Attitude Towards Testing: A Key Contributor to Software Quality

S. Murugesan

CASE Research Group, Faculty of Business and Technology
University of Western Sydney, Macarthur; NSW 2560, Australia
Fax: +61-46-284 289; email: s.murugesan@uws.edu.au

Abstract

Increasing emphasis placed on high quality and customer (user) satisfaction of software calls for rethinking on the objectives and management of testing. Test and evaluation methods and tools, in themselves, do not guarantee effective testing and ensure high quality of software. The key to improving the effectiveness of testing is to improve the attitude of software developers towards testing and the nature and culture of the organisation. Also, testing has to be seen in a broader perspective of maximising 'customer satisfaction' and providing feedback for process refinement, rather than just detecting and correcting errors in the software. This paper addresses software testing from these perspectives. It highlights human factor and management issues in current software testing practices and offers suggestions for improvement.

1. INTRODUCTION

"Errors are more common, more pervasive and more troublesome in software than with other technologies."
-David L. Parnas

Quality improvement is the theme of 90's and there are emerging perspectives on what constitutes high quality of software. Software quality is more than an attribute that is normally attempted to build into software products. It is also dependent on the customer satisfaction of the software [1]. It is now generally considered as 'features and characteristics of a software product that bear upon its *ability to satisfy customer's stated or implied needs.'* It is considered as a vital requirement of software products, a business essential, a competitive necessity or a survival issue for a software industry. Strong quality focus is emerging in all the phases of software development and evolution, with increasing emphasis on product quality, process maturity and continual improvement. This trend is extended to software testing since it is a vital element in software quality assurance.

As the complexity of applications and the software increases, software testing and evaluation becomes more difficult and its effectiveness falls below expectations. Software testing is not an exact science; it is both an art and a science. But testing has often been pursued purely on technical grounds and during the past two decades there have been considerable advances in software testing techniques and methodologies, test case generation, CASE tools, etc. Test and evaluation methods, techniques and tools in themselves do not guarantee effective testing and ensure high quality of software. The key to improving the effectiveness of testing is to improve attitude of software professionals and project managers towards testing and to broaden the objectives of testing.

The foundations of software testing are (Fig.1):

- Test process, test cases and test plan.
- Techniques, methodologies, tools and standards.
- The people and the organisation

The test process, techniques and tools are significant contributors to effective and efficient testing and quality assurance. They can, however, offer better results only when they are built upon the "people foundation" and sound managerial and organisational culture. It is the people and the culture of the organisation that *determine* how any system is practiced.

0-7803-2608-3

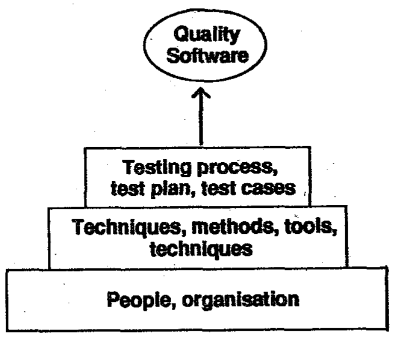
Quality Software
Testing process, test plan, test cases
Techniques, methods, tools, techniques
People, organisation

Figure 1 Foundations of Software Testing

Software testing, as generally practiced now by most developers now, suffers from attitude of the people involved and the project management towards testing, culture, myths and lack of management support. It often receives 'second rate' consideration, especially during the latter and crucial stages of the software development process. In order to improve the effectiveness of testing, organisations must make transition into higher software culture and this calls for change in attitude of software developers towards testing. Failure to change can retard growth and lead to failure of the software and its 'developer.' The overall organisational culture needs to be fostered with a constancy of purpose and continuous improvement. Further, testing needs to be focused on maximising 'customer satisfaction', rather than just detecting and correcting errors in the delivered software.

This paper addresses software testing from these perspectives, discusses current attitude of software developers towards testing and emphasise the need for the change. It identifies human factor and management issues in current software testing practices and offers suggestions for improvement.

2. SHIFT OF FOCUS OF TESTING: CUSTOMER SATISFACTION

Most software testing efforts are concerned with *minimising* customer dissatisfaction and hence the focus is on *detecting* and *correcting errors* by reviews, walkthroughs and testing. The aim of testing has been to reduce errors - which is a negative quality of software. It is, however, not good enough; 'absence of negative does not make a positive.' Besides minimising defects, testing should contribute to enhancing the positive qualities from the customers' (users') perspective. The focus of testing, therefore, has shifted to *maximising customer satisfaction* with the software product, rather than just minimising errors in the software. The first principle of software development and testing is "*know the user and the application.*"

Testing has to concentrate on critical, significant, high value software elements from the customers' perspective. This requires a greater awareness of the application environment and deeper understanding of the customer's requirements. There are three different types of requirements and they have different impact on customer satisfaction [2].

Normal (implied) requirements: These are what is typically got as formal requirements of a software product from the customer (user). These requirements satisfy (or dissatisfy) customers in proportion to their presence (or their absence) in the delivered software.

Expected requirements: These are the requirements a customer may not consider to mention them, until the software fails to meet his/her 'expectations.' Customers *assume* that these requirements (examples: on-line help, warnings, error messages and validation of input and/or output) will be met by the software. While their presence in the system meets expectations of the customers, these do not normally enhance their satisfaction. Absence of these requirements, on the contrary, is very dissatisfying to customers. Meeting expected requirements goes unnoticed by most customers, but not meeting them is disastrous for customer satisfaction.

Exciting requirements: These include features that are unexpected, or beyond customer's expectation; yet they are highly satisfying if delivered (example: additional features that are desirable but not essential). These are, of course, difficult to identify and foresee. Their presence greatly enhances customer satisfaction, but the absence does not dissatisfy, because they were not expected.

As the objective of software development to satisfy the customer's requirements, software should be tested not only *for normal (implied) requirements*, but also for *expected* and *exciting requirements*. Further, software should be tested for its robustness and test cases should include invalid inputs to check the operation of the software under invalid or erroneous conditions.

For effective testing within the schedule and resource constraints, testing should focus on those areas that are of great significance to customer (user). What is more important in testing is to ensure that there are no serious defects that are of concern to the customer in the delivered software, rather than *'zero defect'* in the software. Absolute correctness of software is not mandatory in many applications, as harmless or low risk errors can normally be tolerated by customers. Further, it is very difficult to ensure absolute correctness of a large, complex software. Consequently, testing has to move from *product-driven (or engineering-driven)* focus to *customer-driven* focus.

3. CURRENT PRACTICES AND PROBLEMS

The ills of software testing practices, attitudes and organisational culture include:

- Reduction in testing time often resulting from delay in software design and coding,
- Shortcuts in testing,
- 'Let go' -.deliver now, correct errors later - attitude,
- Lack of management support,
- Poor planning and coordination,
- Inadequate knowledge of application environment,
- Lack of user involvement,
- Poor testability - inadequate considerations on testability in software design,
- Improper staffing, and
- Poor documentation

3.1 Reduction in Test Duration

One of the recurring problems in software development is meeting the schedule. Late delivery of software has major consequences and causes incredible annoyance to the management and the customers. Time required for development and testing is often estimated poorly, grossly on the lower side, and estimates are done before the requirement phase of the life cycle [3, 4]. And design and coding usually take more time than expected or planned for. In order to catch-up with the deadline and to deliver the software product without undue delay, the project management cuts short the testing time from what was planned for. This results in shortcutting the test process and/or test cases contributing to poor quality of the software. Added to that, more problems than unanticipated normally crop up during testing requiring extra time for fixing them.

Often many project managers see delivery of software, rather than testing and quality of software as a high priority, and as a consequence testing becomes a victim. There is a tendency to deliver software product without adequate testing, leaving behind many errors (both knowingly and unknowingly) in the delivered software.

Testing is seen in practice by many software developers as a 'cushion' in the software development schedule that can be squeezed as desired. *Let-go* attitude - 'deliver now, correct errors later, if required' approach - takes precedence over quality.

3.2 Inadequate Knowledge of Application Environment

Testing team generally lacks a detailed knowledge of the application of the software being tested, its users and the environment in which it is going to work. This leads to incorrect focus on testing, often giving less importance to those areas that are of great significance to the user. Also, in the absence of application knowledge, software can not be tested for *implied requirements* and the significance or impact of errors can not be assessed.

3.3 Poor Planning and Coordination

Though planning for testing should start at the early phases of software development, often testing is not given due consideration till the later stages of the project. Often the quantum of work and time involved in testing is underestimated. Also there is lack of coordination and compatibility between test team and design teams. The customer is intentionally kept out of most part of the testing process. Due to lack of early planning, required provisions for easy and effective testing- *test instrumentation* - can not be incorporated in the design of the software, and this contributes to poor testability of the software.

3.4 Assignment of Improper Persons for Testing

Testing is looked 'down' by many software project managers and organisations. They consider design and development to be glamorous and assign skilled person to design teams and less skilled, inexperienced and less motivated persons to testing team. But the reality is that software testing is a creative and challenging activity requiring skilled, active and self-motivated personnel who care for quality. The management has to realise this and assign their human resources appropriately, and provide the required support and resources.

4. NEED FOR CULTURAL AND ATTITUDE CHANGE

The quality culture of the organisation is the first aspect that need to be appraised and improved, if required. By culture we mean the way in which quality is viewed, talked about and implemented/interpreted in the organisation. "Culture is a comprehensive term: it includes the complex values, beliefs, norms, language, attitudes, behaviors, technology. It is organisation of ideas and beliefs; it is the way people think, interact and produce goods." Quality is everybody's job, but management's responsibility.

Potential areas of improvement include:

- Management and organisational commitment and culture,
- Focus of testing,
- Better planning and effective coordination,
- Design for testability (DFT),
- Involvement of users - participative testing (PT),
- Staffing and team culture, and
- Feedback and quest for continuous improvement,

4.1 Management Commitment

Management need to have better attitude towards testing. It should:

- Realise the importance and value of testing in delivering quality software.
- Have commitment towards testing by extending its full support,
- Insist on accepting, using and delivering *only* quality products,
- Join and deal with *quality-oriented* organisations, and
- Break down barriers between departments.

There is a general fear that management is likely to victimise software development personnel for defects found in the software developed by them. In view of this many software errors are not documented or reported within the organisation. Similar errors continue to occur in later projects. Management should drive out this fear so that everyone in the team/organisation works effectively by discussing the problem they had encountered and how their occurrence could be prevented in the future. Such an attitude and 'professional atmosphere' can greatly avoid errors and 'reinventing solutions' again.

Staffing: Testing is a challenging, creative team work and this makes it essential that everyone in the team contributes towards the success of testing. Assignment of right people for testing has great influence on the success of testing. Testing requires persons who are technically competent, active and experienced in testing ***and*** design. The team leader should have, among others, problem solving and leadership skills and ability to manage a team and coordinate with customers.

4.2 Planning for Testing

Software testing involves considerable planning, effort and time. Planning for testing should start very early in a project - at the requirement definition and design stages itself. Detailed planning for testing can reveal many errors at the very early stages and several studies have shown that many errors detected during the testing phase were originated in the early phases. Planning for testability improves the design itself! Also better design and development practices substantially reduce serious problems before testing begins. This translates to decrease in test and debug time and minimal reworks. Software testing should start early in the system development process and cover the whole the life cycle of the software. Agreement on test plan early in the requirement phase have significant impact on effectiveness, success and cost of testing.

Design for Testability (DFT): Testing is affected by design decisions. Software should be designed for ease of testing, and detection and location of defects byenhancing the observability and controllability of software, by providing additional access to specific software segments.

4.3 Involvement and Participation of Users

Significant benefits can be gained by involving the user (customer) in the testing process. User should be encouraged to play active roles in testing, as success of a product also depends on how the customer perceives its uses and his/her confidence in the software. Central notion of user participation is "the right of people to have a direct influence on matters that concern them.' The concepts of joint application design (JAD) [4] and the participative design (PD) [5, 6] were born out of these needs and are getting greater acceptance in software development and in other areas. They facilitate active, dynamic, interaction between users and developers. Involvement of users in testing can range in a continuum of roles from consultative and representative to active participation in testing throughout the life cycle of the software. The developers need to attract the interest of users in testing and encourage their participation in test planning, system testing and acceptance testing.

4.4 Process Improvement

Inspect the code to find out not only the defects in the code, but also the defects in the software development process. A byproduct of testing is valuable information about the types and number of defects found during testing. Document this information and provide feedback to the design team, to identify the root cause and to eliminate them. If you eliminate the root causes, the origin of the defects, then you eliminate defects *for ever*. As you, 'take away the causes and the effect ceases.'

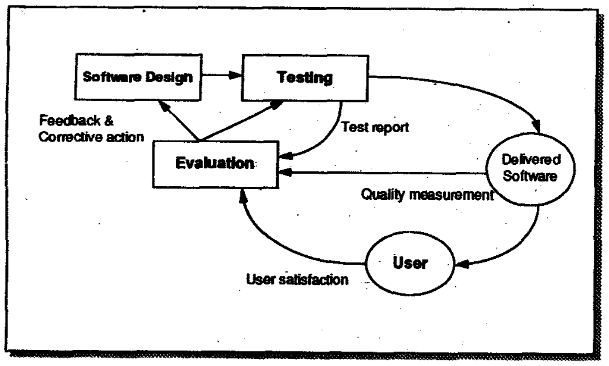

Figure 2 Evaluation for continual process improvement

Another fundamental requirement is the continuos improvement in the testing and software development process. Improvement is accomplished by evaluation, effective feedback and appropriate correction (Fig. 2). Effectiveness of testing is evaluated based on number of errors surfaced during testing and during its actual use, severity of errors - ranging from minor to major-, the origin of errors and complaints from users. Also user satisfaction-level survey would be very useful. The information collected is to be analysed, summarized and documented, and defects in the design and test process need to be corrected on a continual basis.

5. CONCLUSION

Software testing techniques, methodologies, tools and standards can only aid in testing, but it is the management and the people involved who have to plan for and carry out effective testing. Testing need to focus on maximising 'customer satisfaction', rather than just detecting and correcting errors in delivered software.

Cultural and attitude change, management realisation of the value of testing and its commitment to testing are the key contributors to enhancing effectiveness of testing in ensuring quality of software. We addressed some of the issues in software testing from these perspectives and discussed current practices and attitude towards testing, emphasising the need for rethinking about testing. We identified attitude and human factor issues in current software testing practices and offered suggestions for improvement. In order to enhance the effectiveness of testing and to improve the software quality, software houses must make transitions to higher software culture.

"Knowing is not enough; we must apply.
Willing is not enough; we must do."
- Gothe, German Philosopher

References

1. P.J. Denning, *Communication of the ACM*, Jan 1992.
2. R.E. Zultner, TQM for technical teams, *Communications of the ACM*, Vol. 36, October 1993, pp 79 - 91.
3. R.L. Glass, "The software crisis ... not?" *Computer*, April, 1994, p104.
4. A.L. Lederer, "Information system cost estimating: A management perspective," *MIS Quarterly*, June 1990.
5. D. Schuler and A. Namioka, *Participatory Design: Principles and Practices*, Erlbaum, NJ, 1993.
6. Special issue on Participatory Design, *Communications of the ACM*, June 1993.
7. J.H. August, *Joint Application design: The group session approach to system design*, Yourdon Press, Englewood Cliffs, NJ 1991.
