PA2557 – QUALITY CONTROL

ASSIGNMENT 3

SRI SAI RIPUGHNA RISHITOSH SONTI
Department of Software Engineering
P-no: 19960916-8357
sosr17@student.bth.se

GANNA ANIL
Department of Software Engineering
P-no: 19960710-7514
gaam17@student.bth.se

MANPREETH BOLLU
Department of Software Engineering
P-no: 19970403-1237
bomn17@student.bth.se

SRAVANI BEJAWADA
Department of Software Engineering
P-no: 19970525-9563
besr@student.bth.se

PAVAN KUMAR PULA
Department of Software Engineering
P-no:19970506-6919
pupa17@student.bth.se

***Abstract* – This paper mainly focuses on improving the code quality using the goal question plus strategy (GQM+). This helps us to learn and understand how to implement a GQM+ strategy and increase our understanding of the improvement of the code quality. The quality of the code is improved based on the software quality metrics which were measured in the evaluation of code quality through code reviews assignment.**

SECTION – I

INTRODUCTION

The GQM approach develops a set of measurable goals for improving the aspects of the software productivity and quality through deriving questions for the goals, taking specific measures to answer those questions, check the answers satisfy the questions in a quantifiable way [1]. The GQM+ strategies method adds as the extensions on top of the GQM approach to achieve the business goals and strategies. These strategies are explicit to deal with the business goals such as profit, production costs, quality of the software, functionality of the software, customer satisfaction etc [1]. The GQM+ strategies method consists of six different steps in focus to develop, implement and learn. These steps are followed iteratively until the organizational goal is achieved. To implement the GQM+ strategies, measurement goals and strategies are taken and the GQM approach is considered to achieve these measurement goals and strategies of the organization. In this assignment, we implement the GQM+ strategy to improve the quality of the code considering the measures of the code review evaluation.

SECTION – II

DESCRIPTION OF THE QUALITY ATTRIBUTES

The quality attributes we want to improve are

- READABILITY.
- MAINTAINABILITY.

These quality attributes are considered as they are useful to satisfy our organizational goal which helps in improving the quality of the code while using GQM+ strategies.

In our assignment on the code quality evaluation through code reviews, there were critical code smells or bugs identified. After the evaluation of the code quality by tools, several instances of duplication of the code, styling, and complexity value issues in various part of the code were detected [2]. These code smells are again detected in the manual evaluation of the code quality.

Justification Of The Quality Attributes:

Readability Of Code:

Readability of the code is commonly believed to impact the overall quality of software [3]. Developers make readability as their top information needs quality attribute. Code that is difficult can have wide-ranging consequences [3]. This may lead to changes that introduce potential bugs. This makes the code readability and understandability difficult. To achieve the goal i.e., customer satisfaction, the code should be easy to use and read which results for a better understanding of the outcomes from the software [4].

In our case, the readability of the software can be improved by reducing the faults and adding more comments to the lines of code as we are dealing with a code of game software. We consider this because if the customer is satisfied with the performance of the software, more accurate results may conclude that the software is working efficiently, and developments can be made later with the given review or feedback from the customer.

Maintainability Of Code:

The maintainability of code can be defined as the ease with which a software project or system can be modified to reduce the code smells or bugs, improve the performance, efficiency or developing with the change in environment frequently [5] [6]. The maintainability can affect the performance of the software and its efficiency. This gives bad feedback from the customer which means failure of the project.

Several articles [7] [8] [9]stated that maintainability of software is only difficult because of the most frequent issues in complexity and duplication of the code. For this assignment, we have chosen these two code smells apart from the other code smells such as styling as these two code smells are the most frequently occurring that affect the maintainability and results in the code difficult to understand.

SECTION – III

In this section, GQM+ strategy is implemented. We discuss the implementation of the GQM approach followed by the GQM+ strategy. For GQM+ strategy certain rules are to be implemented. The process is as follows:

GQM+ STRATEGIES ORGANIZATIONAL PLANNING AND CONTROL:

The organizational goal is to improve customer satisfaction. To achieve this goal, there are certain measurements to be considered and to achieve this we follow the GQM approach. The description of the goal should be in detail in the form of GQM+ strategies goals template.

- **Activity:** Improve.
- **Focus:** Customer Satisfaction.
- **Object:** A Chinese-Checkers Game Software.
- **Magnitude (Assumption):** 15% of positive feedback more than the previous release.
- **Timeframe:** At-least 1 year.
- **Scope:** Development in the organization teams.
- **Relations with other goals:** Readability, Maintainability.

Questions: Q1. Does the software perform responsively, or does it seem unnecessarily slow?

Q2. What changes should be made to the software from the customer feedback?

Metrics: M1. Efficiency (Performance).

There are 2 strategies which help in achieving the organizational goal. The strategies are

Strategy 1: Increase the productivity of the software.

Strategy 2: Improve quality/ Customer Satisfaction.

For the above strategies, the following measurements are considered.

Strategy 1: Increase the productivity of the software.

Measurement 1: Commented Lines of Code per Effort.

Strategy 2: Improve quality/ Customer Satisfaction.

Measurement 1: Commented Lines of Code.

Measurement 2: Customer Feedback.

We have considered 2 quality attributes that are needed to improve i.e., readability and maintainability. To achieve the organizational goal through these quality attributes, we implement the GQM approach for these quality attributes:

Goal 1: Increase the readability of the code.

Question 1: What developments are done to software to settle in the changing requirements?

Metrics 1: Commented Lines of Code.

Goal 2: Improve the maintainability of the code.

Question 1: What are the instances of code duplication in the code?

Question 2: What is the level of the complexity in the code?

Metrics 1: Duplicated Code.

Metrics 2: Cognitive Complexity.

Metrics 3: Cyclomatic Complexity.

Description of Metrics used in relation to the quality attributes of code:

- **Commented Lines of Code (CLOC):** The commented lines of the code are defined as the count of several lines of code, only the commented lines mentioned in the code without the whitespaces. With the increase in the CLOC count, the complexity of the code decreases and the readability of the code increases [3]. This leads to the readability and the understandability will be easy to maintain. This improves the quality of the code and helps the customer to meet the requirements.

- **Duplication of Code:** The code duplication metric is defined as the count of the instances of the code repeated or duplicated more than twice [2]. The occurrence of the duplicated code results in difficult to understand by the developers and the understandability increases. Hence the maintainability increases i.e., the code is difficult to maintain. The reason for difficult to maintain is that if the code needs to be modified then the instances of the duplicated code should be modified which cause several errors in the code.

- **Cognitive Complexity:** The cognitive complexity is defined as the level of inheritance of a certain part of the system which is dependent on the other parts of the system [10]. If the cognitive complexity increases, it is difficult for the developers to understand the logical operations of the code and will be difficult to maintain. Hence the maintainability increases.

- **Cyclomatic Complexity:** The cyclomatic complexity quantifies software complexity. Complexity is estimated by counting the maximum number of linearly independent systems in code. In other words, cyclomatic complexity is directly proportional to the complexity of the software [6]. This leads to difficulty in understandability by the developers and hence the maintainability increases i.e., the code will be difficult to maintain.

REFERENCES:

[1] V. R. Basili *et al.*, “GQM+StrategiesÆ: A Comprehensive Methodology for Aligning Business Strategies with Software Measurement,” p. 15, 2007.

[2] M. V. Mäntylä and C. Lassenius, “What Types of Defects Are Really Discovered in Code Reviews?,” *IEEE Transactions on Software Engineering*, vol. 35, no. 3, pp. 430–448, May 2009.

[3] U. A. Mannan, I. Ahmed, and A. Sarma, “Towards understanding code readability and its impact on design quality,” in *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering - NL4SE 2018*, Lake Buena Vista, FL, USA, 2018, pp. 18–21.

[4] C. F. Kemerer and M. C. Paulk, “The Impact of Design and Code Reviews on Software Quality: An Empirical Study Based on PSP Data,” *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 534–550, Jul. 2009.

[5] M. Riaz, E. Mendes, and E. Tempero, “A systematic review of software maintainability prediction and metrics,” in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, Lake Buena Vista, FL, USA, 2009, pp. 367–377.

[6] G. K. Gill and C. F. Kemerer, “Cyclomatic complexity density and software maintenance productivity,” *IEEE Transactions on Software Engineering*, vol. 17, no. 12, pp. 1284–1288, Dec. 1991.

[7] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, “The influence of non-technical factors on code review,” in *2013 20th Working Conference on Reverse Engineering (WCRE)*, Koblenz, Germany, 2013, pp. 122–131.

[8] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, “The impact of code review coverage and code review participation on software quality: a case study of the qt, VTK, and ITK projects,” in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, Hyderabad, India, 2014, pp. 192–201.

[9] A. Bacchelli and C. Bird, “Expectations, outcomes, and challenges of modern code review,” in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 712–721.

[10] J. Rilling and T. Klemola, “Identifying comprehension bottlenecks using program slicing and cognitive complexity metrics,” in *MHS2003. Proceedings of 2003 International Symposium on Micromechatronics and Human Science (IEEE Cat. No.03TH8717)*, Portland, OR, USA, 2003, pp. 115–124.