Vol. 16, No. 1, April 2024

IMPROVE

SSN(e): -/ ISSN(p) : 1979-8342

Software Quality Assurance (SQA) dalam Rekayasa Perangkat Lunak Modern

Muhammad Ibnu Choldun Rachmatullah[1]

[1] *D3 Manajemen Informatika, Universitas Logistik dan Bisnis Internasional*

[1]muhammadibnucholdun@ulbi.ac.id

Abstrak— Software Quality Assurance (SQA) merupakan disiplin yang memastikan proses, produk, dan layanan perangkat lunak memenuhi kebutuhan pemangku kepentingan melalui kombinasi pendekatan proses, standar kualitas, pengukuran, verifikasi–validasi, dan tata kelola perubahan. Dalam periode 2020–2024, SQA mengalami pergeseran paradigma yang kuat: dari model "quality gate" yang cenderung berada di akhir siklus menjadi "quality-as-code" yang tertanam dalam pipeline CI/CD, didorong oleh Agile/DevOps, otomasi pengujian, observabilitas, serta kebutuhan kualitas untuk sistem kompleks seperti microservices dan AI-based software. Artikel review ini merangkum konsep inti SQA, evolusi praktiknya, integrasi dengan standar kualitas (mis. ISO/IEC 25010:2023), tren platform SQA berbasis layanan (SQAaaS), dan temuan-temuan literatur mutakhir terkait faktor sukses DevOps, otomasi pengujian, metrik kualitas, serta tantangan kualitas khusus pada perangkat lunak berbasis AI. Rangkaian pembahasan menekankan keterkaitan antara definisi kualitas, strategi pengendalian proses, pemilihan metrik, dan mekanisme umpan balik yang berkesinambungan agar kualitas dapat dikelola sebagai sistem sosio-teknis, bukan sekadar aktivitas pengujian.

Kata kunci— Software Quality Assurance, kualitas, metrik

Abstract— Software Quality Assurance (SQA) is a discipline that ensures software processes, products, and services meet stakeholder needs through a combination of process approaches, quality standards, measurement, verification–validation, and change governance. In the 2020–2024 period, SQA is undergoing a strong paradigm shift: from a "quality gate" model that tends to be at the end of the lifecycle to "quality-as-code" embedded in CI/CD pipelines, driven by Agile/DevOps, test automation, observability, and the quality requirements for complex systems such as microservices and AI-based software. This review article summarizes the core concepts of SQA, the evolution of its practices, integration with quality standards (e.g., ISO/IEC 25010:2023), trends in service-based SQA (SQAaaS) platforms, and recent literature findings on DevOps success factors, test automation, quality metrics, and specific quality challenges in AI-based software. The series of discussions emphasizes the relationship between the definition of quality, process control strategies, selection of metrics, and continuous feedback mechanisms so that quality can be managed as a socio-technical system, not just a testing activity..

Keywords— *Software Quality Assurance, quality, metrics*

I. PENDAHULUAN

Kualitas perangkat lunak tidak lagi dipahami hanya sebagai "bebas bug", melainkan sebagai kemampuan sistem untuk memberikan nilai dalam konteks penggunaan nyata, mencakup aspek fungsional, kinerja, reliabilitas, keamanan, kemudahan interaksi, serta kemampuan berevolusi ketika kebutuhan berubah. Perubahan ekspektasi pengguna, akselerasi rilis, dan meningkatnya ketergantungan layanan digital menjadikan kualitas bukan sekadar tujuan teknis, melainkan komitmen organisasi yang memengaruhi reputasi, kepatuhan, dan keberlanjutan bisnis. Dalam konteks ini, Software Quality Assurance (SQA) berperan sebagai pendekatan sistematis untuk memastikan kualitas dibangun sejak awal melalui pengendalian proses, bukti pengujian, dan evaluasi berbasis metrik yang dapat diaudit.

Selama bertahun-tahun, praktik SQA sering dipersepsikan identik dengan aktivitas pengujian, terutama pengujian fungsional menjelang rilis. Namun, literatur modern menunjukkan bahwa pengujian hanya salah satu komponen dari SQA. SQA mencakup kebijakan, standar, prosedur, audit internal, manajemen konfigurasi, pengukuran kualitas, serta mekanisme perbaikan berkelanjutan. Pada organisasi yang mengadopsi Agile/DevOps, SQA juga berubah menjadi aktivitas yang melekat pada pipeline pengembangan: kualitas dipantau terus-menerus, risiko diidentifikasi lebih dini, dan kontrol kualitas menjadi otomatis dan dapat direproduksi.

Perubahan besar pada 2020–2024 terjadi karena dua tekanan utama. Pertama, tuntutan time-to-market yang cepat memaksa organisasi untuk mengintegrasikan kontrol kualitas ke dalam CI/CD agar rilis sering tetap aman. Temuan dari systematic literature review tentang critical success factors (CSF) DevOps memperlihatkan bahwa keberhasilan DevOps sangat terkait dengan kemampuan membangun umpan balik cepat, otomasi, serta praktik lintas fungsi yang menstabilkan kualitas dalam ritme rilis yang padat [4]. Kedua, meningkatnya kompleksitas sistem—microservices, cloud-native, dependensi pihak ketiga, dan AI-based software—membuat definisi "kualitas" semakin multi-dimensi dan kontekstual. ISO/IEC 25010:2023 memperbarui model kualitas produk perangkat lunak menjadi sembilan karakteristik yang dapat

menjadi acuan dalam spesifikasi, pengukuran, dan evaluasi kualitas sepanjang siklus hidup [1]. Hal ini memperlihatkan bahwa SQA modern membutuhkan kerangka kualitas yang lebih kaya daripada sekadar daftar bug.

Selain itu, munculnya platform dan layanan kualitas seperti Software Quality Assurance as a Service (SQAaaS) menandai tren industrialisasi SQA dalam skala luas. Bernardo dkk. memperkenalkan konsep SQAaaS yang memfasilitasi asesmen otomatis metrik kualitas dan baseline standar, termasuk integrasi pembuatan pipeline CI/CD untuk pengujian otomatis serta pemberian "digital badges" sebagai insentif pemenuhan baseline kualitas [2]. Dengan demikian, SQA tidak hanya menjadi aktivitas internal tim, tetapi juga dapat menjadi ekosistem layanan yang memperkuat transparansi, reproducibility, dan governance kualitas.

Namun, evolusi SQA juga memunculkan pertanyaan metodologis: bagaimana menilai bukti kualitas secara rigor, bagaimana memilih metrik yang benar-benar merepresentasikan kualitas, dan bagaimana menghindari "metric gaming" atau optimasi semu. Dalam konteks riset software engineering, studi Yang dkk. tentang quality assessment (QA) pada systematic literature reviews menekankan pentingnya prosedur penilaian kualitas bukti yang konsisten dan instrumen QA yang lebih beragam serta rigor, karena kesimpulan sangat bergantung pada kualitas studi yang ditinjau [3]. Walau fokusnya QA pada riset, prinsip dasarnya relevan untuk SQA: kualitas keputusan hanya sebaik kualitas bukti yang dikumpulkan.

Berdasarkan latar tersebut, artikel ini disusun sebagai review terstruktur empat bab untuk: (1) membangun fondasi konseptual SQA dan hubungannya dengan model kualitas; (2) menjelaskan praktik SQA modern dalam Agile/DevOps dan otomasi; (3) membahas pengukuran, platform, serta tantangan mutakhir termasuk AI-based software; dan (4) merumuskan kesimpulan serta arah riset dan implementasi ke depan.

II. Landasan Konseptual SQA dan Evolusi Kerangka Kualitas

Secara prinsip, SQA adalah serangkaian aktivitas terencana dan sistematis untuk memberikan keyakinan (assurance) bahwa proses dan produk perangkat lunak memenuhi persyaratan kualitas. "Assurance" berbeda dari "control": quality control cenderung berfokus pada deteksi cacat pada output, sedangkan quality assurance berfokus pada pencegahan cacat melalui proses yang benar, standar yang jelas, dan mekanisme audit yang dapat menunjukkan bahwa pekerjaan dilakukan sesuai kebijakan. Karena perangkat lunak adalah artefak yang mudah berubah, SQA harus memadukan kontrol proses dan kontrol evolusi, sehingga kualitas bukan kondisi statis melainkan kemampuan sistem mempertahankan performa dan reliabilitas di tengah perubahan.

Kerangka kualitas menjadi pusat SQA karena "kualitas" hanya dapat dikelola jika didefinisikan. ISO/IEC 25010:2023 menawarkan model kualitas produk yang dapat diterapkan pada produk TIK dan perangkat lunak, dan ditujukan untuk spesifikasi kebutuhan, pengukuran, dan evaluasi [1]. Dalam praktik SQA, model ini dapat digunakan untuk menerjemahkan tuntutan stakeholder menjadi karakteristik dan sub-karakteristik yang terukur. Yang penting, standar tersebut menegaskan model kualitas dapat dimanfaatkan oleh berbagai pihak—pengembang, pembeli, staf QA/QC, hingga evaluator independen—dan relevan sepanjang lifecycle, mulai dari elisitasi kebutuhan hingga kriteria penerimaan [1]. Artinya, SQA modern idealnya mengikat definisi kualitas dengan artefak operasional: user story, acceptance criteria, test plan, non-functional requirements, dan metrik pemantauan.

Evolusi Agile dan DevOps mengubah lokasi dan ritme SQA. Pada pendekatan tradisional, SQA sering dipusatkan pada fase verifikasi akhir, mengikuti konsep "test after build". Agile mendorong iterasi pendek, sehingga SQA harus memfasilitasi feedback cepat, definisi "done" yang konsisten, dan pengujian berlapis sejak awal sprint. DevOps memperluasnya dengan memasukkan operasi (monitoring, incident response) sebagai bagian dari quality loop. Studi CSF DevOps menunjukkan keberhasilan DevOps terkait faktor teknis, organisasi, serta sosial-budaya; keberhasilan tidak hanya ditentukan oleh toolchain, tetapi juga kolaborasi lintas tim, standardisasi proses tertentu, dan kualitas umpan balik yang mengalir dari produksi ke pengembangan [4]. Dari sudut SQA, hal ini berarti "assurance" mencakup kualitas perilaku organisasi: apakah proses mampu mengubah sinyal produksi menjadi perbaikan terstruktur.

Seiring meningkatnya kompleksitas sistem, SQA juga bergeser dari fokus tunggal pada pengujian fungsional menuju kualitas multi-aspek. Dalam domain pengujian, literature review tentang software product line (SPL) testing menggambarkan tantangan sistem konfigurasi tinggi: pengujian harus mempertimbangkan variasi fitur, biaya pengujian, serta non-functional testing yang sering belum tercakup memadai [6]. Ini memperlihatkan bahwa SQA menghadapi keterbatasan klasik: ruang uji yang eksplosif, keterbatasan sumber daya, dan kebutuhan strategi seleksi–prioritisasi pengujian. Karena itu, SQA modern bergantung pada teknik seleksi risiko, otomasi cerdas, dan metrik untuk memandu trade-off.

Pengukuran kualitas, meskipun penting, membawa tantangan interpretasi. Alsulami mengulas software metrics melalui systematic literature review dan menekankan metrik sebagai pengukuran karakteristik perangkat lunak yang dapat dihitung/diukur, khususnya terkait maintainability prediction dan kebutuhan alat serta prosedur yang sistematis [7]. Dalam SQA, metrik tidak boleh dipakai sebagai ornamen laporan, melainkan sebagai alat keputusan: menilai tren kualitas, memprediksi risiko cacat, dan memprioritaskan refactoring atau hardening. Namun, penggunaan metrik sering gagal ketika organisasi tidak menyepakati definisi kualitas yang sama atau mengabaikan konteks produk. Karena itu, integrasi model kualitas (mis. ISO/IEC 25010) dengan metrik operasional adalah kunci: metrik harus dipetakan ke karakteristik kualitas yang memang relevan bagi stakeholder.

Perkembangan lain yang penting adalah munculnya "platformisasi" SQA. Bernardo dkk. menunjukkan SQAaaS sebagai platform yang melakukan asesmen kualitas otomatis terhadap baseline kriteria, membuat pipeline CI/CD secara otomatis, dan memberi insentif melalui digital badges atas pencapaian kualitas tertentu [2]. Konsep ini memperluas SQA dari kegiatan internal menuju layanan yang dapat distandardisasi, direplikasi, dan dievaluasi lintas proyek. Dalam konteks riset dan open science, ini menjawab masalah reproducibility dan kualitas perangkat lunak riset yang sering tidak memiliki proses QA kuat; tetapi secara umum, konsep serupa dapat diadopsi perusahaan sebagai "quality platform" lintas tim.

Dengan demikian, landasan konseptual SQA modern dapat dirangkum sebagai tiga pengikat: definisi kualitas berbasis model (untuk menyamakan bahasa), mekanisme proses yang mencegah cacat (untuk menstabilkan delivery), dan bukti kualitas berbasis metrik serta pipeline otomatis (untuk memastikan "assurance" bukan klaim, tetapi dapat diverifikasi).

III. Praktik, Teknik, dan Tren SQA

Implementasi SQA modern dimulai dari integrasi kualitas ke dalam alur kerja pengembangan. Dalam organisasi Agile, SQA menjadi bagian dari desain sprint melalui acceptance criteria yang eksplisit, definisi "done" yang memasukkan pengujian otomatis, dan review lintas peran. Praktik ini menyadari bahwa cacat bukan sekadar kesalahan implementasi, tetapi sering berasal dari ambiguïtas kebutuhan, komunikasi lintas fungsi yang lemah, dan perubahan prioritas yang tidak terkendali. Dalam DevOps, integrasi tersebut menjadi lebih ketat melalui pipeline CI/CD yang menjalankan build, unit test, static analysis, security scan, integration test, hingga deployment otomatis. Keuntungan utamanya adalah konsistensi: pengujian tidak lagi bergantung pada "ritual manual", melainkan menjadi proses yang dapat diulang, diaudit, dan dioptimalkan.

Dalam literatur DevOps, keberhasilan pipeline bukan hanya persoalan otomasi, tetapi juga desain feedback loop. Azad dan Hyrynsalmi mengidentifikasi berbagai faktor sukses DevOps dan menunjukkan bahwa aspek teknis, organisasi, serta sosial-kultural saling mempengaruhi, sehingga kualitas harus dipahami sebagai hasil sistem sosio-teknis [4]. Untuk SQA, implikasinya adalah pentingnya governance kualitas: definisi kualitas dan kebijakan rilis harus dipahami bersama oleh developer, tester/QA, security, dan ops. Tanpa itu, pipeline berisiko menjadi "jalur cepat" yang hanya mempercepat rilis cacat.

Otomasi pengujian menjadi tulang punggung SQA modern karena frekuensi rilis meningkat. Systematic literature review tentang test automation dalam DevOps menegaskan peran pivot test automation di berbagai tahap pipeline dan menekankan bahwa continuous testing adalah pendorong penting untuk delivery cepat dengan risiko lebih rendah [5]. Continuous testing di sini tidak berarti semua hal diuji terus-menerus tanpa seleksi, tetapi bahwa setiap perubahan relevan memicu rangkaian pengujian yang cukup untuk memberi keyakinan rilis. Tantangan utamanya adalah biaya pemeliharaan test suite, flakiness, serta waktu eksekusi. Organisasi yang matang biasanya menerapkan strategi pengujian berlapis: unit test yang cepat dan luas, integration test yang lebih selektif, contract test untuk layanan, serta end-to-end test sebagai payung yang jumlahnya terbatas tetapi kritikal.

Di sisi pengukuran, metrik kualitas pada 2020–2024 tidak hanya berfokus pada defect density atau test coverage. Metrik berkembang ke arah produktivitas pipeline dan kualitas proses, misalnya lead time, change failure rate, mean time to recovery (MTTR), serta indikator kualitas kode yang memengaruhi maintainability. Alsulami menegaskan pentingnya metrik untuk memprediksi maintainability dan menunjukkan bahwa studi sistematis diperlukan untuk memahami indikator dan alat pengukuran yang tepat [7]. Ini selaras dengan kebutuhan SQA: maintainability sering menjadi "utang tersembunyi" yang muncul sebagai biaya tinggi ketika perubahan rutin tidak lagi aman. SQA yang kuat harus mampu mendeteksi sinyal-sinyal deteriorasi maintainability sebelum menjadi krisis, misalnya kompleksitas yang meningkat, churn tinggi pada modul rentan, atau ketergantungan yang rapuh.

Untuk menjaga keterkaitan antara metrik dan kebutuhan stakeholder, model kualitas ISO/IEC 25010:2023 dapat dijadikan peta. Standar ini menekankan model kualitas produk yang dapat digunakan untuk menetapkan ukuran (measures) karakteristik kualitas dan mendukung aktivitas seperti menetapkan tujuan pengujian dan acceptance criteria [1]. Dalam praktiknya, organisasi dapat memetakan metrik seperti latency p95/p99 ke performance efficiency, tingkat crash atau availability ke reliability, hasil SAST/DAST serta kerentanan dependency ke security, dan indikator perbaikan kode serta cyclomatic complexity ke maintainability. Pendekatan ini membantu mencegah "kebingungan metrik" karena setiap metrik memiliki alasan eksistensi yang terkait pada dimensi kualitas.

Tren menarik lainnya adalah lahirnya platform SQA terstandar lintas proyek. Bernardo dkk. menunjukkan SQAaaS sebagai platform open-source yang mengotomatiskan asesmen kualitas berbasis baseline, memfasilitasi pembuatan pipeline CI/CD untuk pengujian baseline, dan memberikan digital badges untuk pencapaian kualitas [2]. Dalam konteks perusahaan, konsep serupa bisa diterjemahkan sebagai "internal quality platform" yang menyediakan template pipeline, kebijakan kualitas, aturan scanning, serta dashboard metrik yang seragam. Dampaknya adalah mengurangi variasi kualitas antar tim dan mempercepat adopsi best practice, karena tim tidak perlu "menemukan ulang" proses QA dari nol.

SQA juga berhadapan dengan tantangan konfigurasi dan variabilitas, terutama pada sistem modular, product line, dan platform multi-tenant. Agh dkk. melalui SLR SPL testing menekankan adanya gap pada regression testing dan non-functional testing di SPL serta perlunya pendekatan baru untuk seleksi, prioritisasi, dan minimisasi regression test agar biaya pengujian tetap terkendali [6]. Perspektif ini memperkaya SQA modern: kualitas bukan hanya

memastikan satu varian berjalan benar, melainkan memastikan keluarga konfigurasi yang besar tetap aman, terutama saat rilis kecil terjadi sering. Pada sistem microservices, problem serupa muncul sebagai "variabilitas runtime": kombinasi versi layanan dan dependensi eksternal dapat berubah dinamis. Oleh karena itu, SQA perlu memadukan contract testing, canary release, feature flags, dan observability untuk mendeteksi regresi yang sulit direproduksi di lingkungan staging.

Tema yang semakin dominan pada 2020–2024 adalah kualitas untuk AI-based software. Gezici dan Kolukısa Tarhan meninjau kualitas perangkat lunak untuk sistem berbasis AI dan menyoroti atribut kualitas, model yang digunakan, tantangan, serta praktik yang dilaporkan dalam literatur [8]. Berbeda dari perangkat lunak deterministik, AI-based software memiliki sumber risiko tambahan seperti bias data, drift, ketidakstabilan performa model pada domain baru, serta kesulitan reproduksi hasil karena data dan pelatihan. Ini memaksa SQA memperluas definisi "verifikasi": bukan hanya memverifikasi kode, tetapi juga memvalidasi data, model, dan perilaku sistem pada skenario yang berubah. Dalam praktik modern, hal ini melahirkan gagasan quality gates untuk data dan model, misalnya pemeriksaan kualitas data, monitoring drift, serta evaluasi fairness dan robustness sebagai bagian dari pipeline.

Selain kualitas produk, SQA juga membutuhkan kualitas bukti. Dalam penelitian rekayasa perangkat lunak, Yang dkk. menunjukkan bahwa quality assessment terhadap studi yang direview menentukan kekuatan kesimpulan dan mendorong penggunaan instrumen QA yang lebih rigorous dan prosedur yang lebih jelas [3]. Jika prinsip ini diterapkan pada SQA organisasi, maka bukti kualitas—hasil pengujian, hasil scan keamanan, audit konfigurasi, hasil monitoring produksi—perlu dinilai bukan sekadar "ada", tetapi juga "berkualitas". Misalnya, test coverage tinggi tidak otomatis berarti kualitas tinggi jika test tidak bermakna; hasil scan keamanan tidak berarti aman jika aturan scanning tidak sesuai konteks; monitoring tidak berarti andal jika alert tidak actionable. Dengan demikian, SQA modern menuntut meta-quality: kualitas atas instrumen kualitas itu sendiri.

Secara operasional, SQA modern cenderung menggabungkan pendekatan preventif dan detektif. Preventif mencakup standardisasi coding, code review, design review, threat modeling, serta penggunaan arsitektur yang mendukung testability. Detektif mencakup pengujian otomatis, static/dynamic analysis, serta monitoring runtime. DevOps CSF menekankan pentingnya keseimbangan antara dimensi teknis dan organisasi [4]; ini mengisyaratkan bahwa organisasi yang hanya membeli tool QA tanpa membangun budaya kualitas akan menghadapi kegagalan implementasi karena resistensi, proses yang tidak sinkron, atau "alarm fatigue".

Bila ditarik lebih jauh, SQA modern bergerak menuju "continuous assurance". Ini bukan berarti semua hal dipastikan sempurna setiap saat, melainkan bahwa organisasi memiliki mekanisme yang membuat kualitas selalu terlihat, selalu dibahas, dan selalu memiliki jalur perbaikan. Keberhasilan continuous assurance ditandai oleh stabilitas rilis, penurunan insiden kritis, serta kemampuan recovery yang cepat ketika masalah terjadi. Dalam skenario ini, kualitas diperlakukan sebagai proses belajar: setiap bug dan insiden menjadi data untuk memperbaiki pengujian, prosedur review, dan kebijakan rilis.

IV. KESIMPULAN

Periode 2020–2024 menunjukkan bahwa Software Quality Assurance bukan lagi fungsi "penjaga gerbang" di akhir proyek, melainkan sistem pengelolaan kualitas yang tertanam dalam proses dan pipeline. Model kualitas seperti ISO/IEC 25010:2023 menyediakan bahasa bersama untuk mendefinisikan kualitas sebagai karakteristik yang dapat diturunkan menjadi kriteria penerimaan dan ukuran yang terpantau. DevOps memperkuat ide bahwa kualitas ditentukan oleh feedback loop lintas fungsi dan bahwa faktor teknis, organisasi, serta sosial-kultural berkontribusi pada keberhasilan implementasi kualitas. Otomasi pengujian dalam pipeline DevOps mengakselerasi pengendalian kualitas, namun menuntut strategi pengujian berlapis dan tata kelola bukti kualitas agar kualitas tidak menjadi ilusi angka.

Tren platformisasi SQA melalui konsep seperti SQAaaS menunjukkan arah industrialisasi kualitas: asesmen otomatis berbasis baseline, integrasi CI/CD yang seragam, dan insentif pemenuhan kualitas melalui mekanisme standar. Dalam sistem kompleks seperti product line atau konfigurasi tinggi, literatur mengingatkan bahwa tantangan regression testing dan non-functional testing masih signifikan dan memerlukan inovasi pada seleksi serta prioritisasi pengujian [6]. Dalam AI-based software, definisi kualitas semakin luas karena mencakup data dan model selain kode, sehingga SQA perlu memasukkan validasi data, monitoring drift, dan evaluasi robustness sebagai bagian dari assurance.

Arah pengembangan SQA ke depan, berdasarkan sintesis literatur 2020–2024, mengarah pada empat fokus besar. Pertama, penyatuan model kualitas dengan metrik operasional agar setiap metrik memiliki makna yang ditautkan pada kebutuhan stakeholder, sehingga mengurangi risiko metric gaming dan mendorong keputusan yang tepat. Kedua, penguatan quality governance dalam organisasi DevOps, karena keberhasilan tidak hanya bergantung pada toolchain tetapi juga pada kebijakan, kolaborasi, dan budaya yang memastikan kualitas tetap prioritas meski rilis cepat. Ketiga, perluasan SQA untuk sistem modern—microservices, product line, dan AI-based systems—yang menuntut kombinasi pengujian, observability, serta manajemen risiko yang lebih adaptif. Keempat, peningkatan kualitas bukti kualitas, dengan menilai rigor instrumen dan proses pengujian sebagaimana riset SLR menilai kualitas bukti ilmiah; prinsip bahwa kesimpulan bergantung pada kualitas bukti harus menjadi landasan pengambilan keputusan kualitas.

Pada akhirnya, SQA modern yang efektif bukanlah kumpulan checklist, melainkan kemampuan organisasi untuk membangun keyakinan berbasis bukti bahwa perangkat lunak akan berperilaku baik di kondisi nyata, sekaligus mampu beradaptasi ketika kondisi berubah. Organisasi yang berhasil menerapkan SQA modern biasanya memadukan standar kualitas sebagai kompas, otomasi sebagai mesin, metrik sebagai instrumen navigasi, dan budaya kolaboratif sebagai energi penggerak.

REFERENSI

[1] ISO/IEC, "ISO/IEC 25010:2023 Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—Product quality model," International Organization for Standardization, 2023.

[2] S. Bernardo et al., "Software Quality Assurance as a Service: Encompassing the quality assessment of software and services," Future Generation Computer Systems, 2024, doi: 10.1016/j.future.2024.03.024.

[3] L. Yang, H. Zhang, H. Shen, X. Huang, X. Zhou, G. Rong, and D. Shao, "Quality Assessment in Systematic Literature Reviews: A Software Engineering Perspective," Information and Software Technology, vol. 130, 2021, art. no. 106397, doi: 10.1016/j.infsof.2020.106397.

[4] N. Azad and S. Hyrynsalmi, "DevOps critical success factors — A systematic literature review," Information and Software Technology, vol. 157, 2023, art. no. 107150, doi: 10.1016/j.infsof.2023.107150.

[5] A.R. Patel. "The State of Test Automation in DevOps: A Systematic Literature Review," 2022 Fourteenth International Conference on Contemporary Computing (IC3), 2022.

[6] H. Agh, A. Azamnouri, and S. Wagner, "Software product line testing: a systematic literature review," Empirical Software Engineering, vol. 29, art. no. 146, 2024, doi: 10.1007/s10664-024-10516-x.

[7] M. Alsulami, "A Systematic Literature Review on Software Metrics," International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies, vol. 12, no. 12, pp. 1–13, 2021, doi: 10.14456/ITJEMAST.2021.238.

[8] B. Gezici and A. Kolukısa Tarhan, "Systematic literature review on software quality for AI-based software," Empirical Software Engineering, vol. 27, no. 3, 2022, doi: 10.1007/s10664-021-10105-2.