E-ISSN: 2715-2731

Survei Teknik-Teknik Pengujian *Software* Menggunakan Metode *Systematic Literature Review*

Alfian Arifandi[1], Raihan Nafal Zuhdi Simamora[2], Geovanni Azam Janitra[3], Muhammad Ainul Yaqin[4], Muhamat Maariful Huda[5]

Program Studi Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim, Indonesia
Program Studi Ilmu Komputer, Universitas Nahdlatul Ulama Blitar, Indonesia

[1]19650076@student.uin-malang.ac.id; [2]19650085@student.uin-malang.ac.id; [3]19650089@student.uin-malang.ac.id; [4]yaqinov@ti.uin-malang.ac.id, [5]muhamatmaariful@unublitar.ac.id

INFO ARTIKEL

Sejarah Artikel
Diterima: 18 Januari 2022
Direvisi: 11 Juni 2022
Diterbitkan: 29 Desember 2022

Kata Kunci
Software testing
Systematic Literature Review
Tools

ABSTRAK

Perangkat lunak atau *software* merupakan himpunan file digital berupa program atau perintah untuk menjalankan tugas yang dikelola dan disimpan di *personal* komputer. Tujuan utama penelitian ini dilakukan adalah untuk mengetahui berbagai macam teknik-teknik pengujian *software* yang ada. Teknik-teknik tersebut nantinya dapat dibandingkan \ antara satu dengan yang lain untuk mendapatkan teknik yang paling efektif dan memenuhi standarisasi ISO dalam pengujian *software*. Metode yang digunakan dalam penelititan ini adalah metode tinjauan pustaka sistematis atau *Systematic Literature Review*, yakni metode literatur tinjauan yang menelaah, menilai, dan menafsirkan semua hasil pada suatu topik dalam penelitian yang dapat menjawab pertanyaan pada penelitian. Dengan menggunakan metode SLR penelitian dapat berisifat lebih subjektif dan hasil nya diharapkan dapat menambah literatur tentang penggunaan Metode SLR dalam identifikasi jurnal. Hasil penelitian menunjukkan bahwa beberapa metode yang digunakan untuk menguji *software* antara lain metode *black box*, metode *white box*, dan metode *grey box* dengan metode yang paling sering digunakan yaitu metode black box. Beberapa metode yang memenuhi standarisasi ISO harus memenuhi beberapa aspek seperti *efficiency*, *effectiveness*, *learnability*, *satisfaction*, dan *error*. Metode *black box* dan metode *white box* adalah metode yang memenuhi aspek tersebut.

PENDAHULUAN

Perangkat lunak atau *software* merupakan himpunan file digital berupa program atau perintah untuk menjalankan tugas yang dikelola dan disimpan di personal komputer [1]. *Software* berbeda dengan *hardware*, baik dari fisik, segi sistem yang dibangun dan proses dalam menjalankan pekerjaan atau perintah. Dalam ilmu komputer dan rekayasa *software*, *software* komputer merupakan keseluruhan informasi termasuk program dan data yang diproses sistem komputer. *Software* komputer mencakup program komputer, pustaka, dan data yang tidak dapat dieksekusi, seperti pada media digital atau dokumentasi online. *Software* dan perangkat keras pada komputer bekerja sama dan saling terkait, sehingga tidak dapat digunakan sendiri-sendiri secara realistis.

Software Testing merupakan proses menemukan celah atau kesalahan (*bug*) di setiap elemen perangkat lunak lalu merekam hasilnya yang kemudian dilanjutkan dengan mengevaluasi setiap aspek dari setiap komponen serta mengevaluasi fungsionalitas *software* yang sedang dikembangkan tersebut [2]. Tujuan utama pengujian *software* yaitu mencari *bug* dalam *software* atau kesalahan lain agar *software* sesuai dengan apa yang diinginkan. *Bug* merupakan celah kelemahan dan kerusakan di suatu *software* yang tidak diinginkan karena

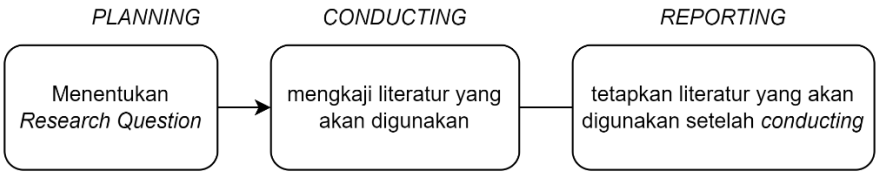
tidak selaras dengan keperluan *software* tersebut [3]. Pengujian *software* juga dapat dilakukan untuk mencari tahu apakah *software* dapat memberikan *output* yang benar untuk *input* yang berbeda, mampu menyelesaikan tugas dalam batas waktu yang dapat diterima, dan dapat berjalan pada *environments* yang berbeda. Mengingat pentingnya pengujian perangkat lunak pada pengembangan perangkat lunak, oleh karena itu pada pembahasan ini penulis mengumpulkan beberapa literatur dari penelitian terdahulu mengenai pengujian perangkat lunak guna mempelajari teknik-teknik pengujian perangkat lunak. Pengumpulan data dilakukan dengan penyaringan literatur jurnal penelitian terdahulu yang membahas tentang pengujian perangkat lunak. Data yang sudah dikumpulkan nantinya akan diidentifikasi dengan menggunakan metode SLR. Alasan dalam menggunakan metode SLR yaitu penelitian dapat bersifat lebih subjektif dan hasil nya diharapkan dapat menambah literatur tentang penggunaan Metode SLR dalam identifikasi jurnal [4].

International Organization for Standarization (ISO) adalah sebuah organisasi Internasional yang misinya adalah membuat dokumen standarisasi internasional yang berisi persyaratan, spesifikasi, pedoman atau karakter yang digunakan untuk memastikan bahan, proses serta produk dan bahkan layanan yang sesuai untuk memenuhi tujuan yang dimaksud [5]. Teknik-teknik yang digunakan dalam pengujian *software* nantinya harus memenuhi standarisasi ISO agar mutu yang diberikan tetap terjaga hingga sampai ke tangan konsumen. Beberapa penilitian terlebih dahulu mengenai *Software Testing* telah dilakukan seperti penelitian yang dilakukan yang berjudul *A Comparative Study of Software Testing Techniques* yang dilakukan oleh Anju Bansal [6]. Pada penelitian tersebut telah menjelaskan ketiga metode pengujian perangkat lunak lengkap beserta taksonomi pengujian, kelebihan dan kelemahan setiap metode, serta tabel perbandingan tiap metode, namun tidak menjelaskan metode apa yang memenuhi standarisasi ISO.

Penelitian berikutnya yang berjudul *Bridge between Black Box and White Box – Gray Box Testing Technique* oleh Acharya and Pandya [7]. dimana pada penelitian tersebut selain menjelaskan ketiga metode pengujian *software* dengan kelebihan serta kelemahan tiap metode serta tabel perbandingannya. Namun belum terdapat adanya taksonomi pengujian, gambar penjelasan tiap metode dan juga standarisasi ISO. Sehingga dilakukannya penelitian yang berjudul *Survei Teknik-Teknik Pengujian Software Menggunakan Metode Systematic Literature Review* selain bertujuan untuk mengetahui dan menilai metode-metode pengujian *software* juga digunakan standarisasi ISO dalam penyusunanya. Digunakannya standarisasi ISO dikarenakan ketidaksesuaian *software testing* dengan standarisasi ISO dapat menghambat terjadinya proses pengembangan *software*. Untuk menyelesaikan masalah tersebut, penulis menyatukan *paper – paper* atau referensi yang didapatkan agar pengembang membuat atau melakukan uji coba *software* dengan standart yang sudah ditetapkan oleh ISO.

METODE

Survei yang dilakukan pada penlitian ini menggunakan metode *Systematic Literature Review* (SLR). Tinjauan pustaka sistematis atau yang lebih sering dikenal sebagai SLR (Systematic Literature Review) merupakan *literature review* yang mengidentifikasi, mengevaluasi, dan menginterpretasikan semua temuan pada topik penelitian untuk menjawab pertanyaan penelitian yang telah diidentifikasi sebelumnya [9] . Umum nya terdapat beberapa tahapan dalam melakukan metode SLR yaitu *Planning*, *Conducting* dan *Reporting*. Tujuan menggunakan literatur review yaitu mendapatkan landasan teori yang dapat mendukung pada penyelesaian masalah yang sedang diselidiki.



Gambar 1. Alur SLR

Planning

Bagian awal dari SLR adalah dengan Menyusun *Research Question* (RQ). Pada tahap ini penulis diminta untuk menyusun *Research Question* atau pertanyaan penelitian. Pertanyaan ini dibuat sesuai dengan topik yang diambil. Pertanyaan-pertanyaan tersebut nantinya akan menjadi acuan dalam melakukan proses pencarian dan ektraksi dari referensi literatur yang didapat. Formulasi RQ wajib mengikuti 5 elemen PICOC seperti Tabel 1.

Tabe1 1. PICOC

P	*Population* (target investigasi)	Teknik-teknik *software testing* yang seseuai dengan standarisasi ISO
I	*Intervention* (aspek detail dari investigasi)	Pembagian dari tipe-tipe teknik-teknik pengujian *software*
C	*Comparison* (perbandingan *intervention*)	Membandingkan setiap teknik-teknik *software testing* berdasarkan ISO
O	*Outcomes* (hasil investigasi)	Keuntungan dan kekurangan dari setiap teknik *software testing*
C	*Context* (setting dan lingkungan dari investigasi)	Studi di bidang industri dan akademik

Berdasarkan *table* di atas, dapat disimpulkan *resesearch question* dari peneliti terkait penelitian ini sebagaimana Tabel 2.

Tabel 2. Research question

RQ1	Apa saja metode yang digunakan untuk pengujian *software*?
RQ2	Apa metode yang paling sering digunakan dalam pengujian *software*?
RQ3	Apa saja metode pengujian *software* yang paling memenuhi standarisasi ISO?

Conducting

Conducting merupakan tahap pada SLR setelah dilaksanakannya *planning*. Pada tahap ini sudah harus memulai dengan menentukan kata kunci dari pencarian literatur (*search string*) berdasarkan PICOC yang sudah dirancang sebelumnya sesuai dengan protokol SLR yang ditentukan. Memahami sinonim dan alternatif yang digunakan untuk menentukan keakuratan tinjauan pustaka yang akan dicari:

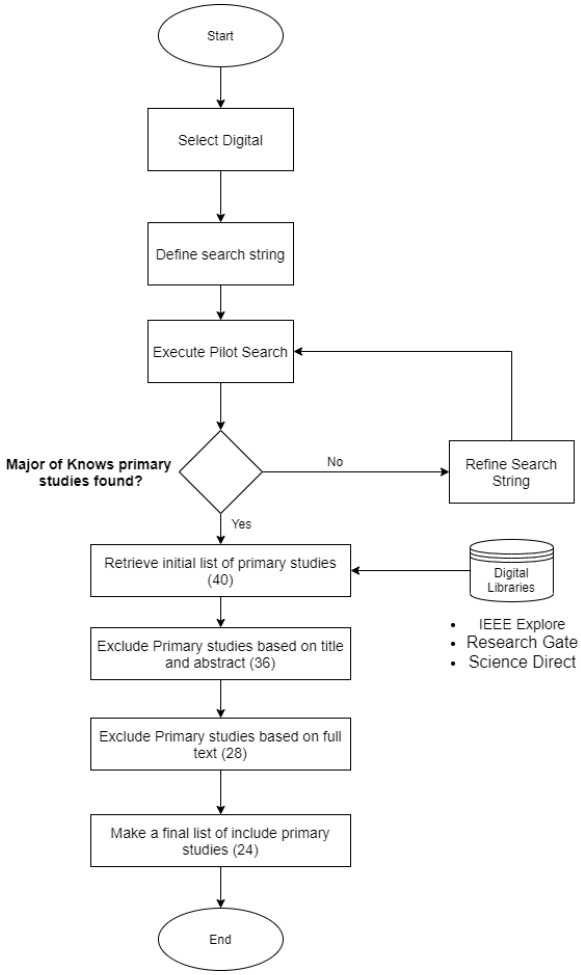
(software)
AND
(testing OR measuring OR examining)*
AND
(technique OR approach OR procedure)*

Selanjutnya, mengidentifikasi sumber tinjauan pustaka. kami menggunakan sumber literatur review yang diambil dari perpustakaan digital berupa google scholar. Sumber pencarian literatur kami meliputi ResearchGate, IEEE Xplore, dan ScienceDirect. Setelah memiliki literatur, gunakan inklusi dan eksklusi untuk mengkurasi literatur yang sesuai. Berikut merupakan Tabel 3, kriteria inklusi dan eksklusi untuk penelitan ini :

Tabel 3. Kriteria *inclusion* dan kriteria *exclution*

Inclusion Criteria	***Exclusion Criteria***
• Topik pembahasan mengenai teknik-teknik *software testing* dengan cakupan literatur tidak terlalu luas • Literatur menggunakan Bahasa Indonesia atau Bahasa Inggris • Membahas implementasi dari teknik-teknik *software testing*	• Cakupan literatur terlalu luas • Literatur tidak menggunakan Bahasa Indonesia atau Bahasa Inggris • Literatur telah dipublikasikan sebelum tahun 2000

Berdasarkan Tabel 3 tersebut dapat disimpulkan bahwa paper mempunyai tolak ukur dalam pengambilan refrensi yaitu paper yang menganalisis tentang teknik-teknik pengujian perangkat lunak yang pada paper tersebut tidak mengulas jangkauan yang terlalu luas, literatur menggunakan Bahasa Indonesia ata Bahasa Inggris, dan paper tersebut menjelaskan tentang implementasi dari teknik-teknik pengujian perangkat lunak. Berikut merupakan *flowchart* dari tahap *conducting* yang memuat pencarian jurnal untuk direview:



Gambar 2. *Flowchart conducting*

Berdasarkan gambar *flowchart* tersebut dapat disimpulkan bahwa pencarian literatur diawali dengan menentukan *keyword* yang kemudian diseleksi berdasarkan *inclusion criteria* dan *exclusion criteria* yang telah ditentukan.

Reporting

Reporting merupakan tahapan penulisan hasil *system literature review* yang dilakukan. Setelah beberapa tahap tersebut dilakukan, peneliti mendapatkan beberapa data diantaranya yaitu *quality assessment*, *data collection*, dan *data analysis*. Berikut adalah hasil *reporting* yang tertera pada deskripsi di bawah

Quality Assesment

Data yang didapatkan kemudian dinilai berdasar pada pertanyaan kriteria penilaian kualitas seperti berikut :

QA1. Apakah dituliskan metode yang digunakan untuk pengujian *software* pada paper jurnal?
QA2. Apakah dituliskan metode yang paling sering digunakan dalam pengujian *software* pada paper jurnal?
QA3. Apakah dituliskan metode pengujian *software* yang paling memenuhi standarisasi ISO pada paper jurnal?

Dari setiap paper, akan diberi nilai jawaban untuk setiap pertanyaannya.

1. Huruf Y (Ya) : untuk metode yang digunakan dalam pengukuran *usability software*.
2. Huruf X (Tidak) : untuk metode yang paling sering digunakan dan yang paling memenuhi standarisasi ISO dalam pengukuran *usability software.*

Data Collection

Mengumpulkan data dari penelitian yang akan menjadi data primer dan data sekunder.

a. Data Primer

Data primer yang digunakan berupa jurnal yang didapatkan dari https://www.scholar.google.com dengan waktu publikasi tidak kurang dari tahun 2010 hingga sekarang karena berbagai alasan berikut ini:

1. *Google Scholar* memberikan akses yang cukup baik.
2. Data yang dibutuhkan mudah untuk dicari, karena terdapat pengaturan rentang waktu yang bisa disesuaikan berdasarkan kebutuhan dari peneliti.

b. Data Sekunder

Data sekunder atau data yang dihasilkan melalui proses pengumpulan data yang diperlukan melalui *website Google*. Data dalam penelitian ini dihasilkan melalui beberapa tahapan sebagai berikut:

1. *Observasi* (Pengamatan)

   Pengamatan dan data yang dikumpulkan secara langsung pada sumber utama, yaitu https://www.scholar.google.com.
2. Studi Pustaka

   Studi pustaka dilakukan pada data yang didapatkan dari https://www.scholar.google.com yang terasosiasi terhadap Metode SLR.
3. Dokumentasi

   Data yang didapatkan kemudian dikumpulkan kedalam *software*. Berikut ini merupakan langkah dari upaya pengumpulan data, yang dimulai dari pengamatan sampai dokumentasi yang didapatkan dari https://www.scholar.google.com.

1. Mengunjungi situs https://www.scholar.google.com.
2. Menggunakan beberapa *keyword* seperti *"software testing"*dan*"teknik pengujian software"* pada form pencarian dengan melihat publikasi *paper* tidak kurang dari tahun 2010.
3. Kemudian diklik tombol "cari", sehingga ditampilkan judul, waktu publikasi, beserta nama penulis.

Data Analisis

Data yang telah didapatkan dan dikelompokkan selanjutnya dianalisis untuk menunjukkan:

1. Metode yang digunakan dalam pengujian *software* (mengarah pada RQ1).
2. Metode yang paling sering dipakai dalam pengujian *software* (mengarah pada RQ2).
3. Metode pengujian software yang paling memenuhi standarisasi ISO (mengarah pada RQ3).

HASIL DAN PEMBAHASAN

Hasil *Search Process*

Hasil dari *search process* dikumpulkan sesuai dengan tipe jurnal, guna memudahkan peneliti untuk mengetahui tipe jurnal yang didapatkan.

Tabel 4. Pengelompokkan berdasarkan jurnal

No.	Tipe Jurnal	Jumlah	Sumber
1.	Pengujian *Software* dengan Menggunakan Metode *White Box* dan *Black Box*	1	[10]
2.	Strategi, Teknik, Faktor Pendukung dan Penghambat Pengujian untuk Pengembang *Software* Pemula	1	[11]
3.	Pengujian *Software* Menggunakan Metode *Boundary Value Analysis* dan *Decision Table Testing*	1	[8]
4.	JURNAL IPTEK – Pengujian Aplikasi Web – Tinjauan Pustaka Sistematis	1	[12]
5.	Pengujian Aplikasi dengan Metode *Blackbox Testing Boundary Value Analysis*	1	[13]
6.	Perancangan Dan Pengujian *Software* Kriptografi Gabungan *Playfair Cipher* Dan *Electronic Code Book* (ECB)	1	[14]
7.	Metode *Systematic Literature Review* untuk Identifikasi Platform dan Metode Pengembangan Sistem Informasi di Indonesia	1	[15]
8.	Pengujian *Black Box* pada Aplikasi Sistem Informasi Akademik Menggunakan Teknik *Equivalence Partitioning*	1	[16]
9.	Pengujian *Black Box* pada Aplikasi Penjualan Berbasis Web Menggunakan Metode *Equivalents Partitions* (Studi Kasus: PT Arap Store)	1	[7]
10.	Pengujian *Software* dengan Menggunakan Model *Behaviour UML*	1	[2]
11.	Pengujian Sistem Informasi Akademik menggunakan *Mccall's Software Quality Framework*	1	[17]
12.	Pengujian *Black Box* pada Aplikasi Penjualan Berbasis Web Menggunakan Teknik *Boundary Value Analysis*	1	[18]
13.	Pengujian pada Aplikasi Penggajian Pegawai dengan menggunakan Metode *Blackbox*	1	[6]
14.	Pengujian Perangkat Lunak Metode Black-Box Berbasis *Equivalence Partitions* pada Aplikasi Sistem Informasi Sekolah	1	[19]
15.	*Grey-Box Technique of Software Integration Testing Based on Message*	1	[20]
16.	*Black-Box Approach For Software Testing Based On Fat-Property*	1	[21]

17.	*A Comparative Study of Software Testing Techniques Viz. White Box Testing Black Box Testing and Grey Box Testing*	1	[22]
18.	*Software Testing Techniques: A Literature Review*	1	[23]
19	Survei Teknik Pengujian *Software*	1	[24]

Hasil Seleksi *Inclusion Criteria* dan *Exclusion Criteria*

Hasil yang didapat dari *search process* kemudian dipilah berdasar pada kriteria batasan (*exclusion*) dan pemasukan (*inclusion*) dan menyisihkan 11 jurnal yang kemudian dilangsungkan pemeriksaan data. Sehingga pada Tabel 1 memperlihatkan hasil kualitas penilaian (*quality assesment*) yang selanjutnya menunjukkan bahwa data tersebut dapat atau tidaknya dipakai dalam penelitian ini.

Hasil Kualitas Penilaian (*Quality Assesment*)

Tabel 5. Hasil *quality assesment*

No.	Tipe Jurnal	QA1	QA2	QA3	Hasil
1.	Pengujian *Software* dengan Menggunakan Metode *White Box* dan *Black Box*	Y	Y	Y	Y
2.	Strategi, Teknik, Faktor Pendukung dan Penghambat Pengujian untuk Pengembang *Software* Pemula	Y	X	Y	Y
3.	Pengujian *Software* Menggunakan Metode *Boundary Value Analysis* dan *Decision Table Testing*	Y	X	Y	Y
4.	JURNAL IPTEK - Pengujian Aplikasi Web - Tinjauan Pustaka Sistematis	Y	X	X	X
5.	Pengujian Aplikasi dengan Metode *Blackbox Testing Boundary Value Analysis*	Y	Y	Y	Y
6.	Perancangan Dan Pengujian *Software* Kriptografi Gabungan *Playfair Cipher* Dan *Electronic Code Book (ECB)*	Y	Y	X	Y
7.	Metode *Systematic Literature Review* untuk Identifikasi Platform dan Metode Pengembangan Sistem Informasi di Indonesia	X	X	X	X
8.	Pengujian *Black Box* pada Aplikasi Sistem Informasi Akademik Menggunakan Teknik *Equivalence Partitioning*	Y	Y	Y	Y
9.	Pengujian *Black Box* pada Aplikasi Penjualan Berbasis Web Menggunakan Metode *Equivalents Partitions* (Studi Kasus: PT Arap Store)	Y	Y	Y	Y
10.	Pengujian *Software* dengan Menggunakan Model *Behaviour UML*	Y	X	X	X
11.	Pengujian Sistem Informasi Akademik menggunakan *Mccall's Software Quality Framework*	Y	Y	Y	Y
12.	Pengujian *Black Box* pada Aplikasi Penjualan Berbasis Web Menggunakan Teknik *Boundary Value Analysis*	Y	Y	Y	Y
13.	Pengujian pada Aplikasi Penggajian Pegawai dengan menggunakan Metode *Blackbox*	Y	Y	Y	Y
14.	Pengujian Perangkat Lunak Metode Black-Box Berbasis *Equivalence Partitions* pada Aplikasi Sistem Informasi Sekolah	Y	X	X	X
15.	*Grey-Box Technique of Software Integration Testing Based on Message*	Y	Y	X	Y
16.	*Black-Box Approach For Software Testing Based On Fat-Property*	Y	X	X	X
17.	*A Comparative Study of Software Testing Techniques Viz. White Box Testing Black Box Testing and Grey Box Testing*	Y	Y	X	Y
18.	*Software Testing Techniques: A Literature Review*	Y	X	X	X
19	Survei Teknik Pengujian *Software*	Y	X	X	X

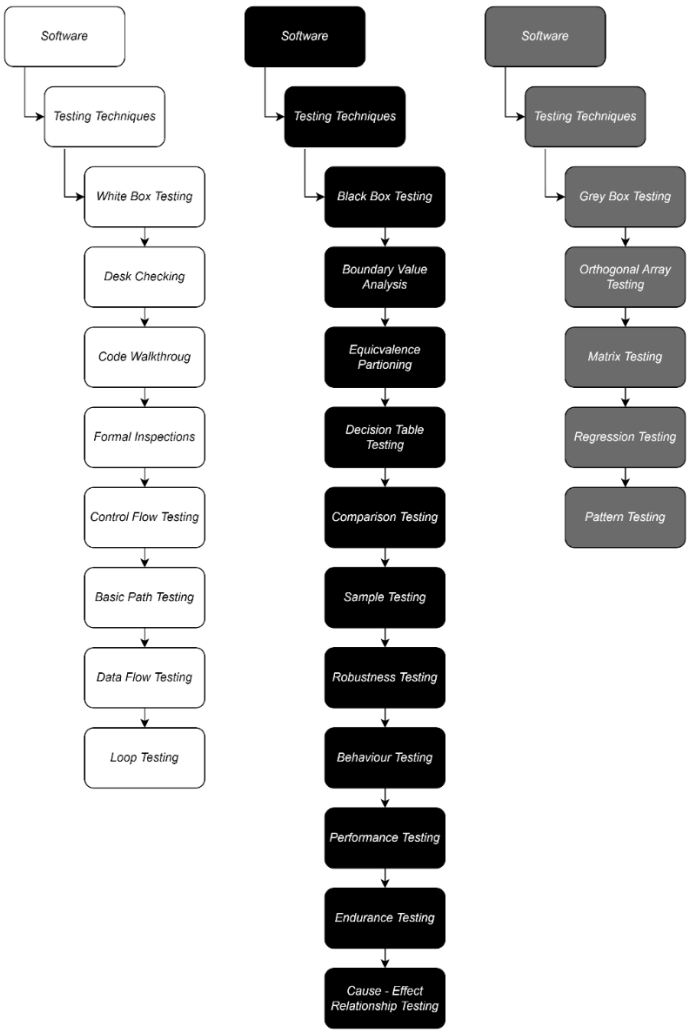
Keterangan Simbol:
Y: menunjukkan data (jurnal) yang dipakai untuk penelitian berdasarkan masalah, pendekatan, serta informasi yang cukup untuk memilih data.
X: menunjukkan data (jurnal) yang tidak dipakai untuk penelitian karena berupa artikel berasal dari *guest editor* yang ditulis berdasarkan pengalaman peneliti, masalah, pendekatan, dan informasi yang kurang mencukupi untuk melakukan pemilahan data.

Analisis Data (*Data Analysis*)

RQ1. Apa saja metode yang digunakan untuk pengujian *software*?

Terdapat 19 jurnal yang sudah melalui tahap *search process*. Kemudian data dipilah berdasarkan *inclusion and exclusion criteria* menggunakan *keyword* "teknik-teknik pengujian *software*", diperoleh 19 jurnal yang selanjutnya dinilai berdasarkan kualitas penilaian (*Quality Assesment*). Hasil ini menjawab RQ1 yang dijelaskan sebagai berikut :

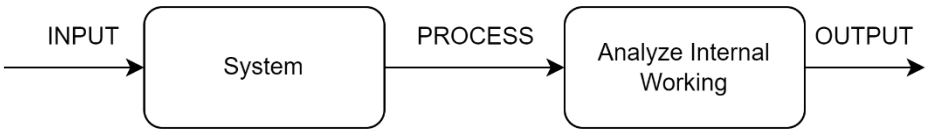
Setelah *search process* dilakukan, dapat disimpulkan bahwa *software testing* secara garis besar dibagi menjadi 3 yaitu *white box testing*, *black box testing*, dan *grey box testing*.



Gambar 3. Taksonomi Software Testing

1. *White Box testing*

White box testing atau teknik uji software dengan menggunakan runtutan logika program yang terkait pada *source code*. *White box testing* terbagi menjadi *basic path testing*, *data flow testing*, *control structure testing*, dan *loop testing*. *White box testing* dilakukan dengan mengikuti analisis internal kerja dan struktur *software*. *White box testing* adalah proses pemberian input ke dalam sistem kemudian memeriksanya bagaimana sistem tersebut memproses input kemudian menghasilkan output yang diinginkan. Hal ini diperlukan bagi penguji untuk memiliki pengetahuan tentang *source code*. *White box testing* berlaku pada integrasi, unit serta tingkat sistem dari proses pengujian software. Teknik uji *basic path testing* merupakan teknik pengujian yang paling sering digunakan.



Gambar 4. Alur kerja *white box testing*

a. *Desk Checking*
Desk Checking adalah salah satu pengujian yang sangat penting untuk dilakukan. Para pembuat *software* yang telah mempelajari bahasa pemrograman dengan sangat baik umum nya akan diikutsertakan dalam pengujian *Desk Checking*.
b. *Code Walkthrough*
Dalam proses pengujian ini, orang-orang yang melakukan nya ternasuk dalam kelompok khusus dimana pengujian ini merupakan salah satu jenis prosedur survei semi-formal.
c. *Formal Inspections*
Inspeksi formal adalah pengujian yang efektif, baik dan efisien untuk menemukan kesalahan dalam garis besar pada kode. Inspeksi formal bertujuan untuk membedakan semua masalah, pelanggaran, dan kesalahan lainnya.
d. *Control Flow Testing*
Pengujian ini merupakan metode pengujian yang terstruktur dengan menggunakan menggunakan aliran kontrol proyek sebagai aliran kontrol model dan membuat semua cara menjadi lebih mudah dan lebih tidak rumit.
e. *Basis Path Testing*
Basis Path Testing mengizinkan kreator yang suka mencoba untuk membuat ukuran kerumitan yang saling berhubungan dari garis besar prosedur kemudian setelah itu ukuran ini aklan digunakan sebagai metodologi untuk menggambarkan serangkaian cara eksekusi yang mendasar.
f. *Data Flow Testing*
Dalam pengujian ini, diagram aliran kontrol diklarifikasi dengan data tentang bagaimana variabel sistem dikarakterisasi dan digunakan.
g. *Loop Testing*
Pada pengujian ini hanya menyoroti legitimasi dari perkembangan *software*.

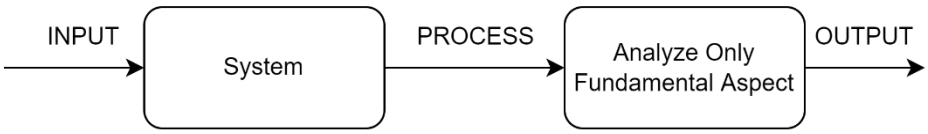
Ada beberapa keuntungan dan kerugian jika menggunakan *white box testing*, diantaranya adalah :

Tabel 6. Keuntungan dan kekurang *white box testing*

Keuntungan	Kekurangan
• Efek samping yang bermanfaat • Kesalahan pada kode yang tersembunyi dapat terlihat • Memperkirakan partisi yang telah dilakukan melalui eksekusi yang setara • *Developer* memberikan alasan detail mengenai implementasi yang dilakukan	• Membutuhkan biaya operasional yang mahal • Ketinggalan beberapa *case* yang hilang dalam kode

2. *Black Box testing*

Black Box Testing yaitu metode uji yang digunakan untuk mengobservasi hasil implementasi dari data uji dan pemeriksaan operasi *software*. Teknik pengujian *Black Box* mempunyai tujuan dalam mencari tahu jika terdapat kesalahan saat memasukkan data apakah bagian dalam sistem aplikasi akan menampilkan *error* tersebut, sehingga teknik uji ini menitik beratkan pada fungsi sistem. Black box testing dilakukan dengan mengikuti analisis spesifikasi software tanpa mengacu pada system internal-nya. *Black box testing* tidak berpengaruh terhadap struktur logis sistem internal karena *black box testing* hanya memeriksa aspek internal dari sistem. Untuk memastikan bahwa *input* yang diterima adalah benar dan *output* yang diberikan juga benar.



Gambar 5. Alur kerja *black box testing*

a. *Boundary Value Analysis*
Nilai yang diisikan pada aplikasi, yang secara umum dikerjakan dengan menjaga satu variabel berada pada nilai normal (nominal) serta tetap membolehkan variabel lain untuk dimasukkan dengan nilai maksimal. Nilai Minimum, Minimum + 1, Normal (nominal), Maksimum -1, dan Maksimum merupakan nilai merupakan nilai penguji keekstriman data.

b. *Equivalence Partioning*
Equivalence Partioning digunakan untuk mencari seluruh kesalahan atau kehilangan dalam fungsi. Kesalahan dapat tampilan struktur data atau akses menuju database serta performa. Dengan cara membagikan domain input ke kelas-kelas yang akan dijadikan kelas uji. Kelas tadi kemudian disiapkan sebagai kondisi input pada kasus uji yang berupa gabungan nilai-nilai benar dan tidak benar. Keadaan masukan bisa berupa range, harga khusus, suatu kumpulan atau boolean. Jika input merupakan beberapa keaadan tersebut, maka kasus ujinya adalah satu benar dan dua tidak benar.

c. *Decision Table Testing*
Pengujian gabungan dari berbagai kondisi dalam pengambilan keputusan, yang digunakan pada uji *software* dalam verifikasi *input* yang beragam tetapi saling menggenapi fungsi *form*.

d. *Comparison Testing*
Uji tiap versi, yang bertujuan untuk menjamin keseluruhan versi mendapatkan hasil yang sama dengan data uji yang sama. Kemungkinan penggunaan redunansi *hardware* dan *software* untuk mengurangi kesalahan

e. *Sample Testing*
Mengambil nilai-nilai yang dipilih dari suatu kelas yang sama. Dengan cara mengikutserjakan beberapa nilai yang terpilih dari data input kelas ekivalensi kemudian diintegrasikan ke kasus uji. Nilai tersebut dapat berwujud variabel limit *testing* atau konstanta. Selanjutnya kasus uji akan memproses titik *singular* (atau nilai batas). Nilai batas merupakan kelas ekivalensi yang menangkap nilai yang sama atau mirip dengan kelas tersebut. *Output* data dari kelas ekivalensi juga dilibatkan dalam pembatasan tes. Jika keadaan masukan adalah *range*, maka kasus ujinya adalah dengan cara menguji titik *singular range* dan nilai *invalid* yang mendekati titik *singular*. Namun jika keadaan masukannya merupakan harga khusus, kasus uji diharuskan memenuhi nilai *minimum* dan *maksimum.*

f. *Robustness Testing*
Data masukan diambil dari luar spesifikasi yang didefinisikan. *Robustness testing* ditujukan sebagai pembuktian bahwa tidak terdapat kesalahan apabila terdapat masukan yang tidak valid .

g. *Behaviour Testing*
Apabila hanya dilakukan sekali uji Hasil uji tidak dapat diniliai, namun akan dapat dinilai jika dilakukan pengulangan uji, seperti yang terdapat dalam pengujian struktur data *stack*.

h. *Performance Testing*
Performance testing merupakan pengujian berdasarkan observasi pada kinerja program, dengan dengan melihat gelombang data, ukuran penggunaan penyimpanan, serta masa implementasi. Digunakan untuk mengetahui beban kerja atau konfigurasi program dan dapat juga untuk menguji batasan lingkungan program

i. *Requirement Testing*
Spesifikasi kebutuhan *software* beserta identifikasi ditahap spesifikasi kebutuhan dan desain. Dengan cara yang pertama yaitu mengatur kasus uji untuk setiap kebutuhan yang berhubungan dengan modul atau *CSU*. Kemudian tiap kasus uji dapat diurutkan berdasarkan kebutuhan peragkat lunak dengan menggunakan matriksnya

j. *Endurance Testing*
Endurance testing menyertakan kasus uji yang diulang-ulang dengan kuantitas tertentu yang bertujuan untuk menguji program apakah sudah sesuai dengan spesifikasi yang dibutuhkan

k. *Cause – Effect Relationship Testing*
Pembagian spesifikasi menjadi bagian-bagian yang sesuai dengan kebutuhan dan kemungkinan kerja dengan cara bagilah spesifikasi kebutuhan menjadi bagian-bagian di mana ada peluang kerja kemudian tentukan *cause* dan *effect* berdasarkan spesifikasi kebutuhan lalu analisis spesifikasi kebutuhan untuk membuat koneksi logika. Tandai graf yang tidak mungkin terkait dengan kombinasi *cause*/*effect* dalam batas spesifikasi

Ada beberapa keuntungan dan kerugian jika menggunakan *black box testing*, diantaranya adalah :

Tabel 7. Keuntungan dan kekurangan *black box testing*

Keuntungan	Kekurangan
• *Black box tester* tidak memiliki relasi dengan kode • Presepsi penguji sangat sederhana • Penguji dan *developer*, keduanya berjalan dengan *independent* • Lebih efektif digunakan untuk eksekusi kode dalam skala besar	• Kasus uji sulit dirancang tanpa speisifikasi yang jelas • Hanya angka kecil yang kemungkinan dapat dimasukkan dan diuji • Beberapa bagian dari *back end* tidak diuji sama sekali

3. *Grey Box testing*

Grey box testing adalah metode gabungan antara *White box testing* dan *Black box testing*. *Grey box testing* digunakan untuk menguji perangkat lunak terhadap spesifikasinya tetapi menggunakan beberapa pengetahuan tentang kerja internalnya juga.

a. *Orthogonal Array Testing*
b. *Matrix Testing*
c. *Regression Testing*
d. *Pattern Testing*

Tabel 8. Keuntungan dan kekurangan *grey box testing*

Keuntungan	Kekurangan
• Memberikan keuntungan gabungan antara penemuan dan prosedur pengujian *White Box*. • Dalam pengujian *Box Testing*, penganalisis dapat merencanakan situasi pengujian yang luar biasa. • Pengujian tidak memihak • Membuat komposisi tes yang bagus	• Ruang lingkup pengujian dibatasi karena pintu masuk ke kode sumber tidak dapat diakses. • Banyak cara sistem yang belum teruji. • Eksperimen bisa berulang-ulang

RQ2. Apa metode yang paling sering digunakan dalam pengujian *software*?

Tabel 9. Hasil pengelompokkan metode yang menjawab RQ2

No.	Nama Metode Pengukuran *Usability Software*	Jumlah kemunculan
1.	*White Box*	9
2.	*Black Box*	13
3.	*Grey Box*	7

Dilihat dari Tabel 9, metode *Black Box* merupakan metode yang sering digunakan dalam mengukur *usability software.* Setelah penulis mengkaji beberapa metode yang ditemukan untuk melakukan *software testing*, dapat disimpulkan perbandingan dari ketiga teknik *software testing* yang penulis dapat seperti Tabel 10.

Tabel 10. Perbandingan teknik *software testing*

Black Box Testing	*White Box Testing*	*Grey Box Testing*
hanya menganalisa aspek fundamental (tidak ada	Pengetahuan penuh mengenai *internal working*	Sebagian pengetahuan mengenai *internal working*

pengetahuan mengenai *internal working*)		
Pengujian ini paling ringan dan hemat waktu	Berkemungkinan paling melelahkan dan memakan waktu	Berada di antara black box dan white box mengenai efektifitas dan efisiensi
Tidak cocok untuk tes algoritma	Cocok untuk tes algoritma	Tidak cocok untuk tes algoritma
Memiliki granularitas yang rendah	Memiliki granularitas yang tinggi	Memiliki granularitas yang sedang

RQ3. Apa saja Metode pengujian *software* yang paling memenuhi standarisasi ISO?

Pada penelitian ini standar ISO yang digunakan yaitu ISO 9126. ISO 9126 merupakan standar pengukuran yang digunakan untuk menguji perangkat lunak, Pengujian menggunakan standar ISO 9126 bertujuan untuk menguji dan mengevaluasi kekurangan, kelebihan, serta performa dari perangkat lunak tersebut. Pada ISO 9126 terdapat enam karakteristik dengan sub-karakteristik disetiap karakteristik. Berikut merupakan keenam karakteristik tersebut:

1. *Functionality*

Functionality atau fungsionalitas merupakan kemampuan *software* dalam menyediakan fungsi-fungsi yang sesuai dengan kebutuhan konsumen atau pengguna *software* tersebut. Dalam *functionality* terdapat 5 sub – karakteristik yaitu *suitability*, *accuracy*, *security*, *interoperability*, dan *compliance*.

Tabel 11. Functionality ISO

Sub - Karakteristik	Deskripsi
Suitability	Kemampuan *software* dalam mengerjakan tugas yang diberikan.
Accuracy	Kemampuan software dalam mengerjakan hasil yang akurat.
Security	Kemampuan software dalam memberikan keamanan yang baik.
Interoperability	Kemampuan software dalam melakukan hubungan atau interaksi dengan sistem yang lain.
Compliance	Kemampuan software dalam memenuhi standar yang dibutuhkan.

2. *Reliability*

Reliability atau realibilitas adalah konsistensi *software* dalam melakukan kinerja atau tugas tertentu.. Dalam *reliability* terdapat 3 sub – karakteristik yaitu *maturity*, *fault tolerance*, *recoverability*.

Tabel 12. Reliability ISO

Sub - Karakteristik	Deskripsi
Maturity	Kemampuan *software* dalam menghindari kesalahan yang dapat terjadi pada *software*.
Fault Tolerance	Kemampuan *software* dalam mentolerir kesalahan yang dapat terjadi pada *software*.
Recoverability	Kemampuan software dalam memulihkan sistem jika terdapat kesalahan yang dapat terjadi pada *software*.

3. *Usability*

Usability atau usabilitas adalah kemampuan *software* dalam menilai tingkat kepuasan pengguna terhdapat *software*. Dalam *usability* terdapat 4 sub – karakteristik yaitu *understandibility*, *learnability*, *operability*, dan *attractiveness*.

Tabel 13. Usability ISO

Sub - Karakteristik	Deskripsi
Understandibility	Kemudahan *software* untuk dimengerti oleh pengguna.
Learnability	Kemudahan *software* untuk didalami oleh pengguna.
Operability	Kemudahan pengoperasian *software* oleh pengguna.
Attractiveness	Tingkat atraktif *software* untuk menarik para pengguna.

4. *Efficiency*

Efficiency atau efisiensi merupakan kemampuan *software* dalam memberikan pelayanan yang baik bagi pelanggan berupa kinerja yang cepat, tepat dan sesuai dengan situasi dan kondisi. Dalam *efficiency* terdapat 2 sub – karakteristik yaitu *time behavior dan resource behavior*.

Tabel 14. Efficiency ISO

Sub - Karakteristik	Deskripsi
Time behavior	Kinerja *software* dalam memberikan waktu yang baik pada saat melakukan tugasnya.
Resource behavior	Kinerja *software* dalam menggunakan sumber daya yang dimiliki dengan baik pada saat melakukan tugasnya.

5. *Maintainability*

Maintainability merupakan kemampuan software dalam pemeliharaan seperti contoh pada saat salah satu spesifikasi *software* tidak sesuai dengan harapan dan ingin diubah. Dalam *maintainability* terdapat 4 sub – karakteristik yaitu *analyzability, changeability, stability, dan testability*.

Tabel 15. Maintability ISO

Sub - Karakteristik	Deskripsi
Analyzability	Kinerja *software* dalam melakukan analisis terhadap kesalahan yang terjadi pada *software*.
Changeability	Kinerja *software* dalam melakukan perubahan pada atribut tertentu *software*.
Stability	Kinerja *software* dalam mengurangi efek tambahan yang disebabkan perubahan atribut.
Testability	Kinerja *software* dalam percobaan perubahan.

6. *Portability*

Portability atau portabilitas merupakan kemampuan software untuk dapat dikirim ke media atau *environment* lain dan kemampuan *software* dalam melakukan penyesuaian terhadap media lain. Dalam *portability* terdapat 4 sub – karakteristik yaitu *adaptability, instalability, coexistence, dan replaceability*.

Tabel 16. Portability ISO

Sub - Karakteristik	Deskripsi
Adaptability	Kinerja *software* dalam melakukan penyesuaian pada media lain atau media yang baru.
Instalability	Kinerja *software* dalam melakukan pengunuduhan pada media lain.
Coexistence	Kinerja *software* dalam menggunakan sumber daya *software* lain dalam media yang sama.
Replaceability	Kinerja *software* dalam mengganti kinerja *software* lainnya

Dengan melihat karakteristik tersebut didapatkan jawaban mengenai RQ3 adalah metode *Black Box* dan *White Box*, karena kedua metode tersebut yang paling sesuai dengan standarisasi ISO, sedangkan *Gray Box* kurang efektif dalam penggunaannya dan kurang diminati karena akses kode yang terbatas.

Metode *Black Box* mempunyai beberapa keuntungan yang telah memenuhi standarisasi ISO antara lain :

a) Pengujian sangat cocok dan efisien untuk segmen atau unit kode besar.
b) Pengujian dapat dengan jelas memisahkan perspektif pengguna dari perspektif pengembang melalui pemisahan tanggung jawab QA dan pengembangan.
c) Tidak memerlukan akses kode.
d) Mudah dijalankan, dimana kasus uji dapat dirancang segera setelah spesifikasi selesai. Ini dapat diskalakan ke sejumlah besar penguji yang cukup terampil tanpa pengetahuan tentang implementasi, bahasa pemrograman, sistem operasi, atau jaringan.
e) Tidak diperlukan keterampilan pengkodean, penguji tidak perlu mengetahui bahasa pemrograman atau bagaimana perangkat lunak telah diimplementasikan

Untuk metode *White Box* sendiri unggul yang tentunya telah memenuhi standarisasi ISO dalam hal sebagai berikut :

a) Peningkatan efektivitas dalam pemeriksaan silang keputusan desain dan asumsi terhadap kode sumber dapat menguraikan desain yang kuat, tetapi implementasinya mungkin tidak selaras dengan maksud desain.
b) Mempunyai jalur Kode Lengkap, dimana semua jalur kode yang mungkin dapat diuji termasuk penanganan kesalahan, ketergantungan sumber daya, dan logika/aliran kode internal tambahan.
c) Identifikasi *error* dalam menganalisis kode sumber dan mengembangkan pengujian berdasarkan detail implementasi memungkinkan penguji menemukan kesalahan pemrograman dengan cepat.
d) Mengungkap cacat kode tersembunyi, dimana akses ke kode sumber meningkatkan pemahaman dan mengungkap perilaku tersembunyi yang tidak diinginkan dari modul program.
e) Pengujian dapat dimulai pada tahap sebelumnya. Seseorang tidak perlu menunggu *GUI* tersedia.

Sedangkan pada metode *Gray Box* dinilai kurang diminati penggunaanya karena beberapa alasan yakni :

a) Akses kode yang terbatas dimana, kode sumber atau binari tidak tersedia serta dalam melintasi jalur kode masih dibatasi oleh pengujian atau cakupannya tergantung pada keterampilan penulis dari penguji.
b) Identifikasi yang cacat, dimana pengujian menggunakan metode *Grey Box* ini masih bergantung pada kemampuan sistem dalam mengeliminasi pengecualian dan menyebarkannya pada sistem web terdistribusi.

KESIMPULAN

Kesimpulan yang dihasilkan berdasarkan pada hasil penelitian yakni metode yang dimanfaatkan dalam menguji *software* terdiri dari *White box testing, Black box testing,* dan *Grey box testing*. Adapun metode yang paling sering digunakan dalam pengujiaan *software* yaitu *Black box testing*, dimana metode ini terdiri dari beberapa teknik pengujian seperti *boundary value analysis, equivalence partioning, decision table testing, comparison testing, sample testing, robustness testing, behaviour testing, performance testing, requirement testing, endurance testing,* dan *cause – effect relationship testing*. Sedangkan untuk metode yang memenuhi standarisasi ISO harus memenuhi persyaratan seperti *functionality, reliability, usability, efficiency, maintability,* dan *portability* . Dari tiga metode tersebut, didapati dua metode yang memeuhi standarisasi ISO yakni metode *black box* dan *white box*. Metode *black box* mempunyai nilai efisiensi yang tinggi untuk unit kode besar, sangat mudah dijalankan karena kasus uji dapat dirancang segera setelah spesifikasi selesai, pengujian dengan metode ini juga efektif karena dapat dengan jelas memisahkan perspektif pengguna dari perspektif pengembang, tidak memerlukan keterampilan pengkodean dalam penggunaanya dan juga tidak memerlukan kode akses. Untuk metode *white box* mempunyai efektivitas dalam pemeriksaan silang keputusan desain dan asumsi terhadap kode sumber dapat menguraikan desain yang kuat, kode akses yang lengkap sehingga dapat dengan mudah mengungkap cacat kode tersembunyi, serta identifikasi *error* dalam menganalisis kode sumber dan mengembangkan pengujian berdasarkan detail implementasi, sehingga memungkinkan penguji menemukan kesalahan pemrograman dengan cepat. Sedangkan untuk metode *grey box* sendiri kurang memenuhi standarisasi ISO karena mempunyai kode akses yang terbatas dan identifikasi yang cacat.

REFERENSI

[1] A. R. Lubis, “Perangkat Lunak Komputer,” pp. 1–9, 2020.
[2] W. Wibisono and F. Baskoro, “PENGUJIAN PERANGKAT LUNAK DENGAN MENGGUNAKAN MODEL BEHAVIOUR UML Waskitho Wibisono , Fajar Baskoro,” *Juti*, vol. 1, no. 1, pp. 43–50, 2002.
[3] M. A. Ridwan and S. Rochimah, “Peningkatan Akurasi Prediksi Waktu Perbaikan Bug dengan Pendekatan Partisi Data,” *J. Sist. Inf. Bisnis*, vol. 8, no. 1, p. 76, 2018, doi: 10.21456/vol8iss1pp76-83.
[4] M. Razavian, B. Paech, and A. Tang, “Empirical research for software architecture decision making: An analysis,” *J. Syst. Softw.*, vol. 149, no. 2019, pp. 360–381, 2019, doi: 10.1016/j.jss.2018.12.003.
[5] L. C. Fauzi, “Review: Analisis Pengaruh Sertifikasi Iso Sebagai Sistem Manajemen Mutu Terhadap Kinerja Perusahaan,” *Farmaka*, vol. 17, no. 1, pp. 144–150, 2019.
[6] V. Febrian, M. R. Ramadhan, M. Faisal, and A. Saifudin, “Pengujian pada Aplikasi Penggajian Pegawai dengan menggunakan Metode Blackbox,” *J. Inform. Univ. Pamulang*, vol. 5, no. 1, p. 61, 2020, doi: 10.32493/informatika.v5i1.4340.
[7] A. Maulana, A. Kurniawan, W. Keumala, V. R. Sukma, and A. Saifudin, “Pengujian Black Box pada Aplikasi Penjualan Berbasis Web Menggunakan Metode Equivalents Partitions (Studi Kasus: PT Arap Store),” *J. Teknol. Sist. Inf. dan Apl.*, vol. 3, no. 1, p. 50, 2020, doi: 10.32493/jtsi.v3i1.4307.
[8] I. M. S. Ardana, “Pengujian Software Menggunakan Metode Boundary Value Analysis dan Decision Table Testing,” *J. Teknol. Inf. ESIT*, vol. 14, no. 11, pp. 40–47, 2019.
[9] D. Budgen, B. Kitchenham, S. Charters, M. Turner, P. Brereton, and S. Linkman, “Preliminary results of a study of the completeness and clarity of structured abstracts,” 2007, doi: 10.14236/ewic/ease2007.7.
[10] A. Rouf, “Pengujian Perangkat Lunak Dengan Menggunakan Metode White Box dan Back Box,” vol. vol 8 no1, pp. 1–7, 2012, [Online]. Available: http://www.ejournal.himsya.ac.id/index.php/HIMSYATECH/article/view/28/27.
[11] E. Rosi Subhiyakto and D. Wahyu Utomo, “Strategi, Teknik, Faktor Pendukung Dan Penghambat Pengujian Untuk Pengembang Perangkat Lunak Pemula,” *Semin. Nas. Teknol. Inf. dan Komun.*, vol. 2016, no. Sentika, pp. 2089–9815, 2016.
[12] D. W. L. Pamungkas and S. Rochimah, “Pengujian Aplikasi Web - Tinjauan Pustaka Sistematis,” *J.*

IPTEK, vol. 23, no. 1, pp. 17–24, 2019, doi: 10.31284/j.iptek.2019.v23i1.459.

[13] T. Snadhika Jaya, "Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik Negeri Lampung)," *J. Inform. J. Pengemb. IT*, vol. 03, no. 02, pp. 45–48, 2018.

[14] A. Y. Putra and I. Yuliani, "Perancangan Dan Pengujian Perangkat Lunak Kriptografi Gabungan Playfair Cipher Dan Electronic Code Book (ECB)," *Enter*, pp. 560–570, 2019, [Online]. Available: http://sisfotenika.stmikpontianak.ac.id/index.php/enter/article/view/932.

[15] E. Triandini, S. Jayanatha, A. Indrawan, G. Werla Putra, and B. Iswara, "Systematic Literature Review Method for Identifying Platforms and Methods for Information System Development in Indonesia," *Indones. J. Inf. Syst.*, vol. 1, no. 2, p. 63, 2019.

[16] D. B. Muslimin, D. Kusmanto, K. F. Amilia, M. S. Ariffin, S. Mardiana, and Y. Yulianti, "Pengujian Black Box pada Aplikasi Sistem Informasi Akademik Menggunakan Teknik Equivalence Partitioning," *J. Inform. Univ. Pamulang*, vol. 5, no. 1, p. 19, 2020, doi: 10.32493/informatika.v5i1.3778.

[17] A. Mulyanto, "Pengujian Sistem Informasi Akademik Menggunakan Mccall's Software Quality Framework," *JISKA (Jurnal Inform. Sunan Kalijaga)*, vol. 1, no. 1, pp. 47–57, 2016, doi: 10.14421/jiska.2016.11-07.

[18] M. Nurudin, W. Jayanti, R. D. Saputro, M. P. Saputra, and Y. Yulianti, "Pengujian Black Box pada Aplikasi Penjualan Berbasis Web Menggunakan Teknik Boundary Value Analysis," *J. Inform. Univ. Pamulang*, vol. 4, no. 4, p. 143, 2019, doi: 10.32493/informatika.v4i4.3841.

[19] M. Komarudin, "Pengujian Perangkat Lunak Metode Black-Box Berbasis Equivalence Partitions pada Aplikasi Sistem Informasi di Sekolah," *J. Mikrotik*, vol. 06, no. 3, pp. 02–16, 2016.

[20] M. TanLi, Y. Zhang, Y. Wang, and Y. Jiang, "Grey-box technique of software integration testing based on message," *J. Phys. Conf. Ser.*, vol. 2025, no. 1, 2021, doi: 10.1088/1742-6596/2025/1/012096.

[21] M. TanLi, Y. Jiang, X. Wang, and R. Peng, "Black-box approach for software testing based on fat-property," *MATEC Web Conf.*, vol. 309, p. 02008, 2020, doi: 10.1051/matecconf/202030902008.

[22] T. Hussain and S. Singh, "A Comparative Study of Software Testing Techniques Viz. White Box Testing Black Box Testing and Grey Box Testing," *Int. J. Adv. Comput. Sci. Appl.*, vol. 3, no. 6, pp. 1–8, 2015.

[23] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software testing techniques: A literature review," *Proc. - 6th Int. Conf. Inf. Commun. Technol. Muslim World, ICT4M 2016*, pp. 177–182, 2017, doi: 10.1109/ICT4M.2016.40.

[24] I. R. Dhaifullah, M. M. H, A. A. Salsabila, and M. A. Yakin, "Survei Teknik Pengujian Software," vol. 2, no. 1, pp. 31–38, 2022.

[25] K. Mohd. Ehmer, "Different forms of software testing techniques for finding errors," *Int. J. Comput. Sci. Issues*, vol. 7, no. 3, pp. 11–16, 2010.

[26] A. Bansal, "A Comparative Study of So ware Testing Techniques A Comparative Study of Software Testing Techniques," *IJCSMC J.*, vol. 3, no. 6, pp. 579–584, 2014, [Online]. Available: www.ijcsmc.com%0AInternational.