Modes and Effects Analysis

FMEA is a methodology for analyzing potential reliability problems early in the development cycle where it is easier to take actions to overcome these issues, thereby enhancing reliability through design.

FMEA is used to identify potential failure modes, determine their effect on the operation of the product, and identify actions to mitigate the failures.

Requirements should be traced to failure modes. When we are conducting requirements analysis, each requirement should be studied in depth. Maximum failure modes that could impact normal operation of this requirement should be documented.

This is very helpful in understanding what would be the complexity of implementation at early stage of requirements analysis.

Functional Approach to Unit Testing

This is the most important technique that, if implemented with discipline, will improve the project delivery rates. These are some of the important points that need to be implemented.

- Developers / White Box Testers to test code 'functionally' along with 'structurally'.
- Developers / White Box Testers to first write test cases with requirements focus and then code with aim of passing all test cases.
- Unit test declared complete only when the code passes the test cases.

This technique has a lot of potential if used smartly. This is a great selling point where you can sell functional unit testing before integration testing. This provides good deal of cost and schedule savings which clients love as it gets their products early to market at less cost.

Best Practices

Project Schedule

When estimating schedule, plan efforts for testing requirements and conducting JAD sessions with stake holders. Most of the time we don't work with disciplined customer – with disciplined I mean is focused to quality work and deliverables. Customer staff often complains that they get distracted from their daily work because of calls / emails from development team.

This is a very common situation. We should try and get clients commit time of few people who would be dedicatedly working with the development team. Many times a person or a team from contracting company is posted at clients place to act as a liaison with client team. Basically, budget time and effort and get business sign off on it – either clients staff work with you or your person sits at client location is all fine so long as the requirements are understood, agreed upon and documented in a systematic and structured manner.

Write test cases before coding

This is a tricky one – how to get your developers, who are so much into coding, to write functional test cases with multiple set of data before writing a single line of code. On top of that they have to make their code pass all the test cases written and document the result.

How is all this possible? What can project managers do when faced with such situation?

Simple answer – include this as a formal step that development team has to carry out. Plan for it by scheduling time and allocating resources for testing. Negotiate for the time from senior executives – remember $10 spent now to test code at initial stage can save your team $500 later when code breaks when application is integrated and client is testing.

Communicate Goals

This is related to above point. When we tell developers to test their code – they usually understand that they need to check if their code meets specification.

If this is what the developers do, then it's not what is going to yield results. If a code that is so well written that it perfectly meets a wrongly written specification – it is still not meeting client requirement.

If requirements have been not understood correctly and have slipped JAD and requirements testing sessions, then unit testing is the second level of defense. These missed requirements can be caught here.

The team should have explicit communication stating that they need to write cases by referring the requirements and not specification.

Associated Risk

These techniques have some risks associated with it. Our success in implementing these techniques largely depends on how well we plan and mitigate these risks. Let us conclude this paper by highlighting few risks that are present.

Resistance to change

When any organization is trying to implement process improvement programs, there is a probability of stiff resistance from within the team. This type of resistance is normal and can be managed by top management using proper communication and sharing the long term vision with grass root level staff.

Setting Expectation

Functional testing of requirement and code base is a relatively new approach for most of the companies. Management should be made aware that it's a long process of experimenting and returns would be generated only after a while.

Building Competency

Developers will not over night learn how to understand requirements and test them. It's a gradual learning curve along with mindset change.

Loss of Productivity

Initially when these initiatives are undertaken, there is a short term productivity loss which is seen. This should not be mistaken for incompetence of staff – it's a learning effect.

Bibliography

1. "Using unit and functional tests in the development process" by Jeff Canna, RoleModel Software, Inc

2. "Managing workflow for customer requirements" by Pragmatic Software Inc

3. "Managing Testing Projects" by Rex Black

About Author

Aakash has over 6 years of experience in the field of quality control. He has worked with organizations like Citibank, Patni Computer Systems and Fidelity and currently is associated with Deloitte Consulting located at their Mumbai office.

He has worked on finance and finance related projects with exposure to insurance and customer service applications as well.

Contact details:

Email: avakil@deloitte.com | aakashvakil@gmail.com
Blog: http://www.sqablogs.com/aakashvakil