International Journal of Engineering Research and Development
e- ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 22, Issue 2 (February 2026), PP 33-40

Quality and Quality Concepts in Software-Industries

Roopa B. Math[1], Prasadu Peddi[2]
[1]*Research Scholar, Department of Computer Science, Sunrise University, Alwar,*
[2]*Research Supervisor, Department of Computer Science, Sunrise University, Alwar*

Abstract

The idea of quality in software industries has become a crucial factor in determining an organization's success in the quickly changing digital age. The aspects of software quality, quality management techniques, measurement strategies, and the difficulties businesses encounter while putting quality-centric procedures into place are all covered in this essay. Global quality standards and new developments that influence software development quality requirements are also covered. The objective is to offer a cogent synthesis of theoretical frameworks and useful insights for improving software services and product quality.

Date of Submission: 24-01-2026 Date of acceptance: 06-02-2026

I. INTRODUCTION

In the past, system stability and error-free code were the main goals of software quality. The emphasis switched to a more comprehensive, all-encompassing view of quality as software systems became more complicated and consumers became more demanding. These days, software quality encompasses a variety of factors that impact the overall user experience and operational effectiveness of software programs, including security, scalability, usability, and even environmental sustainability. It is impossible to exaggerate the significance of software quality. Users demand software to operate smoothly, be user-friendly, and offer a dependable experience due to the quick expansion of the digital economy. Low-quality software can lead to expensive mistakes, unhappy customers, legal troubles, and reputational damage. Software malfunctions can have dire repercussions including financial disaster in vital industries like healthcare, banking, and transportation. As a result, maintaining good software quality is both a fundamental business strategy and an operational requirement.

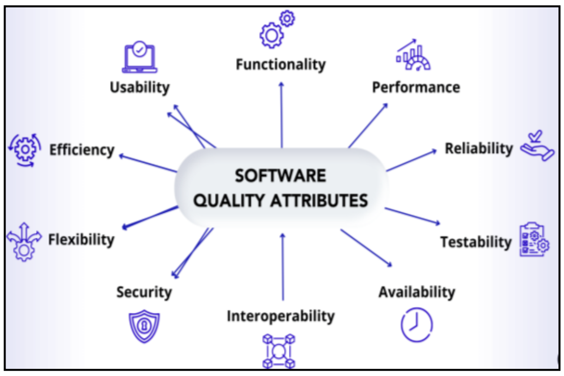
Compared to other technical fields, software quality is more difficult to define because it includes not just error-free code but also dependability, usability, efficiency, and maintainability all of which frequently change over the course of the software lifetime. One of the most important components of software engineering is software quality, which is recognized as a strategic advantage for businesses in the software sector. The degree to which a software product satisfies user needs, complies with specifications, and operates dependably in designated contexts is referred to as software quality. In the context of software development, quality is the extent to which a software product satisfies customer-defined functional requirements, satisfies non-functional performance criteria, and conforms to industry standards and best practices. Low-quality software causes malfunctions, security lapses, and expensive upkeep. Therefore, it is essential for stakeholders in the software industry to comprehend quality and its underlying ideas.

In the context of contemporary software companies, this study investigates the changing notion of software quality. It addresses the significance of quality from the viewpoints of various stakeholders, such as developers, users, and companies. In order to enhance the consistency, dependability, and performance of software products, the article also explores quality management frameworks, quality measurement methods, and tools. It also draws attention to the difficulties businesses encounter when putting into practice efficient quality procedures and the new developments that will influence software quality management in the future.

Giants like Tata Consultancy Services (TCS), Infosys, HCL Technologies, and Wipro consistently rank among India's top software companies based on market capitalization or revenue. These companies are followed by Tech Mahindra, LTI-Mindtree, Cognizant (India operations), Accenture, Persistent Systems, and Mphasis. Global companies like Oracle, IBM, and Microsoft also have significant Indian footprints and concentrate on digital transformation, cloud, AI, and enterprise solutions. These businesses drive innovation in cyber-security, cloud computing, and artificial intelligence by specializing in IT services, consulting, and outsourcing. Bengaluru, Pune, Hyderabad, and NCR are important hubs.

II. DEFINING SOFTWARE QUALITY

Software quality is defined by a collection of attributes that indicate how effectively software satisfies both functional and non-functional requirements, according to ISO/IEC 25010. Quality can be accessed from the viewpoints of the system, the developer, and the consumer. Software Quality Attributes are characteristics that make it easier for experts in software testing to gauge a product's performance. They are essential in assisting software architects in ensuring that a program will function according to the client's demands. In the early phases of software testing, recommend doing tests that evaluate the intended functionality of a system.



Software quality is a complex concept that changes over time as user expectations and the software industry both expand. The degree to which a software product fulfills user needs, meets or surpasses criteria, and successfully, dependably, and efficiently carries out its intended purpose is referred to as software quality. However, many viewpoints on what defines software quality have emerged due to the complexity of software systems and the diverse needs of various stakeholders such as developers, consumers, and enterprises. Various aspects of software quality are listed in Table 1.

Fig. 1 Software Quality Attributes
(Source: https://codoid.com/software-testing/the-basics-of-software-quality-attributes/)

Table 1. Important aspects of software quality

1.	Functionality	Accuracy, suitability, and compliance with specifications
2.	Reliability	Ability to maintain performance under defined conditions
3.	Usability	Ease of use and user understanding
4.	Efficiency	Resource optimization and performance
5.	Maintainability	Ease of modification and correction
6.	Portability	Adaptability to different environments
7.	Security	Protecting data and preventing unauthorized access or misuse
8.	Compatibility	Ability to work well with other software or hardware

III. MODELS FOR DEFINING SOFTWARE QUALITY

Functionality, dependability, usability, efficiency, maintainability, portability, security, and compatibility are just a few of the many attributes that make up the concept of software quality. These attributes, can be customized to particular sectors, applications, or user needs, serve as the basis for assessing the quality of software. A thorough grasp of these elements and the application of best practices are necessary to produce high-quality software that satisfies both technical and user-driven quality criteria.

3.1. The Software Quality Assurance (SQA) Model

Software quality assurance aims to guarantee that software satisfies certain requirements and standards. Throughout the software development lifecycle, SQA uses process audits, inspections, and verification techniques to find problems early and enhance the product before it is released.

3.2. McCall's Quality Model

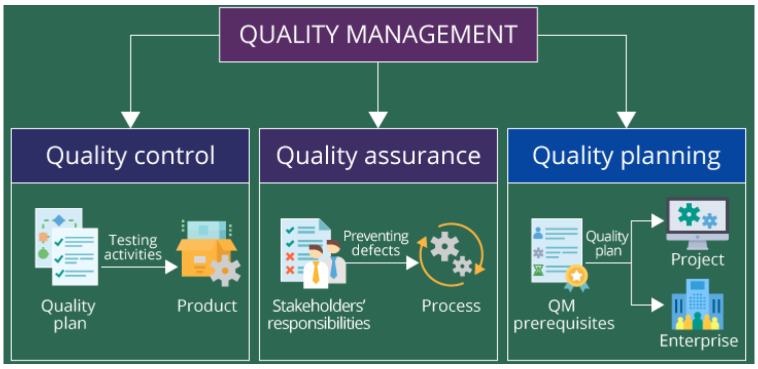
The 11 components of McCall's model, which was created in the 1970s, are divided into three main criteria: product operation, product revision, and product transition. These factors, which closely correspond with the previously described dimensions, highlight elements like dependability, efficiency, maintainability, and mobility.

3.3. Boehm's Model

Boehm's quality model focuses on recognizing and comprehending the essential characteristics such as accuracy, efficiency, integrity, and usability that contribute to software quality. Boehm has put forth the idea of a cost-to-quality link, which holds that investing in quality early in the software lifecycle lowers overall costs because there are fewer faults and subsequent maintenance expenses.

IV. QUALITY ASSURANCE VS QUALITY CONTROL

Meeting the demands, expectations, and requirements of a user who is requesting software or an app is known as software quality (SQ). The goal of SQ is to ensure that the final product is free of any potential flaws, mistakes, or faults. During the development process, certain predetermined standards must be adhered to in order to achieve software quality. Separate the terms "quality" and "assurance" to gain a quick understanding. In organizational terms, assurance refers to enterprise management that guarantees optimal performance of the generated product. Become confident that the result will be positive beforehand. The assurance is a guarantee that the product will work flawlessly and live up to the customer's expectations. To put it briefly, quality assurance (QA) refers to all activities focused on developing procedures and standards for confirming that software satisfies quality requirements, refer Fig. 2.



Quality Control, commonly referred to as QC, is a software testing procedure that guarantees the quality of goods or services complies with the quality management system and technique to confirm the product's quality. Statistical quality control is the process of applying statistical methods and tools to the final software product. In contrast to quality assurance, quality control examines the finished goods' quality instead of the process.

Fig.2 QA Vs QC (Sourcs: https://www.scnsoft.com/software-testing/quality-management-optimization)

Activities that are product-oriented and result-oriented make up quality control, or QC. The goal of quality control is to give businesses the assurance they need to create applications that meet both customer and quality standards. When a quality control procedure finds a problem with the finished product, it should ideally be fixed before the final consumer receives it. In other words, quality control (QC) is the process of making sure a product or service satisfies both traditional quality standards and customer demands. Pre-defining procedures, prompt quality audits, statistical process control, and other quality management-focused techniques are examples of such operations.

Table 2. Quality Assurance Vs Quality Control

Quality Assurance (QA)	Quality Control (QC)
A proactive process	A Reactive process
Take measures in advance (planning)	Performed when flaws are identified
Concerned with defect prevention	Concerned with detecting the flaws
Ensures the product is of greater quality	Validates the quality of the product
Makes sure the developers are doing the right	Makes sure the end result is acceptable
perform the Quality audit, as a part of QA	Testing/analysis is the main process of QC
Create the deliverables	Verify the deliverables
Entire software development team is responsible	Only the testing team is responsible
Enable the software's quality	Verify the software's quality
Involves test planning and execution	Includes creating and maintaining test reports
Quality assurance is a plan	Quality control is to check
It examines the plan to see if it was effective in avoiding any potential flaws	It looks for flaws in the product and try to fix them while it's being made
The goal of QA is to stop software flaws	The goal of QC is to find flaws in any software

This brings an end to our examination of the most controversial topic: the difference between quality control and assurance. The importance of these two jobs to the software development life cycle has been demonstrated in this blog. When we have both QA and QC procedures in place, we can ensure that the product is developed in compliance with customer specifications and with thorough QA testing and analysis. Obtaining results from consistently met QA standards and processes is the aim of QA. It ensures that everything produced is safe, efficient, and of the best quality for customers while avoiding any defects.

V. QUALITY MANAGEMENT FRAMEWORK IN SOFTWARE DEVELOPMENT

By employing structured techniques known as quality management frameworks (QMF), organizations can preserve and improve the quality of their software processes and products. These frameworks provide processes, best practices, and standards that companies can utilize to create dependable, superior software. Furthermore, they ensure that quality is not a one-time effort but is incorporated into every phase of the software development lifecycle (SDLC). A quality management framework in software development typically incorporates quality assurance (QA) methodologies, process improvement models, measurement tools, and standards to ensure that the final software product meets or exceeds user expectations and regulatory requirements. ISO standards, CMMI, TQM, and more modern agile approaches that emphasize quality as a continuous, team-based endeavor are the most widely used quality management frameworks in the software business.

4.1. ISO/IEC Standards

A set of standards created by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) offer a widely accepted framework for software quality management. These standards concentrate on making sure that software is built using repeatable, effective, and efficient procedures and satisfies client needs.

4.1.1 ISO/IEC 9001: Quality Management Systems

The most popular quality management standard in the world is ISO/IEC 9001. It guarantees that they continuously provide goods or services that satisfy client and legal standards and offers a methodical approach to managing organizational operations.When it comes to software development, ISO/IEC 9001 highlights:

- Process management: Every step of the software development process, from planning to delivery, needs to be precisely defined, recorded, and standardized. This lessens product quality variability and increases uniformity.
- Customer focus: The framework emphasizes that software design and development should be guided by the needs and expectations of the user. In order to continuously enhance the product, it makes sure that user feedback is incorporated into the development cycle.
- Continuous improvement: ISO/IEC 9001 encourages a culture of continuous improvement, which means that even after software is deployed, its quality is continuously evaluated and enhancements are made as part of the operational philosophy of the company.

For instance, a business that develops financial software might use ISO/IEC-9001 to standardize their development process. This would guarantee that every step of the process from requirement gathering to post-deployment maintenance is precisely defined and consistently adhered to, resulting in improved quality and client satisfaction.

4.1.2. ISO/IEC 25010: Software Product Quality Model

The quality standards for software products are outlined in ISO/IEC 25010. It focuses on measuring and evaluating software quality in a number of important areas, such as functionality, dependability, usability, efficiency, and maintainability (as covered in previous sections). This standard offers a framework for assessing a software product's overall quality using these several characteristics. By using ISO/IEC 25010, organizations can assess the quality of their software at various phases of development, guaranteeing that every aspect of quality is consistently tracked and enhanced.

4. 2. Capability Maturity Model Integration (CMMI)

One of the most popular frameworks for process optimization in software development is the Capability Maturity Model Integration (CMMI). CMMI offers principles for improving software development processes over time and assists organizations in determining how mature their processes are. By concentrating on a number of process areas that assist companies in producing high-quality software, CMMI is intended to promote continuous improvement.

4.2.1. CMMI Maturity Levels

Each of the five maturity levels in CMMI represents a more sophisticated and efficient method of software development. An organization's ability to consistently produce high-quality software increases with its level of maturity.

1. Level 1: Initial: Ad hoc and unstructured procedures are common at this level. In certain cases, organizations might produce high-quality software, but this is not always the case, and the outcomes could differ significantly.
2. Level 2: Managed: To guarantee that projects are successfully planned, carried out, and monitored, fundamental project management procedures are adopted. Although it is still reactive, quality control is more organized.

3. Level 3: Defined: At this stage, companies have well defined procedures that are recorded and adhered to throughout the whole company. Quality is a planned component of every stage of the SDLC, and software development is consistent.
4. Level 4: Quantitatively Managed: Processes are tracked and managed using quantitative metrics. Software quality data is gathered, and statistical analysis is utilized to forecast and enhance results, guaranteeing that quality is continuously attained.
5. Level 5: Optimizing: The organizational culture incorporates constant improvement. Process optimization is the main focus, and feedback loops are employed to improve both product quality and development procedures. Regular innovations and enhancements are done to increase the performance and quality of software.

As an illustration, a software business moving through the CMMI levels might start with a reactive approach to software testing (Level 1) and eventually develop into a highly sophisticated procedure that employs predictive analytics to guarantee quality at every stage of development (Level 4).

4. 3. Total Quality Management (TQM)

A thorough and organized method for raising quality across the board in a business is called total quality management, or TQM. Every individual in the company, from senior management to front-line workers, is involved in TQM's ongoing process, service, and product improvement. TQM places a strong emphasis on the following ideas in software development:

- Customer satisfaction: By producing software that is dependable, easy to use, and satisfies functional requirements, TQM aims to both meet and beyond customer expectations.
- Employee involvement: Including every team member in quality improvement projects guarantees that quality is deeply embedded in the company's culture. Quality is maintained at every level by developers, testers, managers, and even support personnel.
- Process-centered approach: TQM mandates that businesses concentrate on enhancing the procedures that result in software development. Teams can enhance overall software quality by detecting inefficiencies or bottlenecks in the SDLC.
- Continuous improvement: TQM encourages firms to adopt new technologies, improve processes, and innovate in order to increase quality by fostering a culture of continuous learning and adaptation.

For instance, a business creating a mobile application might apply TQM concepts to involve every member of the team in enhancing usability. Updates are released iteratively depending on customer satisfaction metrics after user feedback is routinely collected and examined.

4.4 Agile Quality Management Frameworks

Agile approaches place more emphasis on flexibility, teamwork, and quick iterations than traditional frameworks like ISO/IEC, CMMI, and TQM, which are concentrated on process standardization and gradual development. Agile emphasizes providing software in manageable chunks and continuously improving the product in response to user feedback. Agile teams frequently use Test-Driven Development (TDD), Continuous Integration (CI), and Agile Testing techniques to continuously and iteratively guarantee product quality.

4.4.1. Agile Principles for Quality Management

- Customer collaboration: Ongoing customer feedback ensures that the program satisfies user needs.
- Adapting to change: Agile teams ensure that software stays high-quality and relevant throughout its lifecycle by swiftly adjusting to shifting needs or market conditions.
- Iterative delivery: Agile teams produce functioning, tested software on a regular basis through brief development cycles (sprints), guaranteeing that quality is continuously evaluated and improved.

4.5. Lean Software Development

Another contemporary approach to quality management that draws inspiration from lean manufacturing is lean software development. It focuses on cutting waste, increasing productivity, and providing value to customers faster. The main goals of lean quality management are:

- Value stream mapping: locating and eliminating activities in the development process that don't contribute value.
- Reducing bottlenecks: Improving flow by fixing inefficiencies in the process.
- Empowering teams: Encouraging decentralized decision-making to allow development teams to quickly address quality concerns without waiting for management approval.

VI. . EMERGING TRENDS AND TECHNOLOGIES SHAPING SOFTWARE QUALITY

Due to changes in industry processes and technological breakthroughs, the software quality landscape is changing dramatically. The way that software quality is seen and attained is changing due to a number of new trends:

1. DevOps and Continuous Integration/Continuous Delivery (CI/CD): Using DevOps to integrate development and operations is one of the biggest trends in software quality today. This change places a strong emphasis on cooperation, automation, and ongoing feedback between the operations and development teams. Automated testing is incorporated into all phases of the software development lifecycle with CI/CD pipelines, facilitating the prompt detection and correction of flaws. This results in quicker delivery cycles without sacrificing quality. Developers and operations teams have a close working connection that reduces silos and increases accountability by fostering a culture of shared responsibility for quality.
2. Artificial Intelligence and Machine Learning in Testing: The emergence of AI-driven testing is a revolutionary development in software quality assurance. Automated tests can adjust to new changes in the codebase without requiring frequent manual updates thanks to AI and machine learning technologies. AI is able to prioritize testing efforts and forecast which software components are most likely to have flaws by analyzing past data. This speeds up and improves the accuracy of defect identification while drastically cutting down on test planning time.
3. Security-Driven Development (DevSecOps): Security is now a crucial component of the quality process due to the growing sophistication of cyber threats. By including security procedures into the DevOps pipeline, DevSecOps makes sure that security flaws are fixed early in the development process rather than after release. This proactive strategy guarantees that the software satisfies regulatory compliance requirements and lowers the possibility of expensive security breaches.
4. Cloud-Native and Microservices Architectures: Software quality faces both possibilities and problems as a result of the migration to cloud-native and microservices architectures. On the one hand, these structures facilitate quicker deployment, flexibility, and scalability. However, they also create new difficulties for quality assurance, monitoring, and testing across many environments and services. To maintain quality in such systems, service-level agreements (SLAs), distributed monitoring tools, and continuous testing are crucial.
5. User-Centered Quality: The notion of software quality has broadened to encompass usability, accessibility, and user happiness as user experience (UX) has taken center stage in software design. Nowadays, businesses are spending more on UX testing and getting user input frequently throughout the development process. This change guarantees that software not only satisfies technical requirements but also provides end users with an easy-to-use and entertaining experience.

VII.CHALLENGES IN ACHIEVING SOFTWARE QUALITY

Even though its significance is well acknowledged, many businesses still struggle to achieve software quality. The following difficulties are especially common:

1. The intricacy of contemporary software systems Software systems nowadays are bigger and more complex than in the past. The complexity of testing and guaranteeing quality rises dramatically with the integration of many platforms, technologies, and outside services. Furthermore, emerging paradigms like distributed systems and microservices architectures introduce further levels of complexity that complicate quality control.
2. Time and Resource Constraints: Software development teams are frequently under pressure to produce new features swiftly in order to stay competitive in today's fast-paced market. Due to the pressure to meet deadlines, testing cycles may be reduced and errors may be ignored, which could result in quality compromises. Because of this, developers and testers have to strike a balance between providing value and giving priority to the most important concerns.
3. Changing Stakeholder Expectations and Requirements: The software industry is well known for its constantly changing requirements. The previously tested program may become out-of-date or unreliable if new features are introduced or user needs change. It might be difficult to keep track of these modifications and make sure the program keeps up with user expectations. This becomes even more difficult in agile development since quick iterations and constant feedback necessitate constant re-evaluation of quality metrics.

4. Lack of Skilled Workforce: To produce high-quality software, experts who comprehend not just coding but also quality assurance concepts, testing procedures, and best practices are needed. Building teams with sufficient knowledge in these areas is still a challenge for many firms, particularly in specialized disciplines like automation testing, performance optimization, and security testing.

VIII. CONCLUSION

As the digital landscape changes, software quality plays an increasingly important role in the software industry. Software quality affects customer satisfaction, long-term commercial outcomes, and an organization's reputation in addition to the immediate user experience. Strong software quality frameworks, processes, and standards are becoming more and more important as companies depend more and more on software to run their operations and provide services. In software development, quality is an ongoing activity that needs to be integrated into the entire development lifecycle rather than being a one-time accomplishment. In addition to using contemporary tools and technology, this process calls for an organizational culture shift that prioritizes quality across the entire process, from requirement collection to post-deployment maintenance. Delivering products that meet or above user expectations requires quality assurance techniques including automated defect tracking, continuous testing, and integration of best practices.

In conclusion, software quality is a crucial pillar of the software industry that demands constant attention, creativity, and flexibility. Software companies may overcome present obstacles and set themselves up for future success with the correct combination of approaches, resources, and a culture of quality. Making sure software is of high quality not only results in happy clients and corporate expansion, but it also advances the technology sector as a whole.

IX. THE FUTURE OF SOFTWARE QUALITY

Automation, artificial intelligence, and cloud technologies will surely have an impact on software quality in the future. The need for automated quality checks, real-time monitoring, and AI-driven defect prediction will increase with the complexity of software systems. In order to guarantee quality at every stage, teams will also need to embrace more flexible, cooperative methods as the distinction between development and operations becomes increasingly hazy. In conclusion, producing high-quality software is a complex task that calls for a blend of cutting-edge technology, efficient management techniques, and a continuous improvement culture. Software development companies may stay ahead of the curve and produce solutions that not only meet but beyond consumer expectations by utilizing quality frameworks, embracing new trends, and cultivating a quality-focused mindset.

Producing software of the highest caliber is not easy. Organizations still struggle to maintain quality despite improvements in tools, processes, and standards because of things like the speed at which technology is developing, the need for prompt delivery, and the increasing complexity of software systems. Organizations must implement scalable, adaptable quality management systems, make training and skill development investments, and stay up to date with new trends like DevOps and artificial intelligence in order to meet these difficulties.

REFERENCES

[1] D. M. B. Paiva, A. P. Freire, and R. P. de Mattos Fortes, "Accessibility and Software Engineering Processes: A Systematic Literature Review," J. Syst. Softw., vol. 171, p. 110819, Jan. 2021, doi: 10.1016/j.jss.2020.110819.

[2] H. Femmer, D. Méndez Fernández, S. Wagner, and S. Eder, "Rapid quality assurance with requirements Smells," J. Syst. Softw., vol. 123, pp. 190–213, Jan. 2017, doi: 10.1016/j.jss.2016.02.047.

[3] S. Khalid, U. Rasheed, M. Abbas, A Model Driven Framework for Standardizing Requirement Elicitation by Quantifying Software Quality factor In 2021 International Conference on Innovative Computing (ICIC), Nov (2021), pp. 1-6, 10.1109/ICIC53490.2021.9693054

[4] H.F. Hofmann, F. Lehner, Requirements engineering as a success factor in software projects, IEEE Softw, 18 (4) (Jul. 2001), pp. 58-66, 10.1109/MS.2001.936219

[5] Association for Computing Machinery-Digital Library, "Requirement Elicitation Framework for Child Learning Application - A Research Plan," in ICSIM 2019: Proceedings of the 2nd International Conference on Software Engineering and Information Management, Association for Computing Machinery, 2019, pp. 129–133. doi: https://doi.org/10.1145/3305160.3305195.

[6] J. Medeiros, A. Vasconcelos, C. Silva, M. Goulão, Quality of software requirements specification in agile projects: a cross-case analysis of six companies, J Syst Softw, 142 (Aug. 2018), pp. 171-194, 10.1016/j.jss.2018.04.064

[7] S. W. Ali, Q. A. Ahmed, and I. Shafi, "Process to enhance the quality of software requirement specification document," in 2018 International Conference on Engineering and Emerging Technologies (ICEET), Feb. 2018, pp. 1–7. doi: 10.1109/ICEET1.2018.8338619.

[8] M. H. Osman and M. F. Zaharin, "Ambiguous Software Requirement Specification Detection: An Automated Approach," in Proceedings of the 5th International Workshop on Requirements Engineering and Testing, Gothenburg Sweden: ACM, Jun. 2018, pp. 33–40. doi: 10.1145/3195538.3195545.

[9] S. M. Abbas, K. A. Alam, U. Iqbal, and S. Ajmal, "Quality Factors Enhancement of Requirement Engineering: A Systematic

Literature Review," in 2019 International Conference on Frontiers of Information Technology (FIT), Dec. 2019, pp. 13–135. doi: 10.1109/FIT47737.2019.00013.

[10] Y. Xu, W. Ge, X. Li, Z. Feng, X. Xie, and Y. Bai, "A Co-Occurrence Recommendation Model of Software Security Requirement," in 2019 International Symposium on Theoretical Aspects of Software Engineering (TASE), Jul. 2019, pp. 41–48. doi: 10.1109/TASE.2019.00-21.

[11] S. S. A. Bukhari, M. Humayun, S. A. A. Shah, and N. Z. Jhanjhi, "Improving Requirement Engineering Process for Web Application Development," in 2018 12th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS), Nov. 2018, pp. 1–5. doi: 10.1109/MACS.2018.8628422.

[12] S. Jeong, H. Cho, and S. Lee, "Agile requirement traceability matrix," in Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, Gothenburg Sweden: ACM, May 2018, pp. 187–188. doi: 10.1145/3183440.3195089.

[13] I. Zafar, A. Shaheen, A. K. Nazir, B. Maqbool, W. H. Butt, and J. Zeb, "Why Pakistani Software Companies don't use Best Practices for Requirement Engineering Processes," in 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Nov. 2018, pp. 996–999. doi: 10.1109/IEMCON.2018.8614913.

[14] S. Anwer, L. Wen, and Z. Wang, "A Systematic Approach for Identifying Requirement Change Management Challenges: Preliminary Results," in Proceedings of the Evaluation and Assessment on Software Engineering, Copenhagen Denmark: ACM, Apr. 2019, pp. 230–235. doi: 10.1145/3319008.3319031.

[15] F. Cowperthwaite, J. Horkoff, S. Kopczyńska, The Effects of Native Language on Requirements Quality

In 2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS), Sep (2023), pp. 913-917, 10.15439/2023F9537