# EXAMINING THE IMPACT OF SOFTWARE TESTING PRACTICES ON SOFTWARE QUALITY IN BATAM SOFTWARE HOUSES

Suwarno[1], Syaeful Anas Aklani[2], Nellsen Purwandi[3]
[1,2,3]Universitas Internasional Batam
Batam, Kepulauan Riau, Indonesia 29444
*e-mail: suwarno.liang@uib.ac.id[1], syaeful@uib.ac.id[2], 2131022.nellsen@uib.edu[3]*

**Abstract** - This research aimed to investigate the impact of software testing practices on software quality in software companies in Batam, Indonesia. It focused on identifying key factors such as Software Testing Knowledge, Software Testing Approach, and Software Testing Complexity and analysing their correlation with software quality. Data was collected from 48 respondents, including project managers, developers, and QA teams, using a questionnaire distributed via Google Forms and convenience sampling. The questionnaire was designed based on related studies to ensure relevance to the respondents' roles. Regression analysis identified significant impacts of testing complexity, approach ($p = 0.000$), and knowledge ($p = 0.003$) on software quality. The F-test result ($F = 32.622$) confirmed a strong relationship between testing practices and software quality. These findings emphasise the critical role of robust testing strategies in enhancing software quality. For companies in Batam, the study offers actionable insights, including adopting structured frameworks, and preferable action on testing approach. Implementing these strategies can help organisations improve software outcomes and maintain competitiveness in the evolving software development landscape.

**Keywords** - Software Quality, Regression Analysis, Software Testing

## I. INTRODUCTION

As the software development and deployment landscape evolves, ensuring software quality and reliability has become increasingly critical for organizations across industries [1][2]. Modern practices such as Agile, automated testing, and CI/CD have significantly enhanced efficiency, maintainability, and quality by enabling rigorous testing and streamlining delivery processes [3]. However, these benefits rely on robust Software Quality Assurance (SQA), which plays a vital role throughout the software lifecycle—from understanding customer needs to product delivery. SQA helps prevent defects, improve processes, and ensure the final product meets expectations, reducing time, effort, and costs [4].

Despite its importance, SQA faces challenges like a lack of skilled professionals, insufficient knowledge, and limited resources, leading some organizations to scale back assurance activities [5]. Nonetheless, software testing remains indispensable for early defect identification and reliable outcomes. Testing employs diverse methods tailored to project needs, ensuring that individual components and integrated systems function as intended [6][7]. By adopting comprehensive practices, companies mitigate risks, enhance reliability, and meet client expectations, gaining a competitive edge both locally and globally.

Meanwhile, The software industry in Southeast Asia has grown significantly, with Singapore, Malaysia, and Indonesia emerging as key players. Among them, Batam, an island city in the Riau archipelago of Indonesia, has rapidly become a prominent hub for software development, serving both domestic and international markets. Its strategic location near Singapore and

Malaysia makes it an attractive investment destination, fueling the expansion of the region's software industry [8].

Building on this context, this study aims to fill the knowledge gap by examining how different dimensions of software testing such as the level of knowledge among practitioners, the approaches adopted by teams, the challenges or difficulties faced during implementation, and the specific testing methodologies employed collectively influence the overall software quality within Batam's software houses. By exploring these interconnected aspects, the research seeks to provide a comprehensive understanding of how testing practices contribute to software reliability, security, maintainability, and other key quality metrics, highlighting specific opportunities for improvement within Batam's software development landscape.

## II. SIGNIFICANCE OF STUDY

### *A. Literature Review*

A study on software testing practices in Cameroon revealed significant challenges in adopting structured testing and automation. Over 80% of respondents relied on developer-led testing without formal methodologies, and automated testing comprised less than 8% of tests. Key barriers included time and cost constraints, lack of perceived usefulness, and concerns about maintaining automated tests. The findings underscore the importance of promoting effective testing practices in resource-constrained environments like Cameroon [5].

A study on test automation maturity in the software industry, based on data from 151 respondents in 101 organizations across 25 countries, revealed significant variation in practices. While 85% reported adequate automation skills, 47% lacked guidelines for automated test design and execution. Higher maturity correlated with a greater percentage of automated test cases and Agile/DevOps adoption. QA engineers were more optimistic about maturity levels than consultants. The findings emphasize the need to address skill gaps, adopt mature practices, and establish guidelines to maximize the benefits of test automation [9].

A qualitative study on software quality practices in startups revealed that these companies often adopt limited approaches shaped by team maturity, organizational culture, and prior experiences. Startups typically address quality reactively, responding only to issues that affect the product, business, or customers, or when technical debt becomes overwhelming. However, the study's reliance on interviews posed a threat to construct validity, as responses from participants could introduce gaps, biases, or inaccuracies in the findings [10].

This study builds on previous research by adopting a purely quantitative approach to minimize biases and enhance objectivity, addressing limitations from earlier studies that often combined qualitative and quantitative methods. Unlike prior research, which primarily focused on technical testing aspects, this study broadens its scope to include testing knowledge and complexity for a more comprehensive analysis. To improve response validity, the survey incorporates specific and targeted questions, moving beyond the general queries used in earlier studies. Additionally, regression analysis is employed to explore the relationships between knowledge, complexity, and testing practices, providing deeper insights and addressing gaps in statistical rigor. While similar studies exist in other regions, there is limited research on software testing practices in Indonesia. This study fills that gap, offering valuable perspectives on the local software industry.

*B. Research Method*

This study used a questionnaire as the primary validation method due to its ability to collect extensive data from a sample size at a relatively low cost. A questionnaire ensures all respondents receive the same set of questions, maintaining consistency in data collection and enhancing result reliability, particularly for complex research questions in specific contexts.

*1. Questionnaire Construction*

Building on issues in software testing procedures identified in related work and their impact on software quality, this study developed research questions and a data collection tool. The goal was to explore how current software testing knowledge, practices, and complexities influence quality across companies. The research questions (RQs) were:

- RQ1: What do managers, developers, and quality control (QC) professionals know about software testing?
- RQ2: What software testing approaches are used by managers, developers, and quality control (QC) professionals in projects?
- RQ3: How do managers, developers, and quality control (QC) professionals perceive the complexity of the testing process?
- RQ4: How successful is the project in terms of software quality attributes?

To design a focused survey, we reviewed similar studies and crafted questions addressing recent themes while minimizing redundancy, and ensuring relevance and clarity for respondents. The questionnaire includes five sections, beginning with respondent profiles through four closed-ended questions: age, job title, years of experience, and company size. This structure aligns with methodologies from related research [11], [12].

The second section of the questionnaire analyzes respondents' understanding of software testing, which is vital for ensuring systems function as intended and meet requirements. Expertise in software testing includes knowledge of various procedures, approaches, and strategies. This section evaluates respondents' proficiency through two key questions assessing both theoretical and practical knowledge. The evaluation uses a 1-5 scale, with 1 representing "very low" and 5 representing "very high." This approach builds on earlier studies [12], which similarly explored the level of software testing expertise among professionals in the field.

The third section examines the testing approaches used in respondents' projects, focusing on objectives, techniques, and perceived benefits. It draws on previous studies, such as Eisty et al. [13], and it categorizes methods into unit, integration, system, acceptance, and module testing [7], [14]. Additionally, testing techniques are classified into black box, white box, and grey box based on prior research [15]–[17]. This section includes closed-ended questions about testing objectives and a series of 1-5 scale questions to assess the frequency, usefulness, and application of each testing method.

Another section examines software testing complexities and their impact on software quality. As modern systems grow more complex, adopting well-defined testing approaches is increasingly vital to understanding how varying methods influence project outcomes. This section has previously been conducted to determine the specific complexities and challenges that have been encountered when conducting software testing in the organization to gain a more comprehensive understanding of the testing landscape [13], [18]. Respondents are asked to rate

the complexity of the testing process in their projects and to evaluate the level of challenges faced during their testing activities using a 1-5 scale.

The final section of the survey addresses software quality attributes, a critical element structured around a standard software quality model. It evaluates key attributes such as reliability, security, maintainability, functionality, performance efficiency, compatibility, portability, usability, and overall quality. These evaluations aim to reveal the influence of different testing approaches and knowledge levels on product quality. Drawing on prior research, including Binboga and Gumussoy [19], which examined factors affecting software quality, this section builds on their findings to assess these attributes comprehensively. Seven targeted questions guide this analysis, with responses rated on a 1-5 Likert scale.

*2. Variables and Hypothesis*

The instrument is based on the model's variables from the research questions, including three independent variables (Software Testing Knowledge, Approach, and Complexity) and one dependent variable (Software Quality), as shown in Figure 1. These variables interact to provide a comprehensive evaluation of Software Quality Assurance, allowing for a detailed analysis of the software product and development environment.

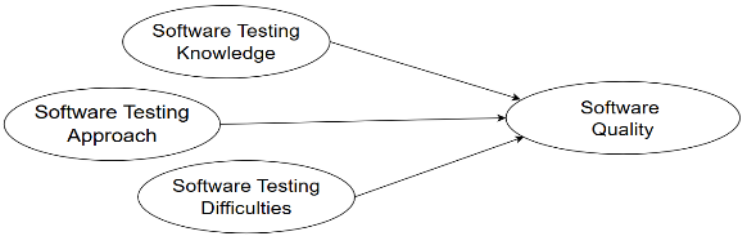



Figure 1. Research Model

Figure 1 uses a model in this study that binds between variables with other variables. Thus, several hypotheses can be made based on the relationship between the variables. The hypothesis used in this study is as follows:

H1 = Software Testing Knowledge Affects Software Quality in Batam Software House.
H2 = Software Testing Approach Affects Software Quality in Batam Software House.
H3 = Software Testing Complexity affects Software Quality in Batam Software House.

*3. Sampling*

Similar to the research by Sultana et al. [15], this study used Google Forms to collect data. The conclusive iteration of the online survey was disseminated using the convenience sampling method as one of the dominant approaches widely used in surveys [9]. Our study targeted 48 valid responses from Developers, QA/QC, and Project Managers in Batam, Indonesia, all of whom were actively involved in the software development process. like the study conducted by author Neves, Lucas, et al. [12]. Afterwards, the authors conducted a pilot test of the questionnaire, following the approach used by author Eisty et al [13] and author Martins, Luana, et al [20], to thoroughly review the questions. The results of the pilot test will be employed to evaluate the reliability and validity of the respondent's answers.

*4. Data Analysis*

We will use multiple regression analysis in SPSS to test the hypotheses and evaluate relationships between software testing practices and software quality. This approach examines the combined impact of independent variables (such as software testing knowledge, approach,

and complexity) on the dependent variable, software quality. Specifically, the study will apply several diagnostic tests to ensure the validity and robustness of the model:

- Validity and Reliability Testing: The survey instruments were assessed using Pearson Correlation Coefficients ($p < 0.05$) to ensure meaningful relationships aligned with the study's constructs, and Cronbach's Alpha (above 0.6) for internal consistency.
- Outlier Detection: Outliers were identified using Z-scores, flagging values greater than 3 or less than -3 to prevent distortion.
- F-Test: Assessed the statistical significance of the regression model, determining if the independent variables explained a significant portion of variance in software quality.
- R Square: Measured how well the independent variables accounted for the variation in software quality, with higher values indicating a better fit for the model.
- T-Test: Evaluated the significance of each independent variable's impact on software quality, helping identify the most influential factors.
- Normality Test: tested residuals for normality to ensure normally distributed errors, which is crucial for valid hypothesis testing in regression analysis.
- Multicollinearity Check: Variance Inflation Factor (VIF) was calculated to detect multicollinearity among independent variables. High multicollinearity could distort the regression results, so any issues were addressed to ensure accuracy.
- Heteroscedasticity Test: Tests for heteroscedasticity were performed to identify non-constant error variances. If heteroscedasticity was present, adjustments were made to maintain the efficiency and reliability of the regression model.

This study refers to previous research by Authors Sitepu [21] and Siahaan [22], which also used regression analysis to test the impact of various factors on the optimization of information technology use. The findings from that research serve as a reference for the approach used in analyzing the survey results in this study.

## III. RESULTS AND DISCUSSION

### *A. Respondents Profile*

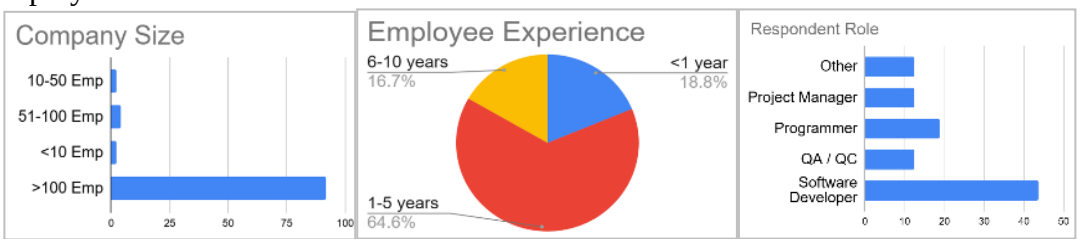
Based on the questionnaire results, as shown in Table 1, most respondents fall within the 17-24 age range, indicating a strong presence of younger professionals and students in the participant pool. The second largest group is aged 25-34, representing mid-career professionals with several years of industry experience. A smaller percentage is in the 35-44 age group, while no data was recorded for the 45-54 and 54+ age ranges. This age distribution highlights the workforce composition in the local software industry and may reflect generational trends in software testing practices.

TABLE I
RESPONDENTS PROFILE

| Age Group | Frequency | Percentage |
|---|---|---|
| 17 - 24 years old | 27 | 56.25% |
| 25 - 34 years old | 18 | 37.5% |
| 35 - 44 years old | 3 | 6.25% |
| Total | 48 | 100% |

As shown in Figure 2a, most respondents are employed by companies with over 100 employees. Only one respondent works for a company with fewer than 10 employees, and

another for a company with 10–50 employees. Two respondents are from companies with 51–100 employees.




(a) (b) (c)

Figure 2. (a) Company Size (b) Employee Experience (c) Respondent Role

Regarding software testing experience in Figure 2b, most respondents (31) reported having 1–5 years of experience. Nine respondents indicated less than a year of experience, while eight reported 6–10 years. No respondents reported more than 10 years of experience. For roles within the company, as shown in Figure 2c, the largest group of respondents are Software Developers (43.75%), followed by Programmers (18.75%). Project Managers, QA/QC personnel, and others each represent 12.5% of the total.

### *B. Software Testing Knowledge*

To better understand the respondents' software testing expertise, we concentrated on collecting information for RQ1, as illustrated in Figure 3, explicitly evaluating both theoretical and practical comprehension. Most respondents rated their theoretical knowledge at level 4, followed by levels 3 and 5. For practical knowledge, 47.9% also assessed their comprehension at level 4, with levels 3 and 5 each receiving 25%.

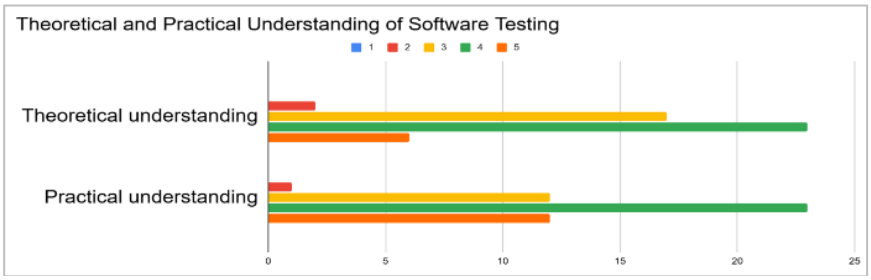



Figure 3. Employee's Software Testing Understanding

### *C. Software Testing Approach*

This Software Testing Approach section will elaborate on RQ2, focusing on software testing goals, testing levels, testing usefulness, and the specific techniques applied by teams in Batam software houses to assess their impact on software quality.

Figure 4 shows the results for software testing goals. At Level 0, only two respondents need to differentiate testing from debugging, showing a basic misconception. Similarly, at Level 1, two respondents view testing to demonstrate correctness, reflecting limited awareness. Level 2 sees a shift, with 10 respondents acknowledging that testing identifies software flaws, signalling a more analytical approach. Level 3, supported by 19 respondents, emphasizes testing as a risk management tool rather than proving correctness, showcasing maturity in understanding. Finally, at Level 4, 15 respondents perceive testing as a mental discipline fostering higher-quality software, underlining its role in promoting a culture of excellence.

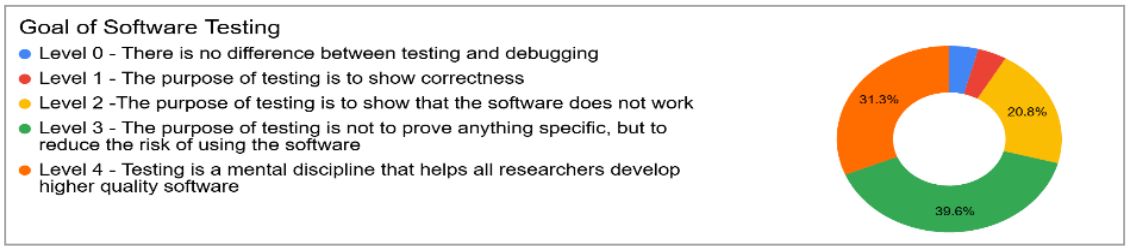



Figure 4. Employee's Software Testing Understanding

The survey results illustrate the frequency of various project testing methods, expressed in percentages, as shown in Figure 5. The software testing level focuses more on unit, system, and acceptance testing, which are used more frequently at higher levels, indicating their critical role in ensuring software quality. Unit testing is consistently emphasized, most likely due to its efficacy in detecting early-stage defects. System and acceptance testing are also heavily utilized, focusing on overall system functionality and stakeholder requirements. In contrast, integration and module testing are more evenly distributed, implying a balanced approach in which these methods are used based on project needs but not with the same intensity as the other testing phases.

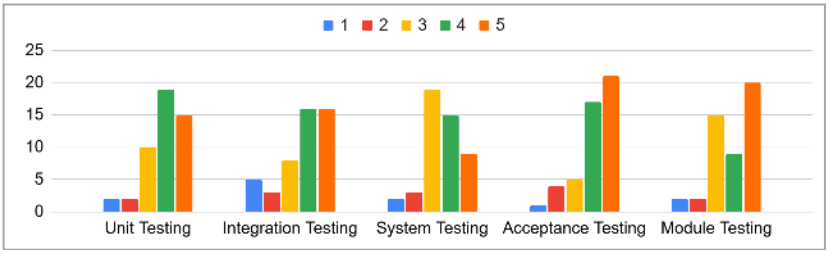



Figure 5. Software Testing Level

The survey shows that most respondents view software testing as highly beneficial, with 47.9% rating it a 5 and 39.5% rating it a 4, indicating its key role in project success. However, 6.5% rated it a 3, suggesting some areas for improvement, while 4% rated it a 2 and 2% a 1, reflecting limited value in some instances. These results highlight software testing as essential but point to opportunities for better integration and effectiveness in some projects.

The use of software testing techniques reveals a strong preference for white box testing, as shown in Figure 6, with the majority of responses concentrated at higher levels, emphasizing understanding internal code structures. Black box testing, which does not require knowledge of internal workings, has a more even distribution but still tends to be used more frequently. Grey box testing falls somewhere in the middle, displaying a range of responses with moderate to high levels of utilization, highlighting its hybrid method that incorporates both internal and external perspectives.

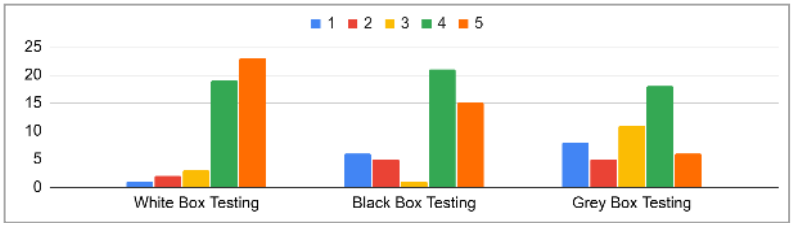



Figure 6. Software Testing Technique

## *D. Software Testing Complexity*

This section addresses RQ3 by analyzing respondents' perspectives on the complexities of software testing and the challenges they encounter. It explores factors influencing the effectiveness of quality assurance activities and how these complexities impact the success of software testing operations.

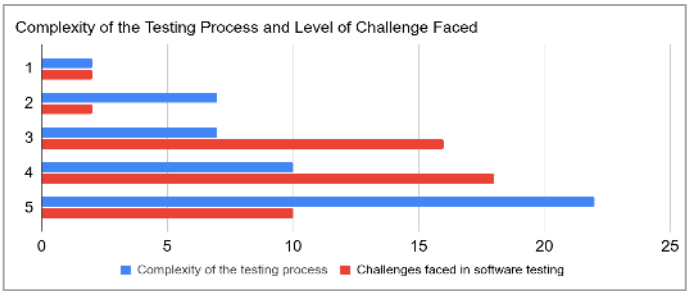



Figure 7. Complexity of the Testing Process and Level of Challenge Faced

Figure 7 illustrates that many respondents perceive the testing process as highly complex, with a significant portion describing it as very challenging. Most teams report encountering substantial complexity in their software testing activities, while only a small minority consider the process straightforward. Similarly, the challenges faced in software testing are generally rated as demanding, with a noticeable proportion of participants identifying them as significant obstacles. A smaller group views these challenges as moderate or minimal, indicating that the testing process remains a rigorous task for most teams.

*E. Software Quality*

This section will elaborate on RQ4 by examining the various dimensions of software quality, including reliability, security, maintainability, functionality, performance efficiency, compatibility, portability, usability, and overall quality.

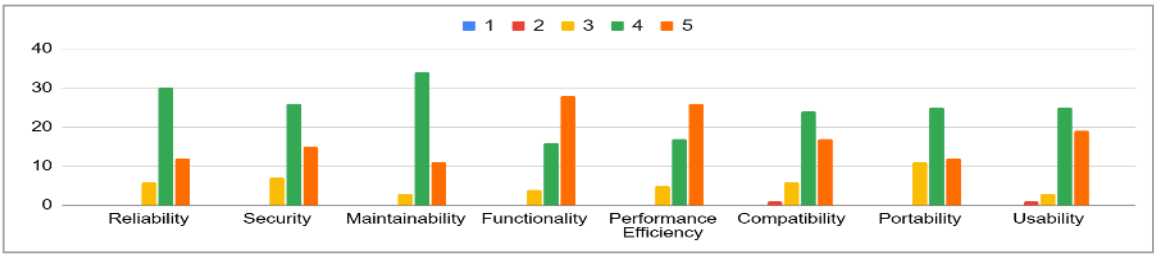



Figure 8. Software Quality Attributes

Figure 8 highlights strong software quality performance across dimensions. Metrics like reliability, security, and maintainability received high ratings, with over 85% of respondents scoring them 4 or 5. Functionality and performance efficiency also rated well, with positive ratings from 88% and 86% of respondents, respectively. Compatibility and portability were similarly well-regarded, earning favorable ratings from over 85%. Usability stood out with the highest positive feedback, as 91% of respondents rated it 4 or 5. The chart underscores the software's strengths in, ease of use, efficiency, and adaptability across platforms.

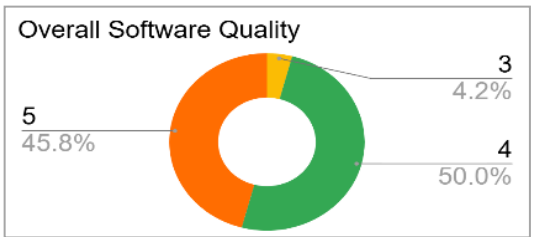



Figure 9. Overall Software Quality

Ultimately, the project's success in terms of quality, as shown in Figure 9, is reflected in the majority of responses concentrated at higher levels, particularly 4 and 5. This indicates that most respondents feel positively about the quality achieved during the project. The minimal responses at lower levels (1, 2, and 3) suggest few concerns regarding quality. Overall, the data demonstrates a strong perception of success in the project's quality outcomes.

### *F. Validity and Reliability*

This research assessed validity and reliability through pilot testing with 30 respondents. Validity was tested using Pearson Correlation Coefficients, requiring significant values below 0.05 and coefficients above 0.05, which all met the criteria. Reliability was evaluated using Cronbach's Alpha, with variables considered reliable if values exceeded 0.6.

TABLE 2
RELIABILITY TEST RESULT

| Variable | Cronbach's Alpha | Number of Items | Status |
|---|---|---|---|
| Software Testing Knowledge | 0.815 | 2 | Valid |
| Software Testing Approach | 0.931 | 10 | Valid |
| Software Testing Complexity | 0.710 | 2 | Valid |
| Software Quality | 0.885 | 9 | Valid |

As shown in Table 2, all indicators have Cronbach's Alpha values greater than 0.6. Therefore, each variable in this study is declared reliable.

### *G. Outlier Test*

In this study, the potential for outliers from the respondents' results was mitigated by the choice of the measurement scale precisely because most of the questions in this research used the 5-point Likert scale. To ensure the integrity of the data, an outlier analysis was conducted. The method for detecting outliers is using Z-scores, which are Standardized scores that indicate how many standard deviations an observation is from the mean. After thorough analysis using these SPSS tools, no outliers were detected in the dataset.

### *H. F-Test and T-Test*

The F-test conducted in this study yielded a p-value of 0.000, indicating a highly statistically significant result. The calculated F-statistic was 32.622, which aligns with the expected critical F-value of 30 from the F-distribution table. This solid statistical evidence suggests that all the independent variables included in the analysis significantly impact the dependent variable being examined in this study. The model also included a T-test to examine the significance of the coefficients. The findings indicate that the significance levels for software testing complexity and approach are 0.000. However, the significance level for software testing knowledge is slightly different at 0.003. These findings indicate that software testing knowledge, software testing approach, and software testing complexity all have a statistically significant impact on the dependent variable, software quality.

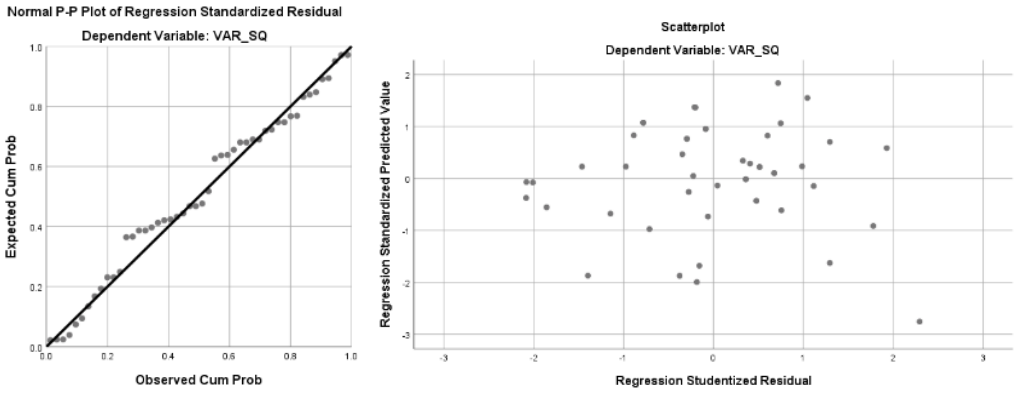
### *I. R-squared*

The R-squared value of 0.690 indicates that 69% of the variance in software quality is explained by the independent variables: software testing knowledge, approach, and complexity. This highlights their significant role in determining software quality. However, the remaining 31% of the variance is due to factors not included in this study, such as team collaboration, project management practices, resource allocation, or external influences like market conditions and

user feedback. Further research is needed to investigate these factors for a more comprehensive understanding of software quality dynamics.

*J. Normality Test and Heteroscedacisty Test*

The normality test results indicate that the data is normally distributed, as the probability plot shows little deviation from the reference diagonal line, as shown in Figure 10a. This suggests the data follows a normal distribution, making it suitable for further statistical analysis. The alignment of the data points with the diagonal line in the probability plot confirms that any deviations from normality are minimal, supporting the assumption of normality required for many statistical models used in this study




(a) (b)

Figure 10. (a) Normality Test Result (b) Heteroscedasticity Test Result

The heteroscedasticity test results reveal no evidence of heteroscedasticity, as shown in Figure 10b. This is supported by the scatterplot, where the data points are evenly dispersed without forming distinct patterns (wavy, expanding, then narrowing). The uniform distribution of the points indicates that the variance of the residuals is consistent, satisfying a key assumption for reliable regression analysis

*K. Multicollinearity Test*

The multicollinearity test results reveal no significant issues among the independent variables: software testing complexity, approach, and knowledge. Software testing complexity has a VIF of 1.806 and a tolerance of 0.554, indicating no strong correlation with other variables. The software testing approach shows a VIF of 1.768 and a tolerance of 0.566, while software testing knowledge has the lowest VIF (1.542) and highest tolerance (0.649), confirming low multicollinearity. With all VIF values below 5 and tolerance values above 0.2, multicollinearity is not a concern, ensuring the model's accuracy and validity.

*L. Interpretation of Hypothesis Test Results*

The first hypothesis demonstrates that Software Testing Knowledge positively affects Software Quality in Batam software houses. This finding indicates that as user theoretical and practical knowledge of software testing increases, it enhances various aspects of software quality. Moreover, this result reinforces prior findings [9], which are positioned in the middle and more to the right side of the scale. This highlights the importance of continuous learning and company development in the field of software testing, suggesting that the company must invest in the knowledge and skills of software testing, which can lead to significant improvements in the quality of software products developed in Batam's

The second hypothesis indicates that the Software Testing Approach positively impacts Software Quality in Batam software houses. Survey results suggest that aligning testing with higher-level goals (Levels 3 and 4) improves software quality by helping organizations develop more effective strategies. This study supports prior research [5], confirming that unit testing and integration testing are the most widely used levels. Unit testing is effective in identifying defects early in development, while integration testing ensures seamless component interaction. System and Acceptance Testing received high ratings for validating overall functionality and meeting user requirements. However, Module Testing received moderate ratings, highlighting areas for improvement and the need for structured testing guidelines. Most respondents emphasized the importance of software testing in enhancing quality. White-box testing emerged as the most frequently used method, valued for its ability to detect defects early and improve reliability. In contrast, black-box and grey-box testing received mixed evaluations, suggesting the need for refinement in their application. These findings challenge prior research, which positions black-box testing as the most common method, and reveal a context-specific preference for white-box testing in Batam's software houses. This preference underscores the practical value of white-box testing in addressing early-stage defects and ensuring better software outcomes. The study suggests that software practitioners prioritize white-box testing while complementing it with black-box methods for a more comprehensive quality assurance strategy.

The third hypothesis demonstrates that complexity significantly impacts software quality in Batam software houses. This variable encompasses two key indicators: the software testing process's complexity and the users' challenges. Both indicators prove to be influential factors affecting software quality. As the testing process becomes more complex, it often involves multiple testing gates, stages, or criteria, serving as comprehensive parameters for ensuring successful software development. On the other hand, the challenges users encounter during testing play a crucial role. Rather than being solely obstacles, these challenges serve as opportunities for gaining valuable experience and refining the testing process. They indicate that the software testing is thorough and rigorous, enhancing the final product's overall quality. The complexity of testing procedures and the user-faced challenges are positive indicators of a robust and effective software testing process, ultimately leading to improved software quality. This study reinforces prior findings [5], as some difficulties occur during testing, impacting the software testing activity. Practitioners are encouraged not to avoid complex testing processes but to view them as opportunities to identify deep-seated issues that can enhance the overall quality of the software.

This study demonstrates that software testing knowledge, appropriate approaches, and effective complexity management collectively improve software quality. For instance, teams with a deep understanding of specific testing approaches are better equipped to address complexity challenges, ultimately producing more reliable, secure, and user-oriented products. Collectively, these independent variables influence the dependent variable, offering more profound insights into the dynamics studied. For example, the combined effect of testing approaches and complexity management on software security provides a more holistic view of how software quality is enhanced. Future research could further explore how these variables interact to impact specific dimensions of software quality, such as performance efficiency or compatibility, offering more targeted insights for the industry.

## IV. CONCLUSION

In conclusion, this study aimed to evaluate the impact of software testing practices on software quality. The results revealed that software testing knowledge, approach, and complexity significantly influence software quality. Software testing knowledge was found to directly enhance software quality by enabling testers to effectively identify and resolve issues, with a significant p-value (0.003) showing that increased knowledge improves defect detection and resolution. The software testing approach, including methods like unit, system, and acceptance testing, ensures comprehensive coverage and addresses potential issues at different stages. Unit testing detects early defects, system testing checks component interactions, and acceptance testing verifies user requirements. The preference for white-box testing further emphasizes its role in improving software quality by focusing on internal structures. Additionally, testing complexity helps in identifying edge cases and reducing undetected errors, contributing to more robust software. The study also identified areas for further exploration, such as the frequency of testing cycles, integration of testing tools, and emerging technologies in testing, which were not covered in the current model but could significantly enhance testing effectiveness. Future research should explore these aspects to build a more comprehensive understanding of their impact on software quality. Finally, as the data was limited to a single geographic location, the findings may only generalize across some contexts. Expanding research to more diverse settings would help validate and broaden the applicability of these insights.

## REFERENCE

[1] S. O. Barraood, H. Mohd, F. Baharom, and A. Almogahed, "Verifying Agile Black-Box Test Case Quality Measurements: Expert Review," *IEEE Access*, vol. 11, pp. 106987–107003, 2023.

[2] H. Sama, "School Operational Effectiveness Analysis Web Based Event System with Dynamic Systems Development," *J. Inf. Technol. Educ. Res.*, vol. 3, no. 1, pp. 76–84, 2019.

[3] P. N. Mata, J. M. Martins, and J. C. Ferreira, "New Software Product Development: Bibliometric Analysis," *J. Knowl. Econ.*, pp. 1–24, 2024.

[4] T. Agrawal, G. S. Walia, and V. K. Anu, "Development of a Software Design Error Taxonomy: A Systematic Literature Review," *SN Comput. Sci.*, vol. 5, no. 5, 2024.

[5] T. Maxime Carlos and M. N. Ibrahim, "Practices in Software Testing in Cameroon Challenges and Perspectives," *Electron. J. Inf. Syst. Dev. Ctries.*, vol. 87, no. 3, pp. 1–17, 2021.

[6] V. Vukovic, J. Djurkovic, M. Sakal, and L. Rakovic, "An Empirical Investigation of Software Testing Methods and Techniques in the Province of Vojvodina," *Teh. Vjesn.*, vol. 27, no. 3, pp. 687–696, 2020.

[7] Kusum, P. Talwar, A. Puri, and G. Kumar, "Overview of Software Testing," *Glob. J. Eng. Technol. Adv.*, vol. 19, no. 1, pp. 104–112, 2024.

[8] R. P. Abdi, "Asessing the Impacts of Paradiplomacy on Batam-Singapore Cooperation: A Case Study in Tourism, Economic Growth, and Infrastructure Development," *J. Parad. City Networks*, vol. 2, no. 1, pp. 12–20, 2023.

[9] Y. Wang, S. Demeyer, K. Wiklund, S. Eldh, and T. Kairi, "Software Test Automation Maturity - A Survey of the State of the Practice," in *arXiv preprint arXiv:2004.09210.*, 2020.

[10] A. Pizzini, R. Bortolo Vieira, R. Deda Gomes, G. Santos, A. Malucelli, and S. Reinehr, "Software Quality Practices in Growing Startups," in *SBQS '21: Proceedings of the XX Brazilian Symposium on Software Quality*, 2021, no. 13, pp. 1–10.

[11] B. Latif and T. Rana, "A Preliminary Survey on Software Testing Practices in Khyber PakhtunKhwa Region of Pakistan," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 28, no. 1, pp. 575 – 589, 2020.

[12] L. Neves, O. Campos, R. Santos, C. Magalhaes, I. Santos, and R. de S. Santos, "Elevating Software Quality in Agile Environments: The Role of Testing Professionals in Unit Testing," in *2024 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2024, pp. 293–296.

[13] N. U. Eisty and J. C. Carver, "Testing Research Software: A Survey," *Empir. Softw. Eng.*, vol. 27, no. 6, pp. 1–32, 2022.
[14] S. M. Melo, V. X. S. Moreira, L. N. Paschoal, and S. R. S. Souza, "Testing Education : A Survey on a Global Scale," in *34th Brazilian Symposium on Software Engineering (SBES '20)*, 2020, pp. 554–563.
[15] N. Sultana, M. Syeed, and K. Fatema, "An Empirical Investigation on Quality Assurance Practices in Software Industries: Bangladesh Perspective," *Int. J. Softw. Eng. Comput. Syst.*, vol. 6, no. 2, pp. 1–10, 2020.
[16] J. Skalka and M. Drlik, "Development of Automatic Source Code Evaluation Tests Using Grey-Box Methods: A Programming Education Case Study," *IEEE Access*, vol. 11, pp. 106772–106792, 2023.
[17] M. A. Umar and C. Zhanfang, "A Comparative Study of Dynamic Software Testing Techniques," *Int. J. Adv. Netw. Appl.*, vol. 12, no. 03, pp. 4575–4584, 2020.
[18] V. Garousi, M. Felderer, M. Kuhrmann, K. Herkiloğlu, and S. Eldh, "Exploring the Industry's Challenges in Software Testing: An Empirical Study," *J. Softw. Evol. Process*, vol. 32, no. 8, pp. 1–28, 2020.
[19] B. Binboga and C. A. Gumussoy, "Factors Affecting Agile Software Project Success," in *IEEE Access*, 2024, vol. 12, pp. 95613–95633.
[20] L. Martins, V. Brito, D. Feitosa, L. Rocha, H. Costa, and I. Machado, "From Blackboard to the Office : A Look Into How Practitioners Perceive Software Testing Education," in *EASE '21: Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering*, 2021, pp. 211–220.
[21] R. B. Sitepu, M. S. Ilham, T. Handriana, and P. Yulianti, "Effect of Information Technology Capital: Technology Infrastructure, Database, Software, and Brainware Toward Optimize the Use of Information Technology (Case Study: UIN Sunan Ampel Of Surabaya)," *Libr. Philos. Pract.*, 2021.
[22] M. Siahaan and N. Legowo, "The Citizens Acceptance Factors of Transportation Application Online in Batam: An Adaptation of the UTAUT2 Model and Information System Success Model," *J. Theor. Appl. Inf. Technol.*, vol. 97, no. 6, pp. 1666–1676, 2019.