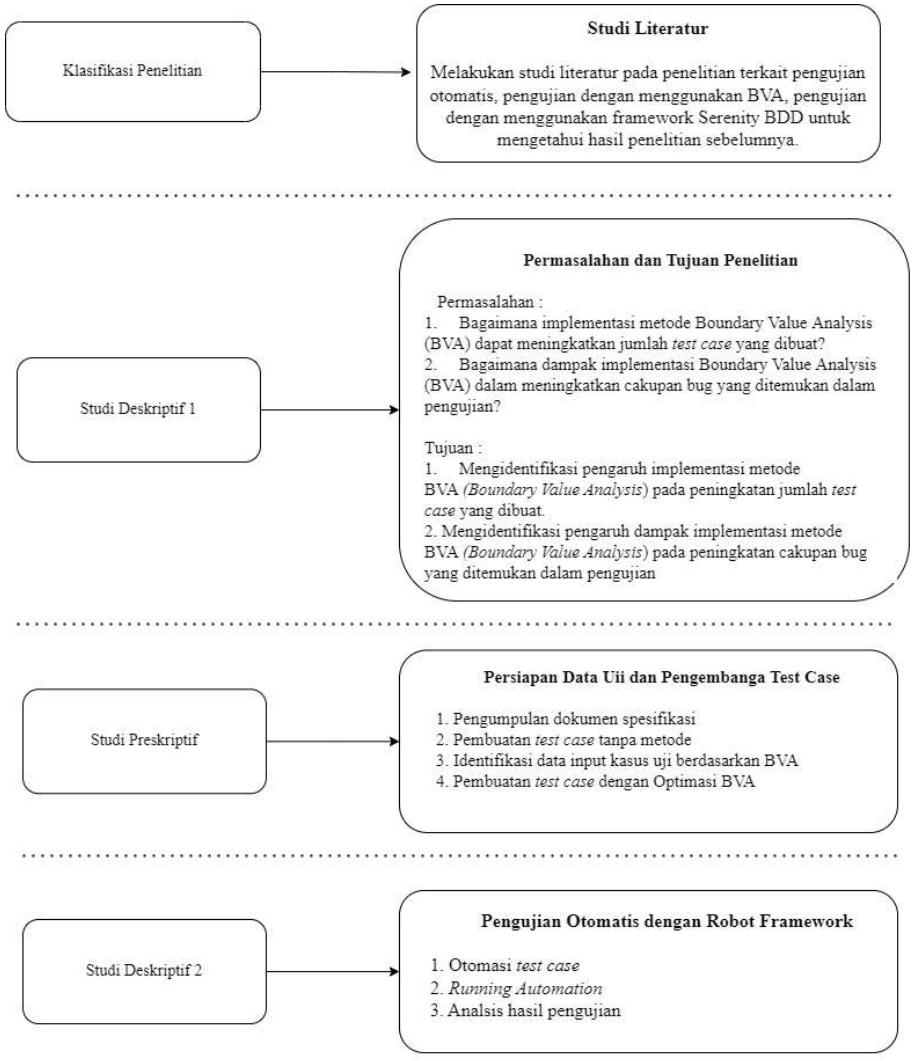
BAB III
METODE PENELITIAN

3.1 Metode Penelitian

Pada penelitian ini dirancang dengan mengadopsi metode *Design Research Methodology* (DRM), yang dikembangkan oleh Blessing dan Chakrabarti, melibatkan pendekatan terstruktur dalam penelitian ilmu desain yang terdiri dari empat tahapan: Klarifikasi Penelitian, Studi Deskriptif 1, Studi Preskriptif, dan Studi Deskriptif 2. Metodologi ini digunakan untuk memahami, mengeksplorasi, dan meningkatkan praktik desain secara sistematis dengan menghasilkan pengetahuan deskriptif dan preskriptif (Ebneyamini, 2022). Gambar 3.1 menunjukan skema penelitian yang dirancang berdasarkan DRM.



Gambar 3.1 Desain Penelitian

Berdasarkan desain penelitian pada gambar 3.1 berikut penjelasan dari masing-masing tahapan :

1. Klarifikasi Penelitian

Pada tahap ini, studi literatur dilakukan dengan tujuan untuk mengidentifikasi penelitian-penelitian sebelumnya yang berkaitan dengan pengujian otomatis, penggunaan *Robot Framework* pada pengujian otomatis, penggunaan metode BVA (*Boundary Value Analysis*) serta optimasi *test case* untuk menghilangkan redudansi sebagai bahan rujukan, baik berupa jurnal hingga buku-buku yang relevan.

2. Studi Deskriptif 1

Tahap berikutnya, mendefinisikan permasalahan dan tujuan penelitian berdasarkan pemahaman yang diambil dari studi literatur. Permasalahan merujuk pada rangkaian pertanyaan yang akan dipecahkan. Sementara tujuan penelitian akan mencerminkan hasil yang diharapkan pada penelitian yang dilakukan yaitu peningkatan jumlah *test case* yang dihasilkan denga Optimasi BVA dan peningkatan jumlah temuan *bug*.

3. Studi Perspektif

Pada tahap ini akan mengimplementasikan langkah-langkah pengembangan desain *test case* menggunakan metode BVA yang kemudian akan di optimasi untuk menghilangkan redudansi pada *test case* hasil BVA. Pada tahap pertama dilakukan pengumpulan dokumen spesifikasi perangkat lunak yang berasal dari hasil analisa kebutuhan perangkat lunak. Selanjutnya dilakukan pembuatan *test case* berdasarkan dokumen spesifikasi tanpa menggunakan metode. Pembuatan *test case* tanpa metode ini dibuat tanpa menghiraukan batasan *input*, sehingga hanya berfokus pada pengujian fungsional bagaimana setiap *field* menerima data *input*.

Selanjutnya masuk tahap ketiga yaitu identifikasi data *input* kasus uji berdasarkan metode *Boundary Value Analysis* yaitu dengan mencari nilai batas-batas pada setiap data *input*, mulai dari nilai di bawah batas bawah, batas bawah, batas atas dan nilai di atas batas atas. Setelah proses identifikasi kasus uji selesai, maka akan didapatkan batasan input dari setiap *field* untuk dijadikan acuan pembuatan *test case* dengan implementasi optimasi BVA. Optimasi dilakukan untuk menghilangkan redudansi dari hasil BVA.

4. Studi Deskriptif 2

Pada tahap terakhir ini, semua *test case* yang sudah dibuat akan diotomasi menggunakan *Robot Framework* sebagai alat penunjang pengujian otomatis. Setelah dilakukan otomasi pada semua *test case* maka *automation testing* sudah bisa dijalankan. Dari pengujian otomatis yang dijalankan dengan *Robot Framework* akan langsung menghasilkan *report* yang di dalamnya mendokumentasikan keberhasilan dan kegagalan *test case*. Hasil temuan *bug* akan diukur dan didokumentasikan untuk di analisa jumlah temuan *bug.*

3.2 Instrumen Penelitian

Instrumen penelitian yang digunakan untuk penelitian ini adalah analisis hasil pengujian. Analisis ini bertujuan untuk mengumpulkan dan menganalisis data dari hasil eksekusi *test case.* Data ini digunakan untuk mengevaluasi efektivitas metode optimasi *Boundary Value Analysis* (BVA) dalam mendeteksi cacat perangkat lunak serta untuk menilai efisiensi keseluruhan proses pengujian otomatisasi. Selain itu, penelitian ini juga telah melakukan validasi pada ahli media untuk dijadikan tolak ukur bahwa *test case* yang dibuat menggunakan optimasi BVA telah sesuai dengan standar *test case* yang dapat mencakup lebih banyak cakupan pengujian dan dapat menghasilkan temuan *bug* lebih banyak selama pengujian.

3.2.1 Analisis Pengumpulan Data Penelitian

Penelitian ini diawali dengan pengumpulan data yang akan digunakan sebagai acuan pembuatan *test case,* yaitu pengumpulan dokumen spesifikasi perangkat lunak. Spesifikasi yang diambil dan digunakan untuk penelitian ini hanya mencakup spesifikasi batas *input* untuk setiap *field* yang ada pada 7 modul yang akan diuji. Spesifikasi batasa *input* akan dijadikan acuan untuk membuat *test case* menggunakan BVA yang kemudian di optimasi untuk menghilangkran redudansi. Kemudian eksekusi *test case* secara otomatis akan menghasilkan *log* yang berisi informasi lengkap tentang jalannya pengujian. Data yang direkam mencakup *input* yang digunakan, serta hasil akhir (berhasil atau gagal).

3.2.2 Analisis Pengumpulan Data Hasil Penelitian

Untuk memberikan jaminan terkait optimasi BVA dapat meningkatkan efektivitas pengujian, dapat diukur dengan menggunakan formula pengukuran pengujian. Formula berikut digunakan untuk menghitung akurasi *test case*, akurasi

deteksi *bug*, dan peningkatan cakupan *test case*. Formula ini diadaptasi dari praktik umum yang dijelaskan dalam literatur pengujian perangkat lunak seperti yang dibahas oleh Kim dan Park (2020) dan Shukla dll., (2020):

1. **Akurasi *Test Case***

Akurasi test case dapat diukur dengan menggunakan formula 1, yaitu dengan menghitung persentase *test case* yang berhasil diidentifikasi (baik dalam menemukan *bug* atau tidak) dibandingkan dengan total *test case* yang dijalankan.

$$Akurasi\ Test\ Case\ = \frac{(Jumlah\ Test\ Case\ Berhasil)}{(Total\ Test\ Case\ Dijalankan)} \times 100\% \quad (1)$$

Keterangan :

Jumlah *Test Case* Berhasil: Jumlah *test case* yang menghasilkan hasil yang diharapkan (baik berhasil mengidentifikasi *bug* atau mengonfirmasi tidak adanya *bug*).

Total *Test Case* yang Dijalankan: Total seluruh *test case* yang diujicobakan dalam pengujian.

2. **Akurasi Deteksi *bug***

Akurasi deteksi *bug* dihitung menggunakan formula 2, yaitu dengan membandingkan jumlah *bug* yang berhasil dideteksi dengan metode BVA terhadap jumlah *bug* yang seharusnya ada.

$$Akurasi\ Deteksi\ Bug = \frac{(Jumlah\ Bug\ Teridentifikasi)}{(Total\ Bug\ yang\ Diharapkan)} \times 100\% \quad (2)$$

Keterangan :

Jumlah *Bug* yang Teridentifikasi: Jumlah *bug* yang berhasil ditemukan selama pengujian.

Total *Bug* yang Diharapkan: Jumlah *bug* yang seharusnya terdeteksi (berdasarkan *benchmark* dari *bug* yang hasil pengujian sebelumnya atau yang mungkin ada).

3. **Peningkatan Cakupan *Test Case***

Peningkatan cakupan *test case* dihitung menggunakan formula 3, yaitu dengan membandingkan jumlah *test case* setelah optimasi dengan jumlah *test case* sebelum optimasi.

$$Peningkatan\ Cakupan\ Test\ Case = \frac{(Jumlah\ Setelah\ Optimasi - Jumlah\ \ Sebelum\ Optimasi)}{(Jumlah\ Test\ Case\ Sebelum\ Optimasi)} \times 100\% \quad (3)$$

Keterangan :

Jumlah Setelah Optimasi: Merupakan jumlah *test case* yang berhasil dibuat setelah implementasi Optimasi BVA.

Jumlah Sebelum Optimasi: Merupakan jumlah *test case* yang berhasil dibuat tanpa implementasi Optimasi BVA.

3.2.3 Analisis Efektivitas Optimasi BVA oleh Ahli Media

Analisis ini dilakukan oleh ahli media terhadap *test case* yang berhasil dibuat dengan optimasi BVA, yang kemudian dilakukan pengukuran melalui kuesioner yang mengacu pada penelitian yang dilakukan Farooq dkk., (2017) dan Grano dkk., 2019 yang menyebutkan bahwa pentingnya pengukuran teknik pengujian berdasarkan aspek efektivitas, efisiensi dan kebermanfaatan. Penentuan skor diadaptasi dari skala *likert* berdasarkan rentang nilai yang dijelaskan pada tabel 3.1 sehingga akan diperoleh nilai akhir dari efektivitas *test case* yang telah dibuat untuk meningkatkan jumlah *test case* dan meningkatkan jumlah *bug* yang berhasil ditemukan selama pengujian. Kuesioner yang diajukan pada ahli media dapat dilihat pada Lampiran 1. Kuesioner Ahli Media.

Tabel 3.1

Skala *Likert*

Rentang Nilai	Kriteria Kelayakan
85 - 100	Sangat Layak
70-84	Cukup layak
50 – 69	Kurang layak
1 – 49	Tidak layak

3.3 Alat dan Bahan Penelitian

Pada penelitian ini, menggunakan alat dan bahan sebagai berikut:

3.3.1 Alat Penelitian

1. Perangkat Keras (*Hardware)*

Penelitian ini menggunakan beberapa alat perangkat keras

(*hardware)* untuk menunjang pelaksanaan penelitian, diantaranya seperti yang dijelaskan pada tabel 3.2 berikut:

Tabel 3.2

Kepustakaan Perangkat Keras (*hardware*) yang Digunakan

Kepustakaan / Alat	**Funsgi**
Processor Intel Core i3 BV302T	Sebagai otak dari komputer yang mengelola semua perintah dan operasi yang dilakukan oleh perangkat.
Installed RAM 12 GB	Sebagai penyimpanan sementara yang digunakan oleh sistem untuk menyimpan data yang sedang diolah
Mouse	Sebagai alat input untuk mengontrol kursor di layar.
Keyboard	Sebagai alat input utama untuk memasukkan data, seperti teks dan perintah, ke dalam komputer.

2. Perangkat Lunak (*Software)*

Selain perangkat keras (*hardware*) penelitian ini juga menggunakan beberapa perangkat lunak (*software)* untuk menunjang pelaksanaan penelitian, diantaranya seperti yang dijelaskan pada tabel 3.3 berikut:

Tabel 3.3

Kepustakaan Perangkat Lunak (*software*) yang Digunakan

Kepustakaan / Alat	**Funsgi**
Robot Framework	Kerangka kerja untuk *automation testing* yang mendukung pengujian berbasis *keyword-driven*.
Selenium Web Driver	*Library* untuk mengotomatisasi browser *web* dan berfungsi untuk membantu dalam eksekusi *test case* pada berbagai *browser*.
Google Chrome	Sebagai *browser* untuk menjalankan pengujian otomasi.
VS *Code*	Alat pengembangan untuk menulis dan mengedit kode yang digunakan untuk menyusun dan memodifikasi skrip pengujian
Qase.io	*Platform* untuk mengelola kasus uji, pada

	penelitian ini digunakan untuk mengelola dan mendokumentasikan kasus uji.
Python	Bahasa pemrograman yang digunakan untuk membuat skrip pengujian.

3.3.2 Bahan Penelitian

1. Aplikasi VConnect

VConnect merupalan aplikasi yang akan diuji dengan menggunakan *test case* otomasi. Pada penelitian ini, hanya menggunakan 7 modul pada VConnect sebagai bahan data uji seperti yang dijelaskan pada tabel 3.4 berikut:

Tabel 3.4

Modul Pengujian VConnect

Modul	Jumlah *Field Free Text*
Master Data Meeting Room	3
Master Data Vehicle	6
Master Data Driver Operational	4
Master Data Outsourcing	4
Master data Department	2
Master data employee	5
Master data Truck Driver	4

Ketujuh modul dengan jumlah *field* berupa *free text* pada VConnect dapat dilihat pada Lampiran 2. Modul Pengujian VConnect.

2. Nilai Batas *Input* (*Boundary Values*)

Kumpulan data *input* mencakup nilai di bawah batas bawa, batas bawa, batas atas dan nilai di atas batas atas. Nilai batas *input* ini akan digunakan sebagai data *input* pada *test case* untuk menguji batasan *input* dan memastikan fungsionalitas sistem telah sesuai dengan kebutuhan. Nilai batas *input* untuk modul yang diuji dapat dilihat pada Lampiran 3. Nilai Batas *Input meeting room* dapat dilihat pada tabel 3.5 berikut:

Tabel 3.5

Nilai Batas *Input*

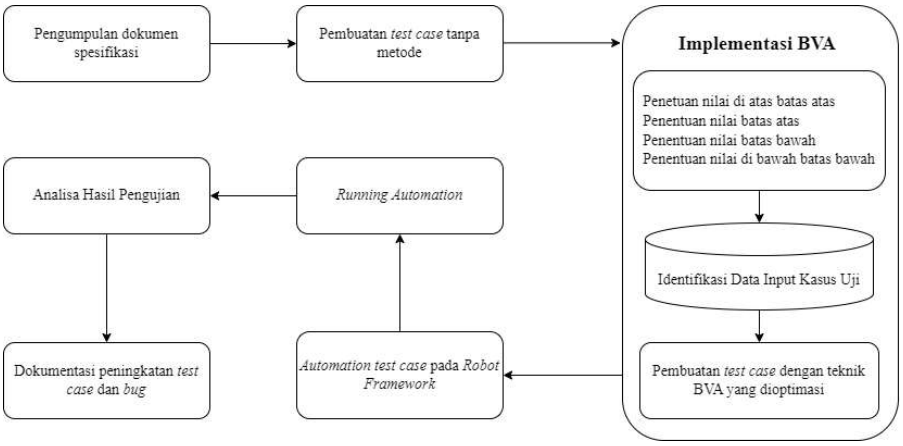
Modul	*Field*	Batas Input
Master Data Meeting Room	*Meeting Room Name*	Min 1 / Max 256
	Room Capacity	Min 1 / Max 100
	Room Facility	Min 1 / Max 256
Master Data Vehicle	*Vehicle Name*	Min 1 / 256
	Vehicle License Plate Number	Min 5 / Max 9
	Vehicle Chassis Number	Min Max 17
	Vehicle Engine Number	Min 6 / Max 20
	Vehicle Color	Min 1 / Max 256
	Vehicle Capacity (seat)	Min 1 / Max 9

3. Dokumentasi Kasus Uji

Dokumentasi kasus uji berupa dokumen yang mencakup deskripsi, langkah-langkah, data *input*, dan hasil yang diharapkan dari setiap *test case*. Dokumen ini menyediakan panduan untuk mengeksekusi *test case* selama proses pengujian, dokumentasi kasus uji yang dikelola melalui Qase.io dapat dilihat pada Lampiran 4. Pengelolaan *Test* Menggunakan Qase.io.

3.4 Prosedur Penelitian

Prosedur penelitian akan mengacu pada alur dari desain penelitian yang telah dijelaskan sebelumnya. Adapun gambaran detail terkait prosedur penelitian ini ditunjukkan pada gambar 3.2 sebagai berikut:



Gambar 3.2 Prosedur Penelitian

Gambar 3.2 menjelaskan prosedur penelitian yang dimulai dari pengumpulan dokumen spesifikasi perangkat lunak yang kemudian dijadikan sebagai acuan dalam pembuatan *test case* tanpa menggunakan metode BVA, yang berarti tidak memperhatikan batas pada setiap nilai *input*. Selanjutnya akan masuk ke tahap pengembangan *test case* menggunakan BVA yang dimulai dengan penentuan nilai atas bawah sesuai dengan algortima BVA. Setelah penentuan nilai selesai, maka akan dijadkan sebagai acuan untuk mengidentifikasi data *input* untuk setiap kasus uji yang berupa *input valid* dan *invalid* sesuai batas-batas yang telah diuraikan sebelumnya. Setelah data input berhasil diidentifikasi, maka dilakukan pembuatan *test case* yang kemudian dilakukan optimasi dengan cara mengelompokkan kasus uji agar tidak menghasilkan kasus uji yang redudansi.

Apabila proses pengembangan *test case* dengan BVA telah selesai, maka *test case* tersebut siap diotomasi dengan bantuan alat penunjang yaitu *Robot Framework.* Setiap *test case* akan diterjemahkan menggunakan bahasa pemrograman *Python* yang agar dapat dipahami oleh *Robot Framework.* Setelah *automation* dijalankan, maka *Robot Framework* secara langsung akan memberikan *report* dari pengujian yang telah dilakukan, yaitu berupa laporan jumlah *test case* yang berhasil dijalankan dan *test case* yang terdapat *bug* di dalamnya. Sehingga dapat dilakukan dokumentasi untuk menganalisis peningkatan cakupan *test case* dengan BVA dan peningkatan *bug* yang dihasilkan selama proses pengujian dengan mengimplementasikan optimasi BVA.