Analisis Perbandingan Performa Metode Pengujian *Black Box Equivalence Class Partition* dan *State Transition* pada Aplikasi Visit Techno

Hubertus Rino Augenio[1], Margareta Hardiyanti[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
hubertusrino@mail.ugm.ac.id
*Korespondensi: margareta.hardiyanti@ugm.ac.id;

Abstract - Testing is a crucial stage in software development. It plays a pivotal role in determining the viability of the developed software. Black-box testing is one of the methods used to assess the functionality of software. PT Cipta Sedya Digital operates a digital service line named Techno Center. Techno Center is currently developing a room reservation application called Visit Techno. During the development process of Visit Techno, there are shortcomings in the testing phase as it hasn't been conducted comprehensively across all features. Additionally, the lack of proper identification of suitable testing methods has resulted in suboptimal previous testing. This final project was conducted to provide a solution to Techno Center to enhance the testing process for greater effectiveness. The project utilized an analysis that compared black-box testing methods: Equivalence Class Partition (ECP) and State Transition (ST) in testing the Visit Techno application. Research findings indicate that the Test Case Failed percentage for ST was 13.51% and for ECP was 10.89%. Test Case Execution stood at 100% for ST and 95.05% for ECP. Test Cases Not Executed were at 0% for ST and 4.95% for ECP. The Fault Detection Rate was 10.8/hour for ST and 14.4/hour for ECP. Severity Point for ST was 7, whereas for ECP it was 5. The Average Percentage Fault Detection was 0.72 for ST and 0.67 for ECP. Based on these results, ST proves more effective compared to ECP for use in testing the Visit Techno application. The research outcomes are expected to provide insights to improve the efficiency of black-box testing for the Visit Techno application and similar applications being developed by PT CSDI.

Keywords – Software Testing, Black-box Testing, Equivalence Class Partition, State Transition

Intisari - Pengujian merupakan salah tahap yang penting dalam proses pengembangan perangkat lunak. Pengujian berperan dalam menentukan kelayakan dari perangkat lunak yang dikembangkan. *Black-box testing* merupakan salah satu metode pengujian yang dilakukan untuk mengetahui fungsionalitas perangkat lunak. PT Cipta Sedya Digital mempunyai lini layanan digital dengan nama Techno Center. Techno Center sedang mengembangkan aplikasi peminjaman ruang dengan nama Visit Techno. Pada proses pengembangan Visit Techno, masih terdapat kekurangan pada tahap pengujian karena belum dilakukan secara menyeluruh pada semua fitur. Kurangnya proses identifikasi metode pengujian yang sesuai juga menyebabkan pengujian yang dilakukan sebelumnya tidak berjalan dengan optimal. Proyek akhir ini dilaksanakan untuk memberikan solusi kepada Techno Center dalam meningkatkan proses pengujian agar lebih efektif. Proyek akhir ini menggunakan analisis perbandingan pengujian *black-box Equivalence Class Partition* (ECP) dan *State Transition* (ST) pada pengujian aplikasi Visit Techno. Hasil penelitian yang didapatkan pada perhitungan *Test Case Failed* ST 13,51% dan ECP 10,89%. *Test Case Executed* ST 100% dan ECP 95,05%. *Test Case Not Executed* ST 0% dan ECP 4,95%. *Rate of Fault Detection* ST 10,8/jam dan ECP 14,4/jam. *Severity Point* ST pada level 7 dan ECP pada level 5. *Average Percentage Fault Detection* ST 0,72 dan ECP 0,67. Berdasarkan hasil tersebut ST lebih efektif dibandingkan ECP untuk digunakan dalam pengujian aplikasi Visit Techno. Hasil penelitian diharapkan memberikan wawasan untuk meningkatkan efisiensi pengujian *black-box* aplikasi Visit Techno dan aplikasi sejenis yang sedang dikembangkan PT CSDI.

Kata kunci – Pengujian Perangkat Lunak, *Black-box Testing, Equivalence Class Partition*, *State Transition*

I. PENDAHULUAN

Pengujian perangkat lunak merupakan tahapan yang penting dalam *Software Development Life Cycle* (SDLC) untuk menentukan kelayakan dari sebuah *software* yang dikembangkan. Pengujian bertujuan untuk menunjukan kepada *end-user* bahwa *software* sudah memenuhi persyaratan yang diberikan dan menemukan adanya kesalahan yang tidak diinginkan [1]. Terdapat dua pendekatan dalam melaksanakan proses pengujian perangkat lunak yaitu *white-box* dan *black-box*. Pengujian *white-box* merupakan pengujian fungsional yang berfokus pada struktur internal aplikasi sehingga membutuhkan pengetahuan mengenai kode program, sedangkan *black-box* secara umum menguji aplikasi secara fungsional, tetapi tidak memperhatikan detail kode implementasi [2] . Pada proses pengujian *black-box*, terdapat beberapa tahapan yang dilakukan oleh seorang penguji, salah satu tahapan yang penting adalah pembuatan kasus uji atau *test case*. Pembuatan kasus uji merupakan tahapan yang penting dalam pengujian karena berpengaruh terhadap efektifitas dan efisiensi [3]. Pemilihan metode dalam pengujian *black-box* akan mempengaruhi kasus uji yang dihasilkan sehingga perlu disesuaikan dengan perangkat lunak yang diuji.

PT Cipta Sedya Digital Indonesia (CSDI) memiliki lini layanan berbasis digital dengan nama Techno Center yang melayani pengembangan layanan digital bagi perusahaan pembiayaan di bawah Astra Grup serta perusahaan lain yang memerlukan jasa terkait [4]. Techno Center sedang mengembangkan aplikasi peminjaman ruang bernama Visit Techno. Pada proses pengembangan Visit Techno pengujian dilaksanakan secara manual dan masih terdapat kekurangan. Kekurangan tersebut terdapat pada pengujian fungsional aplikasi Visit Techno yang belum dilakukan secara menyeluruh dan terbatas pada beberapa fitur. Selain itu, kurangnya proses identifikasi metode pengujian yang tepat terhadap fitur aplikasi Visit Techno menyebabkan pengujian yang dilakukan sebelumnya tidak berjalan dengan optimal.

Hal tersebut membuat aplikasi Visit Techno masih terdapat kekurangan dari segi fungsionalitas.

Oleh karena itu, agar pengujian fungsionalitas aplikasi Visit Techno dapat dilakukan dengan optimal, maka dilakukan analisis performa metode *black-box* untuk diimplementasikan pada pengujian aplikasi Visit Techno. Penggunaan *black-box* dalam pengujian dapat meningkatkan kesesuaian sistem dengan desain yang telah diterapkan [5]. Pada studi literatur yang dilakukan oleh [3] bahwa *Equivalence Class Partition* (ECP) dan *State Transition* (ST) merupakan teknik *black-box* yang banyak digunakan dalam pengujian fungsional. ECP memiliki karakteristik berdasarkan domain atau wilayah data *input*, sedangkan ST berbasis pendekatan spesifikasi perangkat lunak. Kedua metode tersebut cocok jika digunakan dalam pengujian fungsional aplikasi Visit Techno yang memiliki banyak elemen *input* dan interaksi komponen menu. Implementasi pengujian *automation* dengan Katalon Studio digunakan pada penelitian ini karena *open source* serta mendukung *record-replay* dan *script mode*. Melalui analisis perbandingan, diharapkan dapat menentukan penggunaan metode yang tepat antara ECP dan ST untuk melakukan pengujian pada aplikasi Visit Techno maupun aplikasi peminjaman sejenis yang dikembangkan PT CSDI.

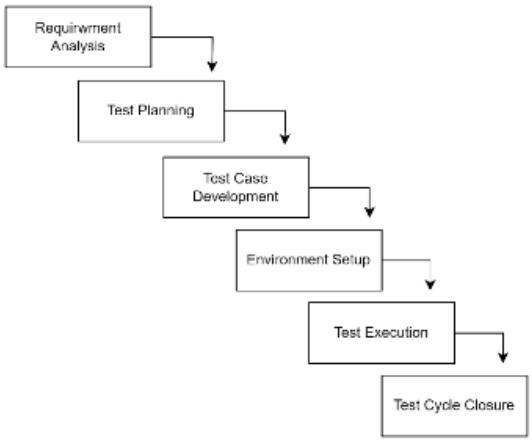
II. DASAR TEORI

A. *Software Testing*

Menurut [6] *software testing* adalah suatu proses atau serangkaian proses yang dirancang untuk memastikan kode komputer melakukan apa yang telah dirancang untuk dilakukan. Perangkat lunak yang baik adalah yang dapat memenuhi hasil yang telah dirancang pada awal tahapan. Penelitian [7] mendefinisikan konsep *testing* sebagai sebuah proses yang perlu untuk direncanakan pada tahap awal *development*. Penelitian [7] membagi *test*ing ke dalam empat tahap yaitu *planning, design, construction, maintenance, dan execution*. Tahap yang dikenalkan oleh Gelperin dan Hetzel disebut sebagai *prevention period* dan banyak diadaptasi untuk digunakan sampai saat ini.

B. *Software Testing Life Cycle*

Software Testing Life Cycle (STLC) adalah siklus hidup atau kerangka kerja yang digunakan dalam pengembangan perangkat lunak untuk merencanakan, merancang, mengelola, dan melaksanakan aktivitas pengujian dengan tujuan memastikan kualitas dan keandalan perangkat lunak. STLC adalah salah satu komponen kunci dalam siklus hidup pengembangan perangkat lunak secara keseluruhan. Setiap STLC memiliki *Entry Criteria*, *Exit Criteria*, *Activities* dan *Deliverable* [8]. Langkah dalam proses STLC seperti pada Gambar 1.



Gambar 1. *Software Testing Life Cycle*

C. *Black Box Testing*

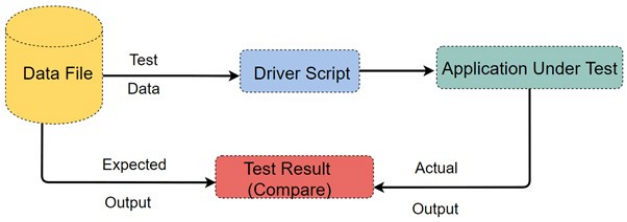
Black-box testing adalah metode pengujian perangkat lunak yang fokus pada pengujian fungsionalitas perangkat lunak tanpa memperhatikan struktur internal kode sumber atau desain perangkat lunak. Pengujian *black-box* juga dikenal dengan isitlah *data driven* atau *input/output-driven testing* [6]. Dalam metode pengujian ini, pengujian dilakukan dari perspektif eksternal, seperti yang akan dilakukan oleh pengguna akhir, dan pengujian dilakukan tanpa pengetahuan tentang bagaimana perangkat lunak diimplementasikan. Teknik yang digunakan dalam pengujian *black-box* berfokus pada domain informasi perangkat lunak, dengan cara membagi domain *input* dan *output* program sehingga memberikan cakupan pengujian yang menyeluruh.

D. *Test Case*

Test case atau uji kasus adalah serangkaian tindakan yang untuk melakukan verifikasi dan validasi pada fitur atau fungsi tertentu. *Test case* merupakan inti dari sebuah pengujian perangkat lunak. *Test case* terdiri dari komponen berupa identifikasi nomor, tujuan*, bisnis role,* deskripsi persyaratan, masukan aktual, keluaran yang diharapkan, kondisi akhir yang diharapkan, dan riwayat eksekusi [9].

E. *Data Driven Testing*

Data Driven Testing (DDT) adalah pendekatan pengujian perangkat lunak yang didasarkan pada penggunaan data eksternal sebagai *input* untuk menjalankan berbagai skenario pengujian [10]. Masukan pengujian dan hasil keluaran yang diharapkan disimpan dalam file data terpisah (biasanya dalam format tabel). Proses *data driven testing* dapat dilihat pada Gambar 2.



Gambar 2. Kerangka Kerja *Data Driven*

F. *Equivalence Class Partition*

Equivalence Class Partition merupakan salah satu teknik dasar dalam pengujian *black-box* yang membagi *domain input* menjadi beberapa kelas atau wilayah [6]. *Domain input* yang telah dibagi ke dalam kelas yang sesuai. Dalam teknik ECP, kelas ekuivalen dibentuk dengan mempertimbangkan kondisi *valid* dan *invalid*. Kondisi *valid* terjadi ketika *input* yang diberikan memberikan informasi yang sesuai. Kondisi *Invalid* terjadi ketika *input* memberikan informasi yang tidak sesuai.

G. *State Transition*

State transition merupakan teknik pengujian *black-box* berbasis *specification* atau model yaitu dengan menggunakan diagram UML untuk menghasilkan *test case*. Prinsip dari teknik ini adalah dengan mendefinisikan keadaan (*state*) dan perubahan (*transition*) dari persyaratan yang sudah ditentukan. *State Transition Diagram* menggambarkan semua keadaan yang dapat dimiliki suatu objek dimana objek tersebut mengubah keadaan (*transition*), melihat kondisi yang harus dipenuhi sebelum transisi terjadi (*guard*), dan aktivitas yang harus dipenuhi selama objek hidup (*action*) [11]. Penelitian [11] menjelaskan pembuatan *state diagram* dari *behavior model diagram* seperti *use case* dan *activity diagram.*

H. *Klasifikasi Bug*

Severity atau tingkat keparahan merupakan konsep untuk menunjukan tingkat keparahan dari kesalahan atau *bug* yang berpengaruh pada fungsionalitas seluruh sistem. Penelitian [12] membagi *severity* ke dalam beberapa level dibawah ini dari tingkat rendah ke tinggi. Kemudian menurut [13] karena tingkat keparahan berbeda antara sistem pelacakan *bug* maka dilakukan pemetaan klasifikasi *bug* berdasarkan *severity level* dengan skala 10 poin yang seragam. Pembagian skala tersebut dapat dilihat pada Tabel 1. Perhitungan *average severity* (1) menggunakan total *severity point* dan jumlah *bug*.

Tabel 1. Klasifikasi *bug* berdasarkan *severity level*

Poin	Tingkat Keparahan	Klasifikasi *Bug*
1	*Enhancement*	
2	*Trivial/Tweak*	*Low*
5	*Minor/Low/Small*	
6	*Normal/Medium*	*Medium*
7	*Major/High*	
9	*Critiacal/Clash*	*High*
10	*Blocker*	

$$Average\ Severity\ ST = \frac{Total\ Severity\ Point}{Jumlah\ Bug} \quad (1)$$

I. *Testing Metric*

Software Testing metric merupakan pengukuran kuantitatif untuk melihat estimasi , produktifitas, dan progres dari pengujian perangkat lunak. Tujuan dari adanya pengukuran melalui *metric* adalah menghasilkan data yang dapat dijadikan acuan perbandingan performa. Penggunaan *testing metric* sangat penting karena dapat meningkatkan efektivitas proses pengujian, berfungsi sebagai indikator tingkat efisiensi dan kebenaran, serta analisis metrik yang ditentukan [2]. Penelitian [14] melakukan pengukuran performa pengujian dengan menghitung *total percentage test case fail* (2), *percentage test case not executed* (3), dan *percentage test case executed* (4).

$$\text{TC } Failed\ (\%) = \left(\frac{Total\ TC\ Failed}{Total\ TC}\right) * 100\% \quad (2)$$

$$\text{TC } Executed\ (\%) = \left(\frac{Total\ TC\ Executed}{Total\ Test\ Case}\right) * 100\% \quad (3)$$

$$\text{TC } Not\ Executed\ (NE)\ (\%) = \left(\frac{Total\ TC\ NE}{Total\ TC}\right) * 100\% \quad (4)$$

J. *Rate of Fault Detection*

Penelitian [15] menjabarkan parameter *Rate of Fault Detection* atau RFT pada (5) untuk mengukur tingkat kesalahan yang berhasil ditemukan. RFT adalah jumlah kesalahan per satuan waktu yang diekspos oleh *test case*.

$$RFT = \left(\frac{Total\ number\ of\ faults}{Total\ time\ execution}\right) \quad (5)$$

K. *Fault Impact*

Fault Impact atau FI pada (6) digunakan untuk menghitung pengaruh kesalahan berdasarkan tingkat keparahan kesalahan tersebut [15]. Pada setiap nilai tingkat *keparahan* ditetapkan berdasarkan dampak yang disebabkan pada produk. FI dihitung dengan membagi jumlah *severity* yang ditemukan pada fitur atau *class* (si) dengan nilai *severity* tertinggi dari fitur atau *class* (Max(s)).

$$FI = \left(\frac{Si}{Max(S)}\right) * 10 \quad (6)$$

L. *Test Case Weight*

Test case weight atau TCW pada (7) digunakan untuk menentukan pentingnya suatu kasus uji dalam pengujian perangkat lunak. Bobot ini dapat digunakan untuk menentukan seberapa penting atau berat sebuah skenario uji dalam rangka untuk mengetahui seberapa baik perangkat lunak dapat beroperasi dalam situasi tersebut [15]. TCW dihitung dengan menjumlahkan *Rate of Fault Detection* dan *Faul Impact* pada masing-masing tes uji pada *class* atau fitur.

$$TCW = RFT + FI \quad (7)$$

M. *Average Percentage Fault Detection*

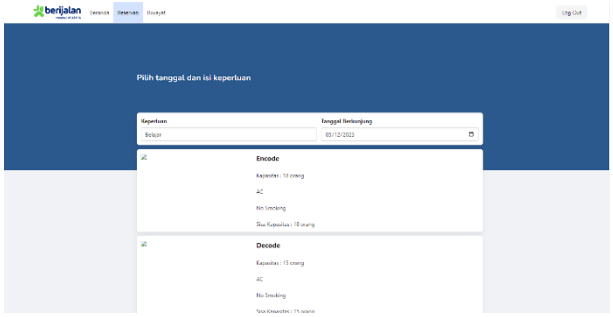
Average Percentege Fault Detection atau APFD merupakan acuan untuk melihat tingkat deteksi kesalahan berdasarkan persentase eksekusi rangkaian pengujian. APFD dihitung dengan mengambil rata-rata dari persentase kesalahan yang terdeteksi selama pelaksanaan rangkaian pengujian [16]. APFD dihitung memasukan data jumlah *test case* yang dijalankan (n), jumlah kesalahan yang berhasil ditemukan (m), dan posisi *test case* dimana terdapat kesalahan atau *fault* Fk (Pos(Fk)) pada (8).

$$APFD = \left(1 - \frac{\sum_{k=1}^{m} Pos(Fk)}{nm} + \frac{1}{2n}\right) \quad (8)$$

III. METODOLOGI

A. Objek Penelitian

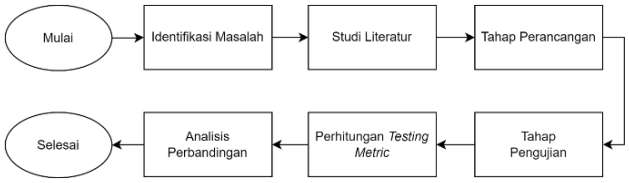
Bahan yang digunakan dalam penelitian ini melibatkan sistem peminjaman ruang berbasis aplikasi web bernama Visit Techno. Visit Techno digunakan sebagai objek yang akan diuji dengan menerapkan teknik ECP dan ST menggunakan pengujian *black-box*. Kemudian untuk data primer yang digunakan dari dokumentasi pengembangan projek aplikasi Visit Techno berupa *Technical Design Spesification* sebagai bahan penunjang dalam proses perancangan *test case*. Tampilan Aplikasi Visit Techno dapat dilihat pada Gambar 3.



Gambar 3.Tampilan aplikasi Visit Techno

B. Tahapan Penelitian

Alur penelitian dibagi ke dalam beberapa tahapan yang dimulai dari identifikasi masalah, studi literatur, tahapan perancangan, tahapan pengujian, perhitungan *testing metric*, dan terakhir analisis perbandingan. Pada tahapan perancangan dan pengujian dilakukan berdasarkan pada alur *Software Testing Life Cycle* (STLC) meliputi *requirement analysis, test planning, test case development, test environment setup*, dan *test case execution*. Tahapan proyek akhir secara garis besar dilakukan berdasarkan pada Gambar 4.

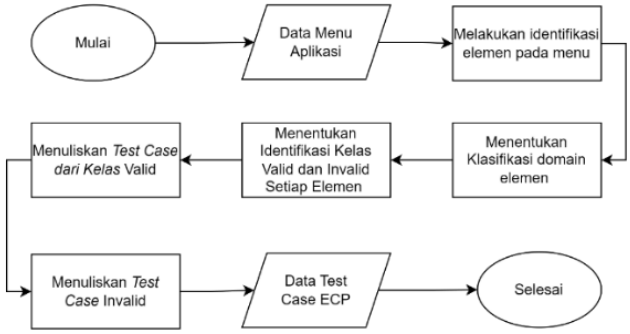


Gambar 4. Tahapan penelitian proyek akhir

C. Perancangan *Test Case*

1. Metode ECP

Pada Gambar 5 merupakan alur perancangan metode ECP. Tahapan dimulai dengan melakukan identifikasi menu, kemudian dilakukan melakukan klasifikasi wilayah domain dari masing-masing elemen seperti input, radio button, dsb. Klasifikasi dilakukan berdasarkan dua wilayah yaitu valid dan invalid. Contoh klasifikasi wilayah pada elemen menu autentikasi user admin pada tipe elemen input dapat dilihat pada Tabel 2. Hasil dari identifikasi wilayah valid dan invalid kemudian akan digunakan menjadi test case dari metode ECP.



Gambar 5. Diagram alir metode ECP

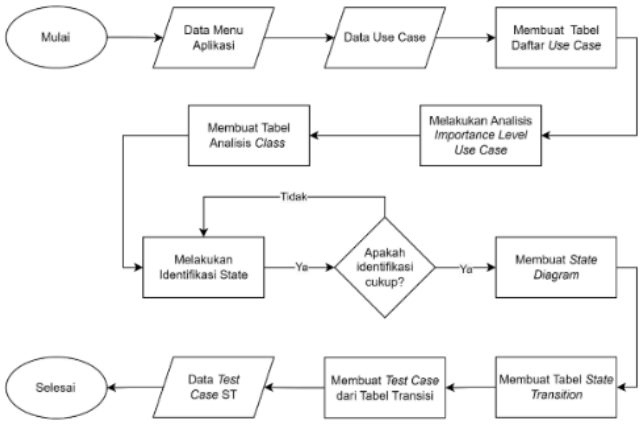
Tabel 2. Identifikasi Wilayah Autentikasi *User Admin*

Elemen	Domain	Valid	Invalid
Username	Panjang karakter	1 - 50	0
	Status akun	Terdaftar	Tidak terdaftar
Password	Panjang karakter	1 - 50	0
	Status akun	Terdaftar	Tidak terdaftar

2. Metode ST

Gambar 6 menunjukkan alur perancangan metode ST yang dimulai dengan analisis *use case* dari *technical document specification*. *Use case* dikelompokkan berdasarkan tingkat kepentingan, dan hasil analisisnya digunakan untuk membentuk *Class*. Identifikasi perilaku dasar seperti manipulasi data dilakukan pada setiap *class* untuk menentukan komponen *state*. Setiap *state* pada *class* digunakan untuk memodelkan perilaku sistem. Setelah semua *state* tercakup, dibuatlah *state transition diagram*. Informasi dari *state transition diagram* diubah menjadi tabel *state*

transition. Contoh tabel *state transition user* admin dapat dilihat pada Tabel 3.



Gambar 6. Diagram alir metode ST

Tabel 3. *State transition class user* admin

State Awal	*Event*	*Guard*	*Action*	State Akhir
S1	Masuk akun	Isian lengkap	Masuk akun	S2
S2	Akun valid	-	Setujui *login*	S3
S3	Masuk sistem	-	Tampilkan *homepage*	S4
S4	Keluar sistem	-	*Logout*	S5
S5	Berhasil keluar	-	Tampilkan *login* menu	S6
S2	Akun tidak valid	-	Tolak *login*	S7

IV. HASIL DAN PEMBAHASAN

Pengujian *automation* pada aplikasi Visit Techno dilakukan dengan menggunakan *tools* Katalon Studio. Pada setiap hasil *test case* yang sudah dibentuk dilakukan implementasikan ke dalam pengujian *automation*. Hasil pengujian yang telah dilakukan dengan *automation testing* menghasilkan pengelompokan *test case* yang dibagi menjadi *pass*, *failed*, dan *not executed*. Kemudian dilakukan perhitungan *metric* untuk mendapatkan hasil perbandingan performa ECP dan ST.

A. Hasil Pengujian

1. ECP

Pengujian yang dilaksanakan menghasilkan 101 *test case*. Total waktu pengujian sebesar 2.750,4 detik atau 45 menit 48 detik (0,73 jam). Hasil rangkuman dapat dilihat pada Tabel 4.

Tabel 4. Hasil rangkuman pengujian ECP

Menu/*Class*	*Pass*	*Fail*	*Not Executed*	Total	*Time (s)*
Registrasi *User* Pengunjung	36	7	3	46	396,9
Masuk Akun Pengunjung	7	0	0	7	185,6
Peminjaman	14	0	1	15	677,8
Masuk Akun Admin	8	0	0	8	165,1
User Management	2	1	0	3	110,4
Room Management	8	2	1	11	442,2
Room Management	10	1	0	11	772,2
Total	**85**	**11**	**5**	**101**	**2750, 4**

2. ST

Pengujian yang dilaksanakan menghasilkan 74 *test case* Total waktu eksekusi pengujian sebesar 3.371,829 detik atau 55 menit 6 detik (0,94 jam). Hasil rangkuman dapat dilihat pada Tabel 5.

Tabel 5. Hasil rangkuman pengujian ST

Menu/*Class*	*Pass*	*Fail*	*Not Executed*	*Total*	*Time (s)*
Ruangan	9	2	0	11	629,6
Peminjaman	15	2	0	17	1021
Riwayat Peminjaman	5	4	0	9	451,3
User Pengunjung	15	0	0	15	604,9
User Admin	7	0	0	7	303,8
User Management	13	2	0	15	361,3
Total	**64**	**10**	**0**	**74**	**3371, 3**

B. Hasil Identifikasi Kesalahan

1. ECP

Bug tersebut didapatkan dari hasil *test case fail*. Identifikasi *bug* pada metode ECP dapat dilihat pada Tabel 6. Hasil perhitungan *average severity* ECP sebesar 4,8 yang termasuk klasifikasi *bug low*.

Tabel 6. Identifikasi *bug* ECP

Menu/Class	Detail *Bug*	*Severity Level*
Registrasi *User* Pengunjung	Menyimpan *input* NIK selain angka (huruf dan simbol)	*Trivial* (2)
	Menyimpan *input* nomor HP lebih dari 13 karakter	*Minor* (5)
	Menyimpan *input* nomor HP dalam huruf dan simbol	*Trivial* (2)
	Menyimpan input nomor HP dengan spasi (setiap dua angka)	*Trivial* (2)
	Menyimpan *input* nomor HP dengan karakter *dash* (-) (setiap 4 angka)	*Trivial* (2)
	Menyimpan *input* nomor HP tanpa diawali dengan '08'	*Minor* (5)
	Tidak sukses menyimpan *input* keahlian lainnya	*Critical* (9)
Pencarian *User*	Menyimpan *input* karakter kolom *search* lebih dari 50 karakter	*Minor* (5)
Tambah Ruangan	Menyimpan kapasitas ruangan dengan nilai < 0	*Major* (8)
	Menyimpan kapasitas ruangan dengan nilai > 20	*Major* (8)
Edit Ruangan	Menyimpan nama ruangan lebih dari 50 karakter	*Minor* (5)
	Total Severity	**53**

$$Average\ Severity = \frac{53}{11} = 4{,}8$$

2. ST

Bug tersebut didapatkan dari hasil *test case fail*. Identifikasi *bug* pada metode ECP dapat dilihat pada Tabel 7. Hasil perhitungan *average severity* ST sebesar 7,3 yang termasuk klasifikasi *bug high*.

Tabel 7. Identifikasi *bug* ST

Menu/Class	Detail *Bug*	*Severity Level*
Ruangan	Sistem menonaktifkan ruangan yang sedang dipesan tanpa ada peringatan	*Medium* (6)
Peminjaman	Sistem dapat memberikan pemesenan kurang dari tanggal di hari *user* pinjam	*Critical* (9)
	Sistem menampilkan ruangan sama yang sudah dipesan oleh *user* lain pada tanggal sama	*Major* (8)
	Sistem memberikan akses pembatalan pada tanggal saat pemesanan berlangsung	*Major* (8)
	Sistem memberikan akses pembatalan pemesanan grup pada tanggal saat pemesanan berlangsung	*Major* (8)
Riwayat Peminjaman	Sistem membatalkan pemesanan grup pada salah satu akun, akun lain yang tercantum tidak terhapus	*Medium* (6)
	Sistem tidak memberikan akses untuk melakukan pemesanan kembali ruangan yang sudah dibatalkan pada pemesanan grup	*Medium* (6)
User Management	Sistem tidak menampilkan foto KTP *user* yang telah diunggah	*Major* (8)
	Sistem tidak menampilkan foto KTP *user lain* yang telah diunggah	*Major* (8)
	Total Severity	**73**

$$Average\ Severity = \frac{73}{10} = 7{,}3$$

C. Hasil *Testing Metric*

Hasil pengujian kemudian dihitung berdasarkan testing metris yang telah ditentukan. *Testing metric* menggambarkan persentase kinerja metode ketika digunakan dalam pengujian aplikasi.

1. Test *Case Failed*

$$TC\ Failed\ ECP\ (\%) = \left(\frac{11}{101}\right) * 100\% = 10{,}89\%$$

$$TC\ Failed\ ST\ (\%) = \left(\frac{10}{72}\right) * 100\% = 13{,}51\%$$

2. *Test Case Executed*

$$TC\ Executed\ ECP\ (\%) = \left(\frac{96}{101}\right) * 100\% = 95\%$$

$$TC\ Executed\ ST\ (\%) = \left(\frac{74}{74}\right) * 100\% = 100\%$$

3. *Test Case Not Executed*

$$TC\ Not\ Executed\ ECP\ (\%) = \left(\frac{5}{101}\right) * 100\% = 4{,}95\%$$

$$TC\ Not\ Executed\ ST\ (\%) = \left(\frac{0}{101}\right) * 100\% = 0\%$$

D. Hasil *Rate of Fault Detection*

$$RFTi\ ECP = \left(\frac{11}{0{,}76}\right) = 14{,}4\ bug/jam$$

$$RFTi\ ST = \left(\frac{10}{0{,}94}\right) = 10{,}6\ bug/jam$$

E. Hasil *Average Percentage Fault Detection*

Pada hasil pengujian dilakukan pengelompokan *bug* dalam *fault matrix*. Kemudian *test case* pada menu/*class* diurutkan berdasarkan *Test Case Weight* (TCW) terbesar sampai terkecil.

1. ECP

Fault matrix pada ECP dapat dilihat pada Tabel 8. Data *fault matrix* digunakan dalam perhitungan APFD. Hasil perhitungan menunjukan pada ECP menghasilkan nilai APFD 0,67.

Tabel 8. *Fault matrix* ECP

Test Case /Faults	T1-T46	T47-T57	T58-T60	T61-T71	T72-T78	T79-T93	T94-T101
F1	T5						
F2	T17						
F3	T18						
F4	T20						
F5	T22						
F6	T24						
F7	T41						
F8		T50					
F9		T52					
F10			T60				
F11				T63			

$$APFD\ ECP = \left(1 - \frac{372}{1111} + \frac{1}{202}\right) = 0{,}67$$

2. ST

Fault matrix pada ST dapat dilihat pada Tabel 9. Data *fault matrix* digunakan dalam perhitungan APFD. Hasil perhitungan menunjukan pada ST menghasilkan nilai APFD 0,72.

Tabel 9. *Fault matrix* ST

Test Case /Faults	T1-T9	T10-T26	T27-T41	T42-T52	T53-T67	T68-T74
F1	T3					
F2	T7					
F3	T8					
F4	TC9					
F5		T13				
F6		T15				
F7			T30			
F8			T34			
F9				T46		
F10				T50		

$$APFD\ ST = \left(1 - \frac{215}{740} + \frac{1}{148}\right) = 0{,}72$$

F. Analisis Perbandingan Hasil

Pada hasil pengujian aplikasi Visit Techno dengan metode ECP dan ST diperoleh masing-masing hasil dari perhitungan *metric*. Perbandingan hasil dapat dilihat pada Tabel 10.

Tabel 10. Perbandingan hasil ECP dan ST

Hasil Pengukuran	**ECP**	**ST**
Percentage Test Case Failed	10,89%	13,51%
Percentage Test Case Executed	95,05%	100%
Percentage Test Case Not Executed	4,95%	0%
Rate of Fault Detection	14,4	10,8
Average Severity Point	4,8	7,3
Average Percantege Fault Detection	0,67	0,72

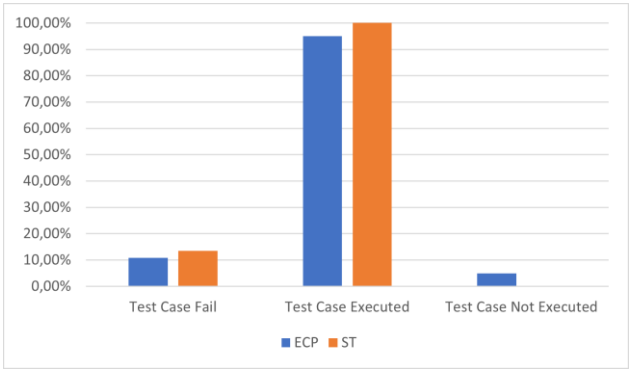
Hasil dari perbandingan diperoleh bahwa pada perhitungan *Percentage Test Case Failed* ST lebih banyak mengungkap kesalahan pada aplikasi dengan persentase 13,51% berbanding 10,89%. Pada perhitungan *Percentage Test Case Executed* ST lebih banyak menghasilkan *test* case yang berhasil dieksekusi dengan 100% kasus uji yang dijalankan dibandingkan dengan ECP dengan persentase 95,05%.

Kemudian perhitungan *Percentage Test Case Not Executed* metode ST unggul karena persentase yang dihasilkan lebih kecil dengan nilai 0% tanpa ada *test* case yang tidak dijalankan, dibandingkan dengan metode ECP yang menyisakan 4,95% *test* case yang tidak dijalankan. Pada perhitungan *Rate of Fault Detection* ECP lebih unggul karena menghasilkan *rate* nilai 14 *bug* yang berhasil dideteksi dalam satu jam dibandingkan dengan ST yang mendeteksi 11 *bug* dalam waktu satu jam. Pada perhitungan tingkat keparahan atau *severity point* ECP mendapatkan skor 4,8 yang termasuk ke dalam kategori rendah , sedangkan ST mendapatkan skor 7,3 yang termasuk ke dalam kategori tinggi. Perhitungan yang

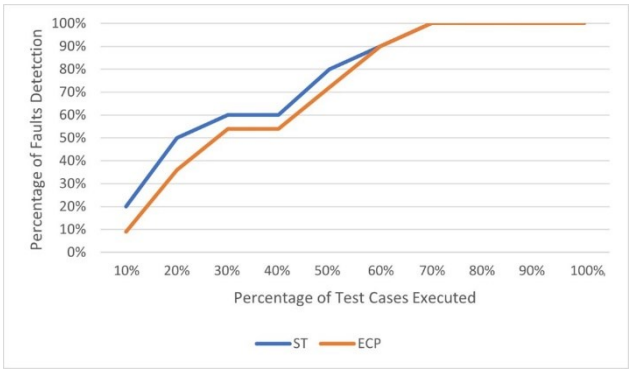
terakhir yaitu APFD dimana metode ST lebih unggul sebesar 0,72 daripada metode ECP sebesar 0,67.

G. Grafik Perbandingan Hasil

Grafik persentase hasil pengujian pada Gambar 7 dibuat untuk menunjukan perbandingan performa pengujian Visit Techno dengan metode ECP dan ST. Kemudian berdasarkan hasil perhitungan APFD didapatkan perbandingan grafik persentase APFD ECP dan ST yang dapat dilihat pada Gambar 8. Pada awal grafik, ST mendeteksi 20% kesalahan dari ECP yang sebesar 10% saja pada eksekusi *test case* sebanyak 10%. Pada grafik tersebut ST selalu unggul dari ECP sampai persentase *fault detection* mencapai 100%. Hal tersebut sesuai dengan perbandingan perhitungan APFD ST lebih unggul dari ECP.



Gambar 7. Grafik perbandingan persentase performa ECP dan ST



Gambar 8. Grafik perbandingan APFD ECP dan ST

V. SIMPULAN

ECP yang berfokus pada elemen pengujian *input* menghasilkan kasus uji sebanyak 101 buah, sedangkan ST yang berfokus pada perilaku pengguna terhadap respon sistem menghasilkan kasus uji 74 buah. Dalam menentukan metode yg efektif dari ST dan ECP, dapat dilihat dari hasil perhitungan *percentage test case fail, test case executed*, dan *average percentage of fault detection*. Hasil perhitungan tersebut menunjukan dengan metode ST menghasilkan angka yang tinggi daripada ECP sehingga metode ST lebih unggul.

Pada perhitungan persentase *not executed* dengan metode ST menghasilkan angka yang kecil daripada ECP sehingga metode ST lebih unggul. Berdasarkan hal tersebut metode ST lebih efektif dibandingkan dengan ECP untuk digunakan dalam pengujian aplikasi Visit Techno. Kemudian pada hasil identifikasi ECP berhasil menemukan *bug* dalam klasifikasi *low* pada komponen input sedangkan ST berhasil menemukan *bug* dalam klasifikasi *high* berupa respon sistem terhadap aksi atau perintah yang diberikan oleh *user*. Berdasarkan hasil tersebut, *aplikasi* Visit Techno perlu melakukan perbaikan kembali karena masih ditemukan *bug* dalam kategori tinggi maupun rendah.

VI. UCAPAN TERIMA KASIH

Terima kasih kepada PT Cipta Sedya Digital Indonesia (CSDI) atas fasilitas dan dukungan dalam penelitian ini. Terakhir terima kasih kepada kolega, teman-teman, dan individu lain yang turut berkontribusi dalam menyempurnakan penulisan jurnal ini.

REFERENSI

[1] I. Sommerville, Software engineering. Pearson, 2011.

[2] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software testing techniques: A literature review," in *Proceedings - 6th International Conference on Information and Communication Technology for the Muslim World, ICT4M 2016*, Institute of Electrical and Electronics Engineers Inc., Jan. 2017, pp. 177–182. doi: 10.1109/ICT4M.2016.40.

[3] N. Setiani, R. Ferdiana, P. I. Santosa, and R. Hartanto, "Literature review on test case generation approach," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jan. 2019, pp. 91–95. doi: 10.1145/3305160.3305186.

[4] F. Ardi and H. P. Putro, "Pengujian Black Box Aplikasi Mobile Menggunakan Katalon Studio (Studi Kasus: ACC Partner PT. Astra Sedaya Finance)," *AUTOMATA*, vol. 2, no. 1, 2021.

[5] Y. F. Achmad and A. Yulfitri, "PENGUJIAN SISTEM PENDUKUNG KEPUTUSAN MENGGUNAKAN BLACK BOX TESTING STUDI KASUS E-WISUDAWAN DI INSTITUT SAINS DAN TEKNOLOGI AL-KAMAL," 2020.

[6] G. J. Myers, Badgett Tom, and Sandler Corey, *The Art of Software Testing Third Edition*, Third Edition. Hoboken, New Jersey: John Wiley & Sons, 2012.

[7] D. Gelperin and B. Hetzel, "The Growth of Software Testing," *Commun ACM*, vol. 31, pp. 687–695, Sep. 1988, doi: 10.1145/62959.62965.

[8] Auliyaa Tri Nur, "Software Testing Life Cycle." Accessed: Sep. 18, 2023. [Online]. Available: https://sis.binus.ac.id/2020/07/06/software-testing-life-cycle/

[9] P. C. Jorgensen, *Software Testing Fourth Edition A Craftsman's Approach*, 4th edition. Boca Raton, Fla: Auerbach Publications, 2013.

[10] A. Lu, W. Fang, C. Xu, S. C. Cheung, and Y. Liu, "Data-driven testing methodology for RFID systems," *Front Comput Sci China*, vol. 4, no. 3, pp. 354–364, 2010, doi: 10.1007/s11704-010-0387-6.

[11] R. S. Presman, *Software Engineering: A Practitioner's Approach*, Seventh Edition. New York: McGraw-Hill COmpanies, 2010. [Online]. Available: www.mhhe.com/pressman.

[12] B. Kucuk and E. Tuzun, "Characterizing Duplicate Bugs: An Empirical Analysis," in *Proceedings - 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2021*, Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 661–668. doi: 10.1109/SANER50967.2021.00084.

[13] B. Zhou, I. Neamtiu, and R. Gupta, "Experience report: How do

bug characteristics differ across severity classes: A multi-platform study," in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, 2015, pp. 507–517. doi: 10.1109/ISSRE.2015.7381843.

[14] I. G. S. Aryandana, A. E. Permanasari, and T. B. Adji, "Comparing method equivalence class partitioning and boundary value analysis with study case add medicine module," in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Jan. 2020. doi: 10.1088/1757-899X/732/1/012072.

[15] K. Gopinath and D. Sureshkumar, "Test Case Prioritization for Regression Testing based on Severity of Fault," *International Journal on Computer Science and Engineering*, vol. 2, Aug. 2010.

[16] R. Pradeepa and K. Vimala Devi, "Effectiveness of Testcase Prioritization using APFD Metric: Survey," *Int J Comput Appl*, pp. 975–8887, 2013.