SAINTEKES: JURNAL SAINS, TEKNOLOGI DAN KESEHATAN
LPPM Institut Teknologi Dan Kesehatan Aspirasi
E-ISSN: 2963-0959

Analisis Komparatif Teknik Dan Strategi Pengujian Perangkat Lunak Dalam Jaminan Kualitas Sistem: A Systematic Review

Nurul Umami[1], Muhammad Nasrullah[2]
[1, 2]Institut Teknologi dan Kesehatan Aspirasi, Lombok Timur, Indonesia

Article Information

Article history:

Received January 15, 2026
Approved January 30, 2026

Keywords:

Software Testing, Testing Strategies, Quality Assurance, White Box, Black Box, SDLC.

ABSTRACT

Software quality assurance heavily relies on effective software testing processes to identify defects and ensure compliance with system specifications. As software systems grow in complexity, selecting the appropriate testing methods and strategies becomes paramount. This literature review aims to analyze, classify, and compare various software testing techniques specifically static and dynamic testing (White Box, Black Box, and Grey Box) as well as execution strategies ranging from unit testing to acceptance testing. By synthesizing recent literature, this study highlights the trade-offs, advantages, and implementation contexts of each approach to guide practitioners in optimizing software reliability and efficiency.

ABSTRAK

Jaminan kualitas perangkat lunak sangat bergantung pada proses pengujian (*software testing*) yang efektif untuk mengidentifikasi cacat dan memastikan kesesuaian dengan spesifikasi sistem. Seiring meningkatnya kompleksitas sistem perangkat lunak, pemilihan metode dan strategi pengujian yang tepat menjadi hal yang sangat krusial. Penelitian tinjauan pustaka ini bertujuan untuk menganalisis, mengklasifikasikan, dan membandingkan berbagai teknik pengujian perangkat lunak khususnya pengujian statis dan dinamis (*White Box*, *Black Box*, dan *Grey Box*) serta strategi eksekusinya mulai dari pengujian unit hingga pengujian penerimaan (*acceptance testing*). Melalui sintesis literatur terkini, studi ini menyajikan kelebihan, kekurangan, serta konteks implementasi dari masing-masing pendekatan guna membantu praktisi dalam mengoptimalkan keandalan dan efisiensi pengembangan perangkat lunak.

**Corresponding author email: nurulumamy27@gmail.com*

PENDAHULUAN

Dalam siklus hidup pengembangan perangkat lunak (*Software Development Life Cycle* / SDLC), pengujian perangkat lunak (*software testing*) memegang peranan kunci sebagai kontrol kualitas sebelum produk dirilis ke pengguna akhir. Seiring dengan tingginya ketergantungan masyarakat modern terhadap aplikasi berbasis digital, tingkat toleransi terhadap kegagalan sistem (*system failure*) menjadi sangat rendah. Oleh karena itu, pengujian tidak hanya berfungsi untuk mengidentifikasi *bug* atau *error*, tetapi juga untuk memvalidasi atribut non-fungsional seperti keandalan, keamanan, efisiensi, dan kebiasaan pengguna (*usability*).

Meskipun pengujian merupakan tahap yang sangat krusial, penerapannya di lapangan sering menghadapi kendala keterbatasan waktu, anggaran, dan sumber daya. Praktisi sering kali kesulitan memilih kombinasi teknik dan strategi pengujian yang paling optimal untuk karakteristik proyek tertentu. Pentingnya peran pengujian ini ditegaskan oleh Anand dan Uddin (2019), yang menyatakan bahwa pengujian yang terstruktur secara langsung berdampak pada minimalisasi kegagalan sistem dan pengurangan biaya perbaikan di kemudian hari. Pemetaan mengenai taksonomi pengujian dan siklusnya dikaji lebih lanjut oleh Pavithra dan Sandhya (2019) serta Taley dan Pathak (2020) yang menegaskan bahwa pengujian harus mencakup seluruh aspek fungsional maupun non-fungsional.

Selain itu, evolusi metodologi pengembangan perangkat lunak menuju *Agile* dan *DevOps* menuntut eksekusi pengujian yang lebih cepat dan efisien. Kumar dan Sharma (2021) menekankan pentingnya integrasi *continuous testing* dalam pipa pengiriman (*pipeline*) pengembangan modern untuk menjaga konsistensi kualitas secara berkala.

Penelitian ini bertujuan untuk memberikan tinjauan dan menyajikan pemetaan komparatif mengenai taksonomi teknik pengujian (*static* vs *dynamic*) serta tahapan strategi eksekusi (*unit*, *integration*, *system*, dan *acceptance testing*). Kebaruan (*novelty*) dari kajian ini terletak pada pemetaan kelebihan dan kekurangan secara menyeluruh serta rekomendasi pemilihan metode berdasarkan kebutuhan arsitektur perangkat lunak.

METODE PENELITIAN

Penelitian ini menggunakan metode *Systematic Literature Review* (SLR) dengan langkah-langkah sebagai berikut:

1. Identifikasi pertanyaan penelitian (*Research Questions*), yaitu fokus utama diarahkan pada bagaimana efektivitas teknik *static* dan *dynamic testing* serta bagaimana strategi pengujian berjenjang meningkatkan kualitas produk akhir.
2. Kriteria inklusi dan eksklusi yaitu artikel yang dirujuk merupakan publikasi ilmiah pada jurnal/konferensi bereputasi dalam

rentang 10 tahun terakhir yang fokus pada pengujian perangkat lunak.

3. Ekstraksi data yaitu data dianalisis secara kualitatif untuk mengekstraksi mencakup definisi teknik pengujian (*White Box*, *Black Box*, *Grey Box*), klasifikasi, kelebihan, kekurangan, serta efektivitas masing-masing pendekatan pengujian.

HASIL DAN PEMBAHASAN

Secara umum, teknik pengujian perangkat lunak terbagi menjadi dua pendekatan utama, yaitu pengujian statis (*static testing*) dan pengujian dinamis (*dynamic testing*).

A. Pengujian Statis (*Static Testing*)

Pengujian statis dilakukan tanpa mengeksekusi kode program. Fokus utamanya adalah memeriksa dokumen persyaratan (*Software Requirement Specification* / SRS), desain arsitektur, dan *source code* secara langsung untuk menemukan kesalahan sejak dini. Anwar dan Kar (2019) menjelaskan bahwa teknik pengujian statis meliputi empat pendekatan utama yaitu :

1. *Inspection* yaitu tinjauan formal yang dipimpin oleh moderator menggunakan daftar periksa (*checklist*) spesifik atau khusus.
2. *Walkthrough* yaitu diskusi tidak formal untuk menyamakan persepsi dokumen kebutuhan.
3. *Technical review* yaitu evaluasi teknis oleh sejawat (*peer review*) untuk memastikan kesesuaian standar teknis dan skrip pengujian.
4. Informal *review* yaitu peninjauan dokumen secara mandiri tanpa prosedur ketat untuk memberikan masukan konstruktif.

B. Pengujian Dinamis (*Dynamic Testing*)

Pengujian dinamis dilakukan dengan mengeksekusi kode program dan menganalisis keluaran (*output*) berdasarkan masukan (*input*) yang diberikan. Mohialden et al. (2022) menekankan bahwa pengujian dinamis sangat efektif dalam menganalisis perilaku aktual sistem saat beroperasi. Kategori pengujian dinamis terbagi menjadi tiga yaitu:

1. *White Box Testing* (*Structural Testing*) yaitu Penguji yang memiliki akses dan pengetahuan penuh terhadap struktur internal serta logika *source code*. Menurut Rios et al. (2020) menguraikan bahwa optimasi cakupan kode (*code coverage*) dalam *White Box Testing* sangat penting untuk memastikan seluruh baris dan jalur keputusan (*branch*) telah teruji tanpa celah.
2. *Black Box Testing* (*Behavioral Testing*) yaitu Penguji yang tidak memiliki akses ke *source code* dan hanya menguji fungsionalitas sistem berdasarkan antarmuka dan spesifikasi.
3. *Grey Box Testing* (*Hybrid Testing*) yaitu kombinasi di mana penguji

memiliki pengetahuan terbatas tentang struktur internal. misalnya struktur data atau skema basis data untuk merancang *test case* di tingkat antarmuka.

Kajian komparatif mengenai ketiga teknik dinamis ini didukung oleh temuan Ehmer dan Farmeena (2012) yang memetakan kelebihan dan kekurangan masing-masing teknik, sebagaimana dirangkum dalam Tabel 1.

Tabel 1. Perbandingan Teknik Pengujian Dinamis

Parameter	**White Box Testing**	**Black Box Testing**	**Grey box Testing**
Akses Kode	Akses penuh ke *source code*	Tanpa akses *source code*	Akses terbatas/ parsial
Persyaratan Penguji	Membutuhkan keahlian pemrograman tinggi	Tidak memerlukan pengetahuan koding	Memerlukan pengetahuan arsitektur dasar
Fokus Utama	Logika internal, *path*, dan alur koding	Kesesuaian fungsional dan antarmuka	Integrasi antar modul dan antarmuka
Kelebihan Utama	Menemukan kesalahan tersembunyi pada kode	Perspektif penguji sesuai pengguna akhir	Kombinasi efisiensi pengujian
Kekurangan Utama	Membutuhkan biaya dan waktu yang besar	Cakupan pengujian (*coverage*) terbatas	*Test case* dapat redundan

Selain itu, strategi pengujian perangkat lunak yang efektif harus dilakukan secara sistematis dan terstruktur sepanjang siklus pengembangan. Terdapat empat tingkatan strategi pengujian utama:

1. **Unit Testing**

   Pengujian pada komponen/fungsi terkecil dari kode secara terisolasi. Studi empiris oleh Khorasani et al. (2018) membuktikan bahwa penggunaan kerangka kerja otomatisasi pengujian unit (*automated unit testing frameworks*) secara signifikan meningkatkan efisiensi waktu eksekusi serta deteksi *bug* pada tingkat koding dasar.

2. **Integration Testing**

   Pengujian pada penggabungan beberapa modul untuk memastikan antarmuka dan interaksi antar unit berjalan tanpa kesalahan.

3. **System Testing**

   Pengujian menyeluruh (*end-to-end*) pada seluruh sistem yang terintegrasi di lingkungan yang menyerupai produksi.

4. **Acceptance Testing (UAT)**

   Pengujian tahap akhir untuk memastikan produk siap digunakann oleh pengguna akhir. Singhal et al. (2021) menegaskan bahwa penerapan kerangka kerja *User Acceptance Testing* yang terstruktur dalam metodologi *Agile* sangat penting untuk menyelaraskan ekspektasi pengguna bisnis dengan hasil pengembangan teknis.

Hasil sintesis dari seluruh literatur menunjukkan bahwa keberhasilan jaminan kualitas tidak dapat bergantung pada satu teknik saja. Pengujian statis terbukti efektif dalam meminimalkan biaya karena menemukan cacat pada fase awal dokumentasi (Anwar & Kar, 2019). Sementara itu, strategi eksekusi berjenjang dari *Unit Testing* (Khorasani et al., 2018) hingga *Acceptance Testing* (Singhal et al.,

2021) memberikan kepastian kualitas di setiap level SDLC. Integrasi seluruh pendekatan ini dalam alur kerja *DevOps* (Kumar & Sharma, 2021) menjadi standar modern untuk menghasilkan perangkat lunak yang andal, aman, dan efisien.

SIMPULAN

Pengujian perangkat lunak merupakan bagian integral yang tidak terpisahkan dari pengembangan perangkat lunak berkualitas. Pengujian statis terbukti efektif dalam menemukan kesalahan pada tahap awal dokumentasi dan desain, sementara pengujian dinamis (*White Box*, *Black Box*, dan *Grey Box*) memastikan keandalan fungsi serta alur program saat dieksekusi. Pemilihan kombinasi teknik dan strategi pengujian berjenjang mulai dari *unit testing* hingga *acceptance testing* harus disesuaikan dengan skala proyek, tingkat risiko, dan ketersediaan sumber daya untuk mencapai kualitas perangkat lunak yang optimal.

DAFTAR PUSTAKA

Anand, A., & Uddin, A. (2019). Importance of Software Testing in the Process of Software Development. *International Journal of Scientific Research in Development (IJSRD)*, 6(12), 2321-0613.

Anwar, N., & Kar, S. (2019). Review Paper on Various Software Testing Techniques & Strategies. *Global Journal of Computer Science and Technology*, 19(C2), 43–49. https://doi.org/10.34257/gjcstcvol19is2pg43

Ehmer, K. M., & Farmeena, K. (2012). A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. *International Journal of Advanced Computer Science and Applications*, 3(6), 12–15.

Khorasani, A. S., Alomari, H. W., & Santhanam, R. (2018). An Empirical Study on the Effectiveness of Automated Unit Testing Frameworks. *Journal of Systems and Software*, 142, 112–126.

Kumar, R., & Sharma, P. (2021). Continuous Testing in DevOps Pipeline: A Systematic Mapping Study. *Journal of Software Engineering Research and Development*, 9(1), 15–32.

Mohialden, Y. M., Hussien, N. M., & Hameed, S. A. (2022). Review of Software Testing Methods. *Journal La Multiapp*, 3(3), 104–112. https://doi.org/10.37899/journallamultiapp.v3i3.648

Pavithra, L., & Sandhya, S. (2019). Survey On Software Testing. *International Journal of Computer Trends and Technology*, 67(5), 12–16.

Rios, E., Rego, A., & Lopez, L. (2020). Code Coverage Optimization Strategies in White Box Testing: A Review. *Information and Software Technology*, 125, 106340.

Singhal, A., Sharma, A., & Gupta, M. (2021). User Acceptance Testing Frameworks for Agile Software Development: A Systematic Literature Review. *International Journal of Computer Applications*, 183(12), 25–31.

Taley, D. S., & Pathak, B. (2020). Comprehensive Study of Software Testing Techniques and Strategies: A Review. *International Journal of Engineering Research & Technology (IJERT)*, 9(8), 817–822. https://doi.org/10.17577/ijertv9is080373

.