Abstract

Quality needs to be designed into a product. The product design is where quality starts. It also needs to be verified at each step in the development process. This added burden has always discouraged project development teams. By studying this issue, we have discovered that like software development, software quality assurance needs to be both comprehensive and agile at the same time. When quality is checked more often customer satisfaction is increased.

1. Introduction

The CS615-616 Software Engineering Seminar is the capstone course for Pace University’s Masters in Computer Science curriculum. The focus of which is to develop soft skills as well as introducing the student to formal software engineering procedures. To develop these real world skills, the students are broken up into teams. Each team is assigned a project which must be delivered on time to a set of customers. The customers must accept the provided solution for the corresponding team to achieve full marks. Our team was assigned the duty of becoming the Quality Assurance team. Our goal was to assist each team in delivering a high quality project, through reviews, testing and the establishment of best practices.

The 2003/4 QA team [5] clearly defined Software Quality:

“The issue of software quality has been debated as long as software has been written for one person to be used by someone else. As software moved away from custom made, machine specific code to off the shelf systems that may run on multiple operating systems with components from multiple hardware manufacturers while having to coexist with multiple other running programs in multi-tasked environments, the issue of quality has become more and more important while becoming more and more elusive. In the current software world the task of making sure that a given software product is capable of running on the number of platforms that it is likely to encounter seems overwhelming. Especially in Web-enabled applications, where the developer has limited if any control over the environment their application will be running in, it seems impossible to produce a truly defect free system that meets the other desirable aspects of quality. Adding to the burden of producing defect free high quality software is the trend towards using reusable software components. If a developer were to develop an otherwise perfect application utilizing an off the shelf database management system that contained numerous flaws, the overall level of quality of their product would suffer greatly.”

Our approach was to first study the results of former teams, using their lessons learned as a guide. We then searched for an approach that could be applied so that the current teams would avoid the same problems that former teams experienced. Our research had suggested that if certain quality attributes were applied early in the development process, each team would be able to develop a more complete design that would lead to a predictable implementation. We communicated the following Quality attributes during the first semester and then observed the progress of each team, tested completed sections of each project and provided feedback to each team. The result of this work is presented in this paper, it serves as a requirements document for the next 615/616 class, and provides a set of documents that the next QA team can build upon.
2. Relevance

The key part of our approach was to use the lessons learned by former Quality Assurance teams. We then found the following approaches have been used successfully to address these issues:

The Serco consulting group states [1]

“The chosen software development lifecycle will determine the points at which testing or review will be performed, but both activities form an integral part of all phases of the development process. It is widely accepted that a "test-early, test-often" approach can greatly assist in the production of quality software. In addition, the design of software can impact on the ease of testing so testing must be in borne in mind from project conception through to completion.”

The paper Introduction to Testing Object-Oriented Software [2] states:

“Under our approach, testing is not an afterthought. Testing is a process separate from the development process, but intimately related to it. We have a motto: Test early. Test often. Test enough. We favor the following iterative development process:

- Analyze a little.
- Design a little.
- Code a little.
- Test what you can.

The paper Building Usable Software Through Early Testing in the [3] offers us this insight:

“Software suffers from the "purity of goals" phenomenon. Current and experimental software development processes are making progress in the area of requirements conflict identification leading to better tests and better software products. Bender believes that more than half of the defects in a system are a result of poor requirements. By examining specifications and requirements, "purity of goals" can be identified and addressed. This early test activity can help mold a software products specifications and requirements so they are complete, accurate, consistent, unambiguous, and testable before the first design or line of code is written. It also promotes improved product quality since quality can truly be built in as the product is developed.

Given the dynamics of specifications and requirements, testing should be viewed as a integral and ongoing component of the software development process. Otherwise, poorly and ineffective tests will be developed. Throughout the software development process, a goal of the test team is to ensure that specifications and requirements remain clear, concise and accurate so effective tests can be developed which can demonstrate the product's compliance.”

The theme we see in agile development and other modern software development mythologies is to test early, test often, test enough. This is why we have chosen to focus on assisting all teams in accomplishing those goals.

3. Methodology

To accomplish our goals we will focus on each projects "quality attributes".

Best Practice Guidelines: To ensure that all projects meet minimum standards of Correctness, Maintainability, Reliability, Usability best practices guidelines for coding, design, testing is provided in this document.

Test for Correctness: All projects will be tested to ensure that they conform to the specifications and expectations which have been laid out during analysis and by the customer. The principle means of doing this will be to develop and execute use cases from the customer requirements. The use cases will be tested with several non-exception scenarios to ensure that the workflow is correct. The use case will also be tested with several potential exceptions along the way (such as entering characters into a field that should be numeric only). Any problems will be reported to the implementation team for rework.

Assuring Maintainability: It is an important quality of software that measures the extent to which the software can be modified at the lowest possible cost. Maintainability can reduce costs for both developers
and customers. All documentation and code will be reviewed to ensure that the team follows best practice guidelines, implements modularity and provides for easy understanding to future software teams.

Evaluating usability: This includes understandability, learn ability, operability, attractiveness and compliance to requirements. The quality team will send out customer satisfaction questionnaires to ensure that the software increments and final deliverables are acceptable to the customers, are efficient to use and learn and handles errors. Usability feedback will be given during reviews and usability testing will be conducted during system test.

Reliability: Reliability is the ability of the software to perform correctly over time. It measures the frequency of failures, as encountered by testers and end-users. Testing and rework should ensure that if failures occur, the system can handle them or can recover easily without compromising data or the system it exists in. The quality team will conduct reviews and provide feedback to ensure that teams provide for reliability.

Security: The teams should aim to protect confidentiality, integrity and privacy in the systems they develop. These systems should be capable of preventing attempts by unauthorized users to access the system without permission. At the same time, the system should not compromise other systems that may exist in its environment. The security features will be tested and feedback will be provided to the implementation teams during the course of quality review.

4. Results

The lessons learned from past QA teams [4] [5] (2002/3 and 2003/4) were:

- Most teams struggled with technology that they were unfamiliar with. That resulted in major changes to requirements and their design.
- Many past projects suffered some Scope Creep. When requirements were not managed, the scope of the project grew. As scope grows it becomes increasing harder to deliver the product.
- When mile stones were not met, and there was schedule slippage, if not addressed quickly the team did not complete all of their design objectives.
- Testing that was done late or not often resulted in less robust products.

Case Studies:

To understand the effect of Quality Assurance, a couple of the projects were studied. These case studies are from the perspective and understanding of the QA team and not that of the implementation teams, nor the customer.

Case Study: Team 2 MIND PAL Telemedicine System

This project involves a good amount of integration between the hardware and the software components. The team communicated effectively with their client during both semesters and tried to manage their requirements well and keep it within scope. Since they had to deal with the working of handhelds, cameras and GPS systems, they encountered several issues while getting them all to work.

They successfully developed a small prototype by the end of the first semester and got a good feedback from their clients.

Over the course of the second semester they improved their server side interface and completed all their server side functionalities successfully. They also managed to develop their PDA user interface and integrate it with server side functionalities. They came up with workarounds when issues with PDA development seemed to create schedule slippage. At the end of the semester they were able to successfully deliver most of their requirements except for the GPS functionalities which involved integration with hardware and the time constraints limited their further research. They successfully delivered their system to their clients and also provided some training. Availability of clients for further training is a concern. To summarize Team 2 kept their targets in mind and tried hard to keep to schedule and developed the...
software trying to achieve quality targets like usability, security, reliability and maintainability. They have provided a basic system which future CS students can build upon and add more functionality if the need arises.

Case Study: Team 5 GCA High School Website

This team started out with three members, but ended up with only one. This did not impact the quality because early on time was spent up front working on the design with the customer. Websites can grow very complex very quickly. This team decided early on to only work with technologies that they understood. Through the design phase they effectively managed their customers expectations, not allowing the project to grow to a point that they could not handle its implementation.

The development phase went extremely fast. This teams approach is a good example of how keeping the design simple and not having any dependences on unfamiliar technologies pays off. Buy using PHP and mySQL they were able to deliver a prototype on time. This allowed for irrigative development taking into account their customers feedback. As we had predicted, because the customer saw the work in progress, they felt part of the whole project and were satisfied early on. Expectations thus were managed in this fashion.

There was one issue that did arise. The focus of this project was to deliver exactly to the customer requirements. Perhaps more time should have been taken to elaborate on the requirements, to develop a more complete design. This was seen during the testing phase. We observed that the web site was developed with only Microsoft Internet Explorer in mind, other browsers did not display the page correctly. Usability was only addressed from the customers specification, other groups, like handicapped users should have been included and addressed in the sites design. Although outside the scope of testing, security was only addressed through PHP’s built-in methods. To be more complete a full section on security should have been included in the design.

5. Conclusions

When Quality Attributes are properly applied it will result in a more complete product.

As the product nears completion, customer satisfaction should grow. That is with each incremental delivery the customer should use the product more, relying on what is delivered. If the customer is satisfied with each delivery then the product is nearing completion. But if the customer does not feel that the requirements are being met, or is growing increasing frustrated, then the product will suffer greater and greater delays until the key issues are resolved.

The lessons learned were:

- Technology still remains the biggest issue that teams struggle with. As the current trend in industry to favor more pervasive computing devices, we think that this issue will still be a stumbling block to teams that have the added burden of developing with such devices.
- To communicate more effectively that Quality Assurance is not added work. That there is an upfront cost to QA, but the later dividends make it a wise investment.
- One of the biggest lessons learned is that increased communications can yield more efficient work. As when two programmers work side by side, two testers can produce better results in a shorter amount of time.
- That testers who themselves are developers, when focused on testing, given the underlying code and set to task of uncovering problems will produce the best results. Not only will they find the most issues, but they will also be able to describe the issue in terms of the underlying source code. This in turn allows the developers to resolve the issue by not just patching the code, but frees the developer to evaluate design and or architecture changes.
- Testing of completed parts of a project by outside individuals is not efficient for a agile development process. Quality needs to be addressed with each iteration, before parts are completed.
- We also learned that incomplete designs that did not address technical issues or were written so that there scope was narrowly focused on only the customer requirements, were more inflexible
and the problems that those projects faced tended to be harder to solve.

6. Recommendations

These are the recommendations to the next QA team for CS615/616:

- Decentralize Quality Assurance, make one member of each team the QA focal point. This position can be rotated to different members of the each team. This way quality can be addressed early enough and constantly through out the project’s life.
- Start as early as possible with the QA part of new projects, reusing or reworking the existing documents from previous QA Teams.
- Have each team set due dates and make them responsible to meet these dates. This is already true for each projects major deliverables. Expanding it to include smaller parts will allow for each project to be tracked better.
- Teams need to have a fully developed project design at the end of cs615. They should have worked out all technical issues. Also projects should not over focus on the customer requirements. Flexible systems will be able to address future change, which results in higher customer satisfaction.
- Make each project’s documents more constantly available. Projects that had easy to access documents that were also up to date, had the best inter-team communication.

7. Summary

This paper was the result of the combined efforts of the QA team for CS615/616. We provided guidelines and constructive feedback to each team. We also took feedback from each team and used it to improve our process. By conducting system/interrogation testing we helped to uncover new issues, and also kept old ones from being added back in, before the customers version.

There is still work to be done. There is opportunity to streamline many QA processes. Communication between the teams and also between team members can be improved. As in other studies, the more frequent and thorough unit testing combined with a frequent and aggressive system/interrogation testing will result in a higher quality product that will have the greatest chance of delivering on time.

8. References


