International Journal on
Advanced Science
Engineering
Information Technology

Vol. 2 (2012) No. 5
ISSN: 2088-5334

Assessing Right Amount of Quality Assurance (QA) for Software Products. "A Quality Assurances for Developing Software Projects"

Murtaza Hussain Shaikh[*], Mir Sajjad Hussain Talpur[#]

[*] *Department of Computer and Information Sciences,*
Norwegian University of Science and Technology (NTNU), Trondheim -7448, Norway.
E-mail:Murtazahussainshaikh@gmail.com

[#] *School of Information Science and Engineering,*
Central South University, Changsha, Hunan Province, 410083, P.R. China.
E-mail: Mirsajjadhussain@gmail.com

***Abstract*— Quality Assurance (QA) is an important aspect of product development in any industry, not least software development. To secure an end-product that is as high a quality as possible, thus satisfying the customers, Quality Assurance is essential. A software application released with several so-called *"bugs"* and other flaws is obviously a product which has passed through a poor Quality Assurance process. Thus, it is important to have a proper, systematic program to follow during the developments, which ascertain the final quality of the product. Too much QA however, can lead to developers focusing too much on analyzing and documenting every part of the development, ending up with an overload of documentation. This would slow down the development progress, and in the worst case, kill of the project. There are two main aspects of this paper; first problems of *"inappropriate amount of Software Quality Assurance"* and second is *"how we can balance between creativity and quality"*? However, we will briefly visit other industries to shed light on the importance of QA as a whole.**

Keywords—Maintainability; Performance; Budget; Standardrization; Assurancel; Verfication; Development; SystemCycle; Stakeholders; Quality.

I. Introduction

A natural starting point to our discussion is to assess what quality and quality assurance in software development involves. It is not easy to define what exactly constitutes a quality product. The ISO (International Organization for Standardization) definition states that quality should consist of *"the totality of characteristics of an entity that bear on its ability to satisfy stated and implied need"* [13]. While in [4] it states that a quality product should *"conform to requirements"*. As for software quality, we can also incorporate these definitions to constitute a quality software product, where requirements are the main concept. A software development project typically starts out with several requirements received from the customer, which is further refined into a complete requirement specification. The final product should naturally conform to these. Still, it is important to state that software quality involves more than just the conforming to requirements, and one should take different viewpoints to decide what represents a quality product. In addition to both functional and non-functional requirements of the product, the developers need to conform to the requirements of all the actors in a project such as the stakeholders, suppliers and so forth. A quality product should also be cost-effective and competitive in comparison to rivalling products [3]. As for quality assurance (QA), one can define it as a process which focuses on monitoring and evaluating the various aspects of a product development project [8]. These activities aims to ensure the manufacturing of a product of sufficient quality, obviously satisfying customers as well as managers involved in the project. More specifically, software quality assurance (SQA) focuses on the same aspects in a software development process. This will usually include the continuous monitoring and evaluation of the various aspects of a software development process such as the requirement specifications, software design, testing and the actual implementation of code [9]. SQA means to essentially find faults in these phases of the process that lead to low quality, thus finding the source of the problem and then dealing with it [2]. Ensuring the quality of these aspects should ensure the appropriate quality setting surrounding the project.

II. Quality Assurance in Planning Process

The key essence of defining SQA is that it is based on systematic planning and implementation. This involves all the various actions and stages during a software development process. In [2] it offers an even broader definition of SQA, implying that it should involve service subsequent to the product's release. By this, one could interpret that the continuous patching of a software product can be seen as a stage in the quality assurance process. In addition, it mentions that the SQA should not be limited to just the technical actions of a project, but also the other aspects such as budgeting and scheduling [2].

In the history of software development, several applications have been released consisting of several flaws, lowering the overall quality of the product and thus diminish the reputation of the development company. Due to such incidents, quality assurance has become an increasingly important aspect of software development. In software development, quality assurance come across several specific complications compared to QA in other industries. A software application can be seen as an "invisible" product, meaning that it is not a physical product you can inspect. For example in quality assurance in car manufacturing, flaws or errors in the product can be detected by visible observation on the physical objects being manufactured. However when it comes to software development, the "building blocks" in the product are code, which is obviously not physical objects [1]. Thus it can be noted that the complications of securing the desired quality in a software product exceeds that of developing products in many other industries. Due to the invisible nature of software code, it is hard to spot errors and flaws in the software while engineering it, without proper quality assurance techniques. And even when the final product is released, errors can still be present, but not yet discovered by the developers. These errors can in the worst case be discovered by the customers when the software is utilized. Obviously it is a need for continuous and comprehensive testing throughout the development process, and this is an integral part of software engineering.

III. Quality Assurance in Development Phase

The various testing activities is naturally an important part in assuring the quality of the end-product, however the process of quality assurance encompasses even more than just the documentation of the testing and implementation of software code. As mentioned earlier, SQA is responsible for monitoring and evaluating the development process as a whole. Software quality assurance is a systematic pattern over all the actions needed for securing the quality and assuring that all the requirements are met. It is a process used to measure all the activities performed in a development life-cycle, accomplished through quantifying the quality of the product and the activities involved in the development. These various activities include the initial analysing phases in a project, the software design, as well as of course the implementation phase. Each of these stages needs to be validated and put under verification. SQA differs from quality control in that it is, as mentioned, a continuous process which aims to guaranteeing the quality of various activities performed during the development [1]. Another aspect to software development we should consider in this essay is the creative freedom of developers, and how too much QA can interfere with this freedom. QA may in some cases encourage the developers to follow a strict pattern or plan in what and how they develop. Even if they follow an agile development process, the QA process might assure that the developers follow a strict pattern in how they should solve the problems that occur. This could occur on either the coding-level or at the project management level. If a developer discovers a creative solution to a problem the developer might be unable to utilize his idea, because it would conflict with the various processes and activities which are involved in the project. In addition, it also requires a larger budget for the project to maintain an elaborate QA process. In a small development team, maintaining such a process might take too much of the focus away from the actual development, thus slowing down the progress [6]. Even in a larger development team, dividing too much effort on securing proper QA might seem counter-effective, and a waste of the project budget. The trick is then, of course, to strike the right balance between "the right amount" of QA, as well as keeping up the creative freedom of the team, and thus the progress of the project.

IV. Balancing the Quality in Software

We talked about how to define quality and quality assurance in software development. In this section we will talk about how to strike the right balance between too much, and too little usage of SQA, that is – the right amount of SQA. We will base this on what the consequences might be of inappropriate use of SQA. Software quality assurance involves all phases of software development, from the planning phase, the design phase, the implementation and testing phase, as well as the maintenance phase – post delivery. Since the cost of rectifying errors grows by about ten times with each stage of development, it is sensible to incorporate SQA at every step of the system development life cycle [6]. The way in which lack of proper QA can affect the software project can be specified from phase to phase. Lack of proper QA during the designing phase, the usability of the system might decrease and the user will have trouble communicating with your system. During implementation phase, bad structured code might lead to errors in part of the code which is not easily fixable, due to the lack of standardization of the code. For instance it might lack comments or documentation for complicated parts of the code, describing the functionality. A consequence of this might that you will spend lots of time and money to detect each error and fixing it. In standardized code however, it is easier to locate the error and find the problem in a shorter time-frame. It is not easy to add more features or renew messy code, for that reason one should implement every section in standardized code. As for testing, it should be important to test often and early in the development. If the testing only occurs late in the software development life-cycle, this may lead to greater costs to find and fix defects in the code [1]. Another aspect to consider during the QA of a project is security. Lack of QA during the security planning of software may of course lead to security problems. Thus, a hacker may attack your system and rub your important information of your system. When a software development project adheres to right amount of standards at its implementation phase, it can have positive effects;

- Programmers can go into any code and figures out what's going on, so maintainability, readability and re-usability are increased. Code walk-throughs become less painful [11].
- People can get up to speed more easily [11].
- People new to language are spared the need to develop a personal style and defend it to death [11].
- People new to a language are spared making the same mistakes over and over again, so reliability is increased [11].
- People make fewer mistakes in consistent environments [11].
- Idiosyncratic styles and college-learned behaviours are replaced with an emphasis on business concern-high productivity, maintainability, shared authorship, etc [11].

Since a very large portion of project scope is after delivery maintenance or enhancement, coding standards reduce the cost of a project by easing the learning or re-learning task when code needs to be addressed by people other than the author, or by the author after a long absence[10]. Coding standards help ensure that the author need not be present for the maintenance and enhancement phase.

V. Discussion

We will go more in-depth and discuss the effects of lack of QA in software development. It has become clear that too little emphasis on QA can lead to a final product of insufficient quality, which is a real danger in many software development projects. An example is the situation mentioned in [2]. In this example the author explains that the opening of the Denver International Airport USA was delayed for 16 months due to failure in the software of the baggage-system, which led to an enormous economic loss [1, 2]. This is a form of QA that allowed for errors, but still maintained a high enough quality for the product to be a commercial success. As in this case, if the product consisted of over 500 bugs, could one call it a "quality product"? Different definitions of quality would give different answers to this question. However, since the product becomes a commercial success, it should at least be considered as to be of "sufficient enough" quality, and thus of sufficient amount of QA.

The use of a QA process should improve the acceptability of the product. But what can happen if we use too much quality assurance in software engineering? The first impression of using too much quality assurance in different projects, is its effects on peoples freedom and strict them to do at a particular context [1]. We can find many examples where enterprises unintentionally reduced or even killed creativity and innovation for the sake of control, performance, and cost reduction. At the one hand if you use too much quality it makes lots of bureaucratic and on the other hand paying too attention to the creativity rather than standardization gives lots of freedoms to the projects, and lots of freedom may will cause chaos. Considering creativity without quality will make your structure crucial, there is no standardization role on your structure and everything without a strong and determined structure will fall down. Creative tasks are inherently connected to high variance of possible outcomes, which is due to the fact that being creative means to be original and come up with novel ideas and solutions. Using creativity at every projects means a high demand of flexibility, This may lead to unwanted consequences, such as losing control of process (losing control of time and budget), low product quality (which may lead to customer dissatisfaction), and lack of external compliance (which can lead to a loss of reputation or even to lawsuits) [13]. We need a structure to manage creativity without sacrificing creativity. As we can see on the figure below, there are lots of activities that should be done at during the project life cycle to keep balancing between quality and creativity. It is obvious that we need to allow creative freedom at our project, because without creativity and innovation no progress is possible, but at the same time, try to enhance creativity by calculated management of the overall phases of our project, like considering aspects such as performance, cost, or risk.

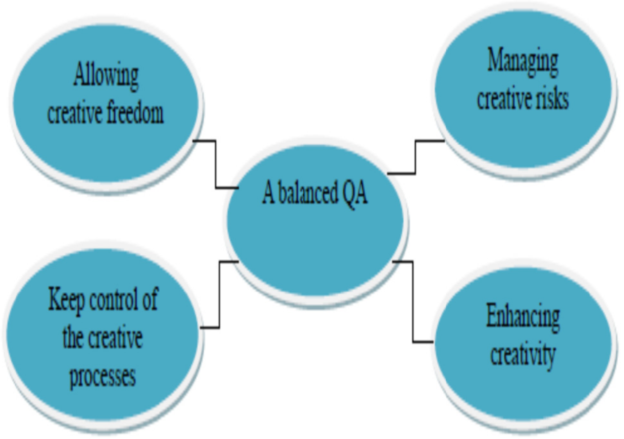
Allowing creative freedom
Managing creative risks
A balanced QA
Keep control of the creative processes
Enhancing creativity

Fig. 1 A Balanced Quality Assurance in a Software product

Quality assurance improves the acceptability of the product. It can be useful for developing the long term strategies for ensuring performance in quality assurance both internally and externally [6]. The principles in defining strategy can be useful for consistent, clear and easy to understand plans [6]. However, sometimes the process may hinder the flow of work. The quality of the product can be assessed by auditing at regular periods. The auditing serves as a process to measure the quality of product [6]. In order to make this effective, the auditors should be trained and should have good overview on project. This can be achieved by either internal training this includes quality measures and policies of company or any external training. The knowledge and experience of the auditor also plays an important role in the quality maintenance [9]. The auditing process is based mainly upon the documents and company standards, so for effective quality assurance it needs documentation related to process involved in product development. It was observed that auditing process sometimes produced reports with no findings at Taylor technology [3]. So they came up with an approach called as "QA signature". QA assurance is a process of documenting QA review has been completed [1]. So this approach removes the needless auditing to be done where there are no findings. The quality assurance becomes critical in agile processes as the development is done iteratively. QA is same for agile projects as others with respect to deliver the product but things will change with agile development as requirements change continuously [13]. Sometimes testing can become part of quality assurance of the product. The most common test strategy adopted in agile settings is TDD (Test Driven Development). So, this ensures that the product developed is quality assured and tested [5].

VI. Conclusion

This paper states the importance of having the right balance of quality assurances in software-development. Also in specific; how we can balance between the quality in on hand, and creativity at the other hand. What we concluded is that too much quality assurance can be waste time and also money, yet not enough will impact the effort and schedule adversely because defects will get through. In the discussion part, we highlighted about the creativity and quality. The important thing that using creativity in each project needs lots of control and management on different parts like risk management and cost management. The main observation behind this paper is that the development process can become dangerous if there is either too much, or too little focus on QA. The case studies with respect to various firms like Taylor technology suggests that if the QA is not up to mark then product is not accepted and if it is too much then the product is delayed [7]. The challenge faced by many companies these days is to provide right amount of QA. The quality assurance process becomes critical in places where development is done iteratively like agile development. The quality assurance also depends on the product being developed [7, 1]. If the product needs too much focus on quality then quality principles can be strictly implemented otherwise it can be relaxed for smooth development of the product. Another aspect that is an important factor in deciding the amount of QA a project should utilize, is the size of the project. By size we mean both in terms of the number of people participating in the project, and of course the project budget [10]. Naturally, often these factors are closely linked. A large project with a substantial budget should have a clear and elaborate focus on the QA process. This is of course due to the impact poor QA can have on the project, which increases proportionally with the project size [5]. The main point however, is that the focus on QA should not upstage the development process, because this could lead to stalling of the progress, and thus leads to loss of profit in this way. On the same point, it is also important to not let the QA process interfere with the creative freedom of the developers actually producing the code. Smart solutions to problems occurring, should not be stopped by maintaining a QA process [12]. By this we can state that a good QA process should reflect the project in terms of project size, project budget and not interfere with the creative minds of the developers. Such usage of QA should in turn lead to a product of good quality, thus satisfying all the different actors involved in a software development project.

Acknowledgment

We thankfully acknowledge the helpful comments from the reviewers, which have improved the paper very significantly. Finally, we are always thankful to our beloved parents for their love, trust and support.

References

[1] Kevitt Mark (2010)" Best Software Test & Quality Assurance practices in the project life-cycle. An approach to the creation of a process for improved test & quality assurance practices in the project life-cycle of an SME". Master of Science thesis, Dublin City University. DOI: 15089. Ireland.

[2] Daniel Galin (2009)"Software Quality Assurance: From Theory to Implementation". ISO 9000-3. American Society for Quality. USA.

[3] Bijay K. Jayaswal and Peter C. Patton (2006)" Design for trustworthy Software" .pp. 213. ISBN 9780132797351. Prentice Hall PTR USA.

[4] Paulk, Mark C., "A Comparison of ISO 9001 and the Capability Maturity Model for Software", Software Engineering Institute, CMU/SEI-94-TR-12, July 1994, pp. 19.

[5] Carmen Zannier, Mike Chiasson, Frank Maurer (2007)" A model of design decision making based on empirical results of interviews with software designers". Information and Software Technology archive Volume 49 Issue 6, June, 2007. , MA, USA.

[6] M. E. Kabay (2010)" Software development and quality assurance. An essential component of security". Network World, Inc. Available online at http://business.highbeam.com/409220/article-1G1-220549761/software-development-and-quality-assurance-essential. [Accessed on: April 15th 2011]

[7] Stefan Seidel, Michael Rosemann (2008)" Creativity Management – The New Challenge for BPM". BP Trends , May 2008. Australia.

[8] Bach, James (1995) "The challenge of 'good enough' software." American Programmer. USA. Available online at: http://www.satisfice.com/articles/gooden2.pdf [Accessed on 26th, February 2011].

[9] Dave Nielsen (2010)" CMM and Project Quality Management". Available online at: http://www.pmhut.com/cmm-and-project-quality-management [Accessed on 26th, February,2011]

[10] O'Hanlon, T., (2011) "Quality Auditing for ISO 9001:2000: Making Compliance Value-Added". American Society for Quality.USA.

[11] Watts S. Humphrey (1989) "Managing the Software Process". SEI Series in Software Engineering. Addison-Wesley, USA.

[12] Hoyle, D (2011)" ISO 9000, Quality Systems Handbook". 4th Edition, Butterworth-Heineman, Woburn, MA. USA.

[13] International Organization for Standardization, Guidance on the Process Approach to quality management systems, ISO/TC 176/SC 2/N544, December 2000.