Group Discussion Report 2
Software Quality Process Implementation Activities

Group Member's Names:
1. Akshay Kalsotra
2. Malika Hafiza Pasha
3. Mustafa Quraishi
4. Sajeel Mohammed Abdul
5. Shoaibuddin Mohammed

Part 1

Due to time and budget constraints, a project leader has decided to introduce "an economy plan" that limits the quality assurance activities to a standard design review – *as required by the contract with the customer (50% filter), and comprehensive system tests (60% filter)*. Considering the model's contribution to defect-removal efficiency and costs:

1. **Find the expected savings, if any, in resources invested for defect removal during the development process compared to the standard quality assurance plan.**

Answer:

Understanding Defect Removal Efficiency and Costs

Defect Removal Efficiency (DRE) is a metric used to assess how well QA procedures find and correct errors. It is the proportion of flaws discovered and corrected during a quality assurance phase to the total number of defects existing prior to the start of that phase. Greater DRE is a sign of more productive QA efforts.

Depending on the stage of the development process at which the problem is found and fixed, there are differences in **cost for defect elimination**. Defective fix costs typically rise as a project moves through its lifespan. The cost of fixing defects discovered later (during system testing or after release, for example) is higher than that of fixing defects discovered early (during design or coding). This is because later-stage defects can have compounding effects, including the need to redo completed work, negative effects on user satisfaction, and potential damage to the organization's reputation.

The "Economy Plan" vs. Standard QA Plan

The "economy plan" restricts quality assurance efforts to thorough system tests (60 percent filter) and a standard design review (50 percent filter). This strategy concentrates on a small number of key quality assurance tasks in order to lower upfront QA expenses. Nevertheless, this constraint also suggests a possible rise in the likelihood of undetected flaws advancing throughout the development process, culminating in increased expenses subsequently.
In comparison, a normal quality assurance plan would usually encompass a broader range of quality assurance tasks, such as design reviews, system testing, acceptance testing, unit testing, integration

testing, and code reviews. With a higher overall DRE from this all-encompassing approach, fewer problems should make it to later stages or post-release.

Calculating Expected Savings

The anticipated cost reductions from implementing a "economy plan" would ideally result from a lower initial investment in quality assurance tasks. To properly evaluate these savings, however, one must take into account both the prospective rise in expenses related to faults that go undetected because of the narrower scope of QA activities, as well as the immediate decrease in QA expenditure.

a. **Immediate Savings:**

lower costs for QA employees, equipment, and procedures as a result of fewer scheduled QA tasks.

b. **Deferred Costs:**

higher repair costs for flaws found after the product is released or later in the development cycle, which could even surpass the initial savings.

Detailed Example Calculation

Our simplified computation revealed that the "economy plan" resulted in greater overall expenses even if the upfront investment in QA activities was lower. This paradox occurs because, as a result of the plan's lower overall DRE, the cost savings from fewer early QA operations are more than offset by the increased costs of fixing the defects that were not discovered early.
This analysis shows that, although restricting QA efforts may seem like a good idea at first, the long-term effects of such choices might result in much higher costs because of the greater costs related to later-stage fault correction. It emphasizes how crucial it is to establish QA strategies taking into account the complete lifetime costs of errors rather than only concentrating on short-term cost savings.

2. **Find the expected effects of the “economy plan” on customer satisfaction. Support your answer with a quantitative comparison to the standard plan.**

Answer:

Effects of Flaws on Client Contentment

The perception of the product's dependability and quality is directly related to customer satisfaction. Visible flaws can irritate customers, erode their faith in the company, and eventually increase the risk of them switching to a rival's product. Customer happiness is greatly impacted by these flaws, depending in large part on their severity and visibility:

a. Small Flaws:
Although they may not have a big effect on functionality, these can lower the product's perceived quality.

b. Major Defects:
These might interfere with essential functioning, which can directly cause customer unhappiness and possibly have an impact on their finances or operations.

The "Economy Plan" and Its Straightforward Consequences

Because the "economy plan" restricts quality assurance efforts to thorough system testing and standard design reviews, there is an inherent risk that faults will find their way into the finished product:

a. Increased Defect Rate:
Compared to a more thorough QA plan, a product with a lower overall defect removal effectiveness (DRE) is likely to have more flaws when it is released. User pleasure and experience may be directly impacted by this.

b. Post-Release Problems:
If a defect is not discovered before release, it is possible that patches or updates will be needed afterward. This may cause users to experience inconvenience and negatively affect their opinion of the product's quality.

Comparing Quantitatively to the Standard Plan

With its wider range of QA operations, a conventional QA plan would probably produce a higher DRE and fewer errors in the finished product. One can deduce the quantitative effect on customer satisfaction by:

a. Defect Rates at Release:
Higher initial customer satisfaction would be directly correlated with a lower defect rate at release (due to a more thorough QA procedure).

b. Post-Release Support Requirement:
A product with fewer flaws usually has a lower post-release support requirement, which improves the customer experience overall.

Evaluating the Effect on Customer Contentment

Although it is difficult to measure customer satisfaction precisely in the absence of actual data, one can estimate the expected impact by taking the probability and seriousness of problems reaching the customer into account. For example, a model based on defect rates, defect severity, and post-release assistance responsiveness could be created to predict customer satisfaction.
If these variables have a measurable impact on consumer happiness, one could contend that:

a. Increased Initial Defects:
Lower initial customer satisfaction scores would result from higher initial defect rates under the "economy plan".

b. Effects of Fixes Released After:
Frequent patches or updates may be required, which may further impair user experience and eventually result in a drop in customer satisfaction and loyalty.

Summary

In summary, a product with more problems at release is probably the result of the "economy plan," which limits the scope and efficacy of QA efforts. As users meet and resolve these faults over the course of the product's lifecycle, this rise in defects may have a detrimental effect on customer satisfaction. The comparison with a typical QA strategy, which strives for an initial quality level that is higher, emphasizes how crucial complete QA operations are to preserving and raising customer satisfaction.

3. **Compare the overall results of the "economy plan" to the results of the standard and comprehensive plans.**

Answer:

Defect Removal Efficiency (DRE)

a. **Economy Plan:**
A limited set of QA activities are used in this technique, specifically comprehensive system tests and standard design reviews, which have been measured with DREs of 50% and 60%, respectively. Projects with tight budgetary or schedule constraints might find the plan's simplicity appealing, but it might also lead to a lower overall DRE. A significant portion of defects might remain undetected until much later in the development process or potentially after the product is released as a result of this diminished efficiency.

b. **Standard Plan:**
A typical quality assurance plan typically includes a greater range of quality assurance tasks, such as comprehensive code reviews, unit testing, integration testing, and acceptance testing, in addition to design reviews and system testing. A higher overall DRE is usually the consequence of fewer issues finding their way into later phases or being detected by end users, thanks to the diversity of testing and validation approaches. The higher upfront costs associated with these activities are mitigated by the decline in costly late-stage defect rectification.

c. **Comprehensive Plan:**
This strategy represents an even stronger dedication to QA in order to optimize DRE; it may use automated testing tools, complex testing techniques, and continuous integration/continuous deployment (CI/CD) processes. Although this technique requires the highest initial investment in quality assurance, there is the strongest chance that it will reduce post-release errors, which will save overall expenses and increase customer satisfaction.

Costs

a. **Economy Plan:**
Because the scope of QA tasks is decreased, it looks cost-effective at first. However, as our previous analysis shown, the increased costs associated with addressing post-release or late-stage faults may result in a higher total cost of ownership.

b. **Standard Plan:**
provides a balanced approach, with moderate upfront QA expenditures and much lower late-stage fault correction costs. When comparing this balance to the economy plan, the total cost of ownership is frequently lower.

c. **Comprehensive Plan:**

decreases the likelihood of late-stage problems and the hefty repair expenses that come with them, but it does require the largest upfront investment in QA. When taking into account the entire lifecycle of the product, this strategy can frequently result in the lowest total cost of ownership.

Customer Satisfaction

a. **Economy Plan:**

Increased defect rates in items that have been released run the danger of having a detrimental effect on customer satisfaction, which could harm the company's brand and cause consumers to lose faith in the product.

b. **Standard Plan:**

This strategy can assist preserve or increase customer satisfaction by lowering the quantity of faults that are delivered to the client, striking a balance between the fair QA costs and the requirement for quality.

c. **Comprehensive Plan:**

strives for the best possible product quality, which could result in the best possible levels of client satisfaction. This strategy works especially well for items where faults might have serious repercussions or if quality is a key differentiator in the marketplace.

Summary

Knowing the trade-offs between an economy, standard, and thorough QA plan will help you make the right decision:

a. **Risk Management:**

Think about the acceptable risk in terms of possible flaws finding their way into production.

b. **Budgetary Constraints:**

The intended degree of quality, the possible long-term expenses of defect correction, and the available budget should all be taken into consideration.

c. **Market Expectations:**

Make that the QA plan meets customer expectations regarding the quality of the product, particularly in areas where quality is a crucial difference.

Eventually, even though the "economy plan" could appear like a good idea from a short-term financial standpoint, it's a risky option for many projects due to its potential for increased long-term expenses and detrimental effects on customer satisfaction. Better long-term value can be obtained from a more thorough or balanced approach to quality assurance, both in terms of cost savings and in preserving or raising customer satisfaction.

4. **Based on your answer to 3), suggest some general rules about selecting the preferred assurance plan.**

Answer:

Choosing the best Quality Assurance (QA) plan for a project is a crucial choice that can have a big impact on the customer's overall happiness as well as the project's success. A number of important factors that balance the requirement for quality with the limitations of time, money, and project complexity should be taken into account while selecting a QA plan. The following general guidelines and factors should be taken into account while choosing the best QA plan:

A. Recognize the risks and requirements of the project

i. Complexity and Size:
To appropriately address the increased potential of faults, larger and more complex projects frequently call for more extensive quality assurance strategies. More affordable QA techniques could be a good fit for straightforward projects.

ii. Criticality:
To reduce the chance of disastrous flaws, projects that are essential to security, safety, or have a big impact on business should give priority to thorough quality assurance.

iii. Regulatory Compliance:
Because of required testing and documentation, projects that must adhere to regulatory compliance may not have the freedom to choose less extensive QA procedures.

B. Evaluate the Availability of Resources

i. Budget Restrictions:
The choice of QA plan can be greatly impacted by the amount of money allocated for QA activities. Though they need more resources, comprehensive QA programs provide the best potential for defect detection and correction. When money is tight, it may be necessary to create a "economy plan," but it's crucial to consider the possible long-term consequences of this choice.

ii. Time Restrictions:
The selection of a QA plan is also influenced by project timelines. Strict timelines might force QA efforts to be compromised, however this should be weighed against the possibility of post-release flaws and the related expenses.

C. Assess the Effect on Client Contentment

i. Quality Expectations:
The degree of investment in QA can be determined by the expectations of customers regarding quality. More thorough QA strategies could be required for projects with high standards of quality in order to guarantee client satisfaction.

ii. Feedback Loops:

Iterative QA methods, in which problems are addressed in continuous cycles of feedback and improvement, may be beneficial for projects that permit early and continual client feedback. This could allow for a more dynamic approach to QA planning.

D. **Take Long-Term vs. Short-Term Costs into Account**

i. Deferred versus Immediate Costs:
It's important to take into account the possibility of higher expenses associated with defect rectification later in the project lifecycle, including post-release, even though a "economy plan" can lower immediate QA costs.Q

ii. Total cost of ownership, or TCO:
Examine the total cost of ownership (TCO), which takes into account the expense of flaws both before and after the product's creation. A more thorough QA strategy may lower total cost of ownership (TCO) by lowering the quantity and seriousness of problems that require post-release repair.

5. **Does the comparison of the above results support the belief that investing in verification processes in the early stages of a project reduces the total costs of the SQA activities?**

Answer:

One method that promotes early defect detection and resolution is to invest in verification processes early in the software development lifecycle (SDLC). This method is based on the idea that as faults move through the stages of development, they get exponentially more expensive and challenging to rectify. Here is a thorough analysis of the project's effects and the wider ramifications of this approach when it comes to early investment in verification processes:

A. **Enhanced Economic Performance**

i. Reduced Cost of Defect Resolution:
It is less expensive to correct defects that are discovered early on (during requirements analysis or design, for example) than those that are discovered later on (during system testing or after deployment, for example). This is due to the fact that early failures need modifications to the design or documentation, which are less expensive and labor-intensive than code modifications or operational corrections after launch.

ii. Diminution of Total Expenses:
Investments in early verification can help projects cut their overall quality assurance costs considerably. Defects that are detected early on are less likely to spread to later stages, which lessens the total load of defect management and resolution.

B. **Better Schedules for Projects**

i. Faster Time-to-Market:
By preventing the delays brought on by late-stage defect detection and resolution, early verification aids in the maintenance of project timeframes. It is less probable for projects to run across major obstacles at the last minute that could cause release deadlines to slip.

ii. Predictable Development Cycles:

Project planning becomes more dependable as a result of fewer surprises in the form of late-discovered flaws, enabling more precise timetables and resource allocation.

C. Improved Quality of the Product

i. Better Quality Outcomes:

Generally speaking, earlier defect identification leads to better overall product quality. The end result is more reliable, performs better, and offers a better user experience since possible problems are fixed before they become deeply ingrained in the software.

ii. Customer Satisfaction:

It goes without saying that products with fewer flaws and more dependability will have happier customers. Better market reception, higher sales, and a better reputation for the brand can result from this.

D. Advantages in Strategy

i. Competitive Edge:

Businesses with effective QA procedures may be able to offer high-quality products more quickly, giving them a market advantage.

ii. Cost-Benefit Optimization:

By more efficiently allocating resources across the development lifecycle, early investment in verification enables firms to maximize their QA expenditures.

E. Accommodates Continuous and Agile Delivery Models

i. Agile Development:

The foundation of agile development approaches is early and ongoing verification, which facilitates quick iterations and the early delivery of working software components.

ii. Promotes Continuous Deployment/Continuous Integration (CI/CD):

For CI/CD pipelines to enable automated testing and deployment with a high level of confidence in software quality, early verification techniques are critical.

Part 2

We introduced the SQA process implementation activities, including establishing SQA processes and their coordination with relevant software processes, coordinating SQA plan and project plan, reviewing proposals and contracts, modeling the cost of software quality, and controlling SQA records and documentation.

In this part of the group discussion, each group is required to

1. **Read through the textbook chapters 6 through 11.**

Answer:

Summary of concepts we have learnt from Chapter 6 to Chapter 11:

Chapter 6: Establishing SQA Processes and their Coordination with Relevant Software Processes

- This chapter focuses on the establishment of Software Quality Assurance (SQA) processes and their coordination with other software processes to ensure the overall quality of software development.

Chapter 7: SQA Plan and Project Plan

- This chapter discusses the creation of a Software Quality Assurance (SQA) plan and its integration with the project plan, providing a roadmap for maintaining and enhancing software quality throughout the project lifecycle.

Chapter 8: Preproject Process - Contract Review

- The pre project process explores the preproject process, specifically the contract review phase, emphasizing the importance of thoroughly reviewing contracts to set the foundation for successful software development projects.

Chapter 9: Cost of Software Quality

- Examines the concept of the Cost of Software Quality, delving into the economic considerations and investments related to ensuring high-quality software development.

Chapter 10: The Effectiveness and Cost of a V&V Plan - The SQA Model

- This explores the effectiveness and cost implications of a Verification and Validation (V&V) plan within the Software Quality Assurance (SQA) model, emphasizing the role of thorough testing and validation processes in achieving software quality.

Chapter 11: SQA Records and Documentation Control

- Focuses on Software Quality Assurance (SQA) records and documentation control, highlighting the importance of maintaining accurate records and effective documentation throughout the software development process to ensure transparency and accountability.

2. **Select one SQA process implementation activity.**

Answer:

We chose the Standards, practices, and conventions for software projects SQA process implementation activity. The following two articles are based on the activity.

3. **Search for at least two recent publications on the SQA process implementation activity you selected.**

Answer:

Article 1:
Name: Impacts of Coding Practices on Readability
Link: https://ieeexplore-ieee-org.libproxy.csudh.edu/document/8343621
Description: This paper highlights the prevalence of coding standards for improving maintainability in software development but acknowledges persistent issues with low readability. The paper outlines a survey investigating the impact of coding practices on code readability, revealing statistically significant effects of certain practices, and exploring correlations between readers' characteristics and their perceptions of readability.

Article 2:
Name: Software Quality as a Subsidy for Teaching programming
Link: https://ieeexplore-ieee-org.libproxy.csudh.edu/document/9637475
Description: With regards to SQA, the predefined standards for software development must be followed in any software development process. This IEEE paper proposes an approach to enhance the teaching of programming by integrating software quality concepts. It highlights the challenge of graduates feeling unprepared for the job market due to a gap between industry standards and academic teachings. The paper introduces the Teacher Mate tool, which helps teachers focus on students' programming difficulties, particularly in writing high-quality code. The approach emphasizes teaching programming with a focus on internal code quality, using code inspection tools to tailor content to students' needs. The paper's contributions include defining an approach to using source code as a basis for teaching quality and providing guidelines and tools to support this approach. The tools developed are open source, available for adoption by the academic community. Future work includes validating the approach in the classroom to enhance students' coding skills and internal code quality.

4. Read and discuss the articles you selected.

Answer:

Summary of Article 1:(Impacts of Coding Practices on Readability):

The article conducts a comprehensive examination of the impact of specific Java coding practices on code readability, leveraging a web survey that includes both students and professional programmers. The evaluation of eleven coding practices, guided by established readability models, uncovers nuanced insights into developer preferences. Positive effects on readability are observed for practices emphasizing the use of blank lines following curly braces and maintaining line lengths within 80 characters. Conversely, a decrease in perceived readability is noted when curly braces are placed on the same line as clauses. This study not only provides practical guidelines for developers and educators but also delves into potential correlations with gender and programming experience. The findings underscore the necessity for broader research encompassing diverse programming languages and industries, offering valuable considerations for optimizing code readability.

Summary of Article 2: (Software Quality as a Subsidy for Teaching programming)

The research paper titled "Software Quality as a Subsidy for Teaching Programming" discusses the importance of integrating software quality concepts into the teaching of programming. It highlights the gap between industry expectations and academic teachings regarding code quality and proposes using software quality as a foundation for teaching programming. The paper introduces the Teacher Mate tool, which utilizes SonarQube for code inspection and provides visualizations to help teachers and students understand and improve code quality. Overall, the paper aims to better prepare students for industry standards and reduce the gap between academia and industry expectations.

5. Summarize your readings as a reading report.

Answer:

Key takeaway from Article 1:

- The study investigates the impact of specific Java coding practices on code readability through a web survey involving students and professional programmers.
- Eleven coding practices are assessed based on established readability models, offering insights into developers' perceptions of code snippets.
- Positive effects on readability are observed for practices promoting blank lines following curly braces and maintaining line lengths below 80 characters.
- Placing curly braces on the same line as clauses is identified as a factor contributing to decreased code readability.
- The web survey methodology captures diverse perspectives within the developer community, highlighting the subjective nature of readability evaluations.
- While some correlations with gender are noted, the article suggests that gender differences may not significantly impact the reasoning behind readability preferences.
- Exploring potential correlations with programming experience, the study indicates the need for broader research to draw definitive conclusions across different languages and industries.
- The collected dataset could contribute to the development of coding samples for educational purposes, tailoring coding standards to better suit the needs of students.
- The article concludes by emphasizing the necessity for additional research encompassing diverse programming languages and industries to generalize findings and enhance our understanding of code readability considerations.

Key takeaway from Article 2:

- The paper emphasizes the importance of well-written code in software development, linking it to the programmer's experience and industry demand for qualified professionals.
- It notes that codes not meeting standards and quality specifications tend to be complex, poorly written, and difficult to understand, leading to costly activities.
- Despite well-defined concepts of code quality in the industry, the paper suggests that academic teachings often fall short of meeting market expectations.
- Graduates reportedly feel unprepared for the job market, facing challenges in competing for the best opportunities, indicating a gap between industry and academia.
- The research aims to bridge this gap by proposing improvements in teaching programming, focusing on internal code quality.
- The proposed approach uses software quality as a foundation for teaching programming, providing guidelines for teachers to focus on internal code quality.
- The paper introduces the Teacher Mate tool, which utilizes SonarQube for code inspection, allowing teachers to identify students' lack of good programming practices.

- The tool provides teachers with a better understanding of students' progress and proposes guidelines to assist in conducting programming disciplines.
- From the student's perspective, the approach aims to motivate them to learn programming with a focus on how solutions are developed, not just the results.
- The paper addresses the challenge of introducing internal code quality teaching in programming subjects without increasing the workload for teachers.
- It highlights the importance of early exposure to software quality concepts and suggests that teaching software quality should begin as early as possible.
- The gap between academia and industry regarding code quality is also identified in the paper, suggesting that good practices could promote critical thinking in students.
- The application of code quality concepts is seen as beneficial for students during the course and after graduating, reducing the gap between academia and industry.
- Overall, the paper advocates for integrating software quality concepts into teaching programming to better prepare students for industry standards and expectations.