BAB III

OBJEK DAN METODE PENELITIAN

3.1. Objek Penelitian

Objek penelitian yang di bahas atau di teliti dalam tugas akhir ini berlokasi di Rental Mobil di CV. Tasya Lacaden pada bagian Kasir. Berikut adalah penjelasannya:

3.1.1 Sejarah Singkat Perusahaan

Rental Mobil di CV. Tasya Lacaden merupakan perusahaan jasa Trasportasi kendaraan dengan nama CV. Tasya Lacaden yang berdiri pada tahun 1999, Rental Mobil di CV. Tasya Lacaden mempunyai kantor pusat yang terletak di Jl.Cihampelas bawah Gg.Terasana No.7 Bandung.

CV. Tasya Lacaden bermula dari usaha peminjaman angkutan kota secara sederhana, kemudian CV.Tasya Lacaden mencoba usaha di bidang Rental Mobil Pada tahun 2007, Dengan segala keterbatasan CV.Tasya Lacaden melangkah sedikit demi sedikit untuk mendapatkan kemajuan dalam usahanya di bidang Rental Mobil.

Persaingan bisnis penyewaan kendaraan yang semakin hari semakin ketat tidak membuat kami ragu untuk melangkah. Didukung oleh tim yang handal dan profesional di bidangnya,

CV.Tasya Lacaden terbukti telah mendapat kepercayaan para pelanggan, sehingga seiring dengan waktu jumlah pelanggan dan armada kendaraan terus bertambah.

Ditunjang dengan banyaknya perusahaan yang menyerahkan pengelolaan kendaraan untuk kegiatan operasional dan pendistribusian kepada pihak ketiga membuat usaha penyewaan kendaraan harus dapat lebih diandalkan dalam segala hal, seperti dalam penyediaan kendaraan, service yang diberikan sampai dengan driver yang bertanggung jawab.

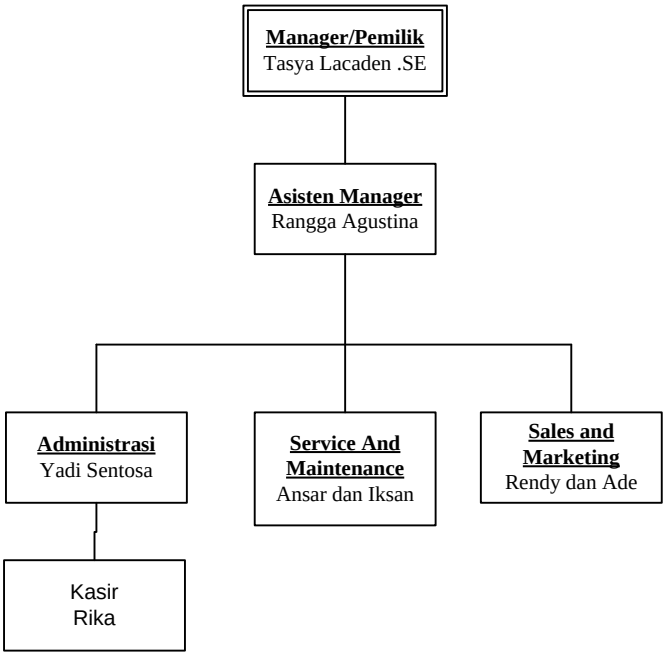
3.1.2 Visi dan Misi Perusahaan

"MITRA BISNIS TERPERCAYA' selalu diterapkan setiap saat dalam melakukan transaksi, karena hal itu merupakan visi utama kami.

Misi Rental Mobil Tasya Lacaden "KEPUASAN PELANGGAN MERUPAKAN PRIORITAS KAMI" apapun konsekuensinya, terbukti telah memberikan dorongan pada perbaikan dan peningkatan mutu fasilitas serta pelayanan demi mendukung mobilitas anda, kapanpun dan dimanapun anda berada.

Komitmen ini tercermin didalam struktur organisasi dimana setiap personil mempunyai visi dan misi yang sama untuk selalu memberikan pelayanan yang terbaik.

3.1.3 Struktur Organisasi Perusahaan



Gambar 3.1. Struktur Organisasi CV.Tasya Lacaden

3.1.4. Deskripsi Tugas

Seperti halnya sebuah perusahaan, CV.Tasya Lacaden terdapat bagian-bagian yang mempunyai tugas dan wewenang serta tanggung jawab dalam menyelesaikan semua pekerjaannya, antara lain :

1. Manager / Pemilik

Manager / Pemilik, bertugas sebagai pemilik CV.Tasya Lacaden yang memiliki tugas terpenting sebagai pemegang penuh kendali perusahaan dan

dalam pengambilan keputusan harus disetujui oleh manager / pemilik perusahaan.

2. Asisten Manager

Asisten Manager, bertugas sebagai tangan kanan manager dimana membantu manager, mengawasi karyawan, serta menggantikan tugas manager apabila manager sedang berhalangan atau keluar kota .

3. Administrasi

a. Memberikan Informasi atau Tata Cara didalam penyewaan mobil.
b. Menentukan Tarif atau Harga Sewa dan Denda serta peraturanatau ketentuan yang berlaku didalam tata cara penyewaan mobil selain itu dia mempunyai tanggungg jawab terhadap tugasnya dan kepada Pimpinan.

4. Service And Maintenace

Service And Maintenace, bertugas melakukan pemeliharaan rutin seperti cek oli, cek mesin mobil, dan perbaikan – perbaikan bila ada kerusakan - kerusakan.

5. Sales And Marketing

Sales And Marketing, bertugas sebagai pemasaran produk – produk yang ada pada CV.Tasya Lacaden, meningkatkan omset perusahaan serta menangani keluhan – keluhan konsumen tentang pelayanan konsumen.

6. Kasir

a. Menerima pembayaran sewa mobil sesuai dengan merk atau type mobil dari pelanggan, jika telah sesuai maka dapat diproses cara penyewaan oleh Kasir.

b. Menerima pembayaran denda sewa.

3.2. Metode Penelitian

3.2.1 Desain Penelitian

Desain penelitian ini termasuk kedalam penelitian yang bersifat deskriptif. Penelitian deskriptif adalah penelitian yang bertujuan untuk memperoleh ciri-ciri variabel, dimana dalam penelitian ini adalah untuk memperoleh gambaran tentang kinerja program yang dirancang dan diimplementasikan kepada pengguna (user) dalam perusahaan.

Penelitian ini dengan mengambil kasus di Rental Mobil di CV. Tasya Lacaden unit pengamatan pada bagian Kasir.

3.2.2 Jenis dan Metode Pengumplan Data

Metode pengumpulan data adalah cara-cara yang dapat digunakan peneliti untuk mengumpulkan data. Pengumpulan data disini bertujuan untuk mendapatkan informasi tentang proses rental mobil di CV. Tasya Lacaden . Berikut beberapa metode yang digunakan peneliti :

3.2.2.1. Sumber Data Primer

Data primer merupakan Data yang diperoleh langsung dari unit pengamatan / responden peneliti. Teknik / metode pengumpulan data primer meliputi:

a. Observasi

Pengumpulan data melalui pengamatan & pencatatan terhadap obyek penelitian pada bagian Kasir di Rentalm Mobil CV. Tasya Lacaden.

b. Wawancara / Interview

Pengumpulan data melalui tatap muka dan tanya jawab langsung dengan responden. Peneliti melakukan tanya jawab dengan Rika yang bekerja di Rental Mobil CV. Tasya Lacaden pada bagian Kasir

3.2.2.2. Sumber Data Sekunder

Data sekunder adalah data yang digunakan untuk mendukung data primer, merupakan jenis data yang sudah diolah terlebih dahulu oleh pihak pertama atau peneliti. Teknik / metode pengumpulan data sekunder meliputi sistem dan teknik perancangan basis data dari sumber-sumber yang berhubungan dengan sistem yang akan dikembangkan, seperti pengumpulan bukti pembayaran sewa mobil dan pengumpulan data pelanggan.

3.2.3.Metode Pendekatan dan Pengembangan Sistem

Dalam metode pendekatan dan pengembangan sistem akan dijelaskan mengenai metode pendekatan sistem, metode pengembangan sistem dan alat bantu analisis pengembangan sistem.

3.2.3.1.Metode Pendekatan Sistem

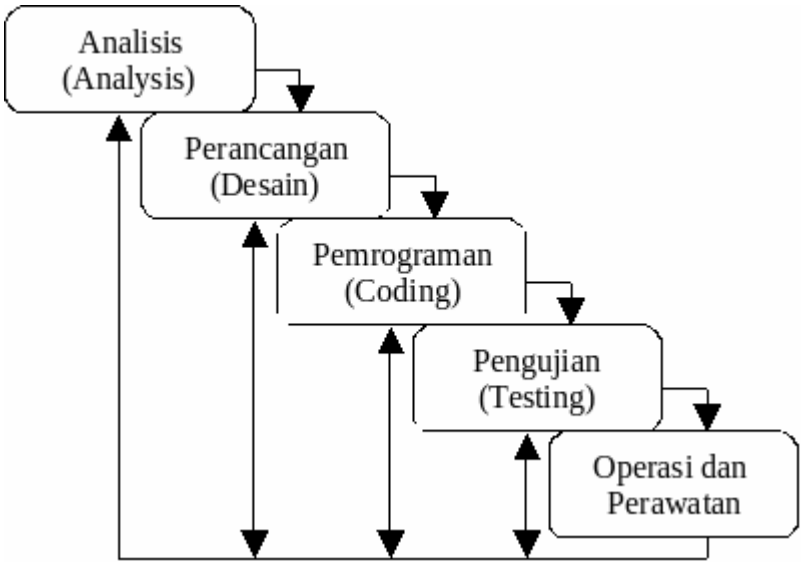
Dalam penelitian ini metode yang digunakan yaitu menggunakan pendekatan berorientasi objek karena dengan pendekatan berorientasi objek ini efektif karena objek-objek dapat mempresentasikan bagian-bagian dari dunia eksternal, mempersempit kesenjangan konseptual antara dunia eksternal dan komponen-komponen perangkat lunak

3.2.3.2. Metode Pengembangan Sistem

Metode Pengembangan adalah sebuah cara yang tersistem atau teratur yang bertujuan untuk melakukan analisa pengembangan suatu sistem agar sistem tersebut dapat memenuhi kebutuhan**.**

Dalam melakukan pengembangan sitem menggunakan metode waterfall karena model ini mengusulkan sebuah pendekatan perkembangan software yang sistematika dan yang mulai pada tingkat dan kemajuan sistem pda seluruh analisis, desain, kode pengujian, dan pemeliharaan. Secara garis besar metode *waterfall* mempunyai langkah-langkah sebagai berikut : Analisa, Design, Code dan Testing, Penerapan dan Pemeliharaan.

Secara garis besar metode waterfall mempunyai langkah-langkah sebagai berikut.



Gambar 3.2 Model Waterfall

1. Analisa

Langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa malakukan sebuah penelitian, wawancara atau study literatur. Seorang sistem analis akan menggali informasi sebanyak-banyaknya dari *user* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan dokumen *user requirment* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen ini lah yang akan menjadi acuan sistem analis untuk menterjemahkan ke dalam bahasa pemprogram.

2. Design

Proses desain akan menerjemahkan syarat kebutuhan ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat coding. Proses ini berfokus pada : struktur data, arsitektur perangkat lunak, representasi interface, dan detail (algoritma) prosedural. Tahapan ini akan menghasilkan dokumen yang disebut *software requirment*. Dokumen inilah yang akan digunakan *proggrammer* untuk melakukan aktivitas pembuatan sistemnya.

3. Coding & Testing

Coding merupakan penerjemahan *design* dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh *programmer* yang akan meterjemahkan transaksi yang diminta oleh *user*. Tahapan ini lah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem.

Dalam artian penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap sistem tersebut dan kemudian bisa diperbaiki.

4. Penerapan

Tahapan ini bisa dikatakan *final* dalam pembuatan sebuah sistem. Setelah melakukan analisa, design dan pengkodean maka sistem yang sudah jadi akan digunakan oleh *user*.

5. Pemeliharaan

Perangkat lunak yang sudah disampaikan kepada pelanggan pasti akan mengalami perubahan.Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (periperal atau sistem operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional.

Keuntungan Metode Waterfall

a. Kualitas dari sistem yang dihasilkan akan baik. Ini dikarenakan oleh pelaksanaannya secara bertahap. Sehingga tidak terfokus pada tahapan tertentu.

b. Document pengembangan sistem sangat terorganisir, karena setiap fase harus terselesaikan dengan lengkap sebelum melangkah ke fase berikutnya. Jadi setiap fase atau tahapan akan mempunyai dokumen tertentu.

Kelemahan waterfall

a. Diperlukan majemen yang baik, karena proses pengembangan tidak dapat dilakukan secara berulang sebelum terjadinya suatu produk.

b. Kesalahan kecil akan menjadi masalah besar jika tidak diketahui sejak awal pengembangan.

c. Pelanggan sulit menyatakan kebutuhan secara eksplisit sehingga tidak dapat mengakomodasi ketidakpastian pada saat awal pengembangan.

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas.

Pengertian "berorientasi objek" berarti bahwa kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya.

Metode pengembangan system yang digunakan dalam penelitian ini yaitu dengan pengembangan berorientasi objek dimana :

a. Abstraksi yang lebih tinggi Karena pengembangan dapat berlangsung di level objek dan mengabaikan rincian-rincian (yaitu data dan fungsi perlu ada) system sepanjang masih perlu. Dengan demikian menjadikan proses perencangan, pengkodean, pengujian dan pemeliharaan system jauh lebih sederhana.
b. Transisi yang seamlessness di antara tahap-tahap di proses pengembangan perangkat lunak. Pada hamper seluruh tahap proses pengembangan berorientasi objek maka konsep, bentukan yang sama yang digunakan.
c. Dapat memanfaatkan keampuhan bahasa berorientasi objek dan mempraktekan teknik pemrograman yang baik.

d. Objek dapat digunaulang karena memodelkan domain persoalan di dunia eksternal secara langsung setidaknya sampai model analisi. Masing-masing objek dapat berdiri sendiri atau di dalam lingkaran sempit kelas-kelas kolaboratornya. Dengan kemampuan lebih mandiri, maka tingkat gunaulang system berorientasi objek menjadi lebih tinggi di banding pendekatan konvensional.

2. Metodologi pengembangan sistem berorientasi objek mempunyai tiga karakteristik utama :

   A. *Encapsulation*

      1. Encapsulation merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses.
      2. Data dan prosedur atau fungsi dikemas bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya.
      3. Data terlindung dari prosedur atau objek lain, kecuali prosedur yang berada dalam objek itu sendiri.

   B. *Inheritance*

      1. Inheritance adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metode dari induknya langsung. Atribut dan metode dari objek dari objek induk diturunkan kepada anak objek, demikian seterusnya.

2. Inheritance mempunyai arti bahwa atribut dan operasi yang dimiliki bersama di anatara kelas yang mempunyai hubungan secara hirarki.
3. Suatu kelas dapat ditentukan secara umum, kemudian ditentukan spesifik menjadi subkelas. Setiap subkelas mempunyai hubungan atau mewarisi semua sifat yang dimiliki oleh kelas induknya, dan ditambah dengan sifat unik yang dimilikinya.
4. Kelas Objek dapat didefinisikan atribut dan service dari kelas Objek lainnya.
5. Inheritance menggambarkan generalisasi sebuah kelas

C. *Polymorphism*

1. Polimorfisme yaitu konsep yang menyatakan bahwa seuatu yang sama dapat mempunyai bentuk dan perilaku berbeda.
2. Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.
3. Kemampuan objek-objek yang berbeda untuk melakukan metode yang pantas dalam merespon message yang sama.
4. Seleksi dari metode yang sesuai bergantung pada kelas yang seharusnya menciptakan Objek.

3.2.3.3. Alat Bantu Analisis dan Perancangan

Pada sub bab ini akan dijelaskan mengenai alat bantu apa saja yang dipakai dalam analisis dan perancangan Sistem Informasi yang akan dibuat yaitu dengan

menggunakan *Unified Modelling Language* (UML) yang terdiri dari Use Case Diagram, Class Diagram, Statechart Diagram, Activity Diagram, Sequence Diagram, Collaboration diagram, *Component diagram, Deployment diagram.*

1. Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. *Use case* diagram menekankan kepada "apa" yang dapat dilakukan oleh sistem, dan bukan "bagaimana" hal itu dapat dilakukan. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, mengelola daftar teman, mengirimkan pesan dan sebagainya. Penggunaan *use case* diagram adalah pada saat dibutuhkan penggambaran mengenai requirement sistem perangkat lunak yang akan dibangun dengan tujuan dapat dikomunikasikan kepada klien yang akan menggunakan perangkat lunak tersebut sehingga didapatkan secara garis besar kebutuhan - kebutuhan perangkat lunak nantinya.

2. Class Diagram

Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungannya satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (*Class Name*)

2. Atributes

3. Metoda (*Operations*)

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.

b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.

c. *Public*, dapat dipanggil oleh *class* lainnya.

Berikut adalah bentuk dari *class* diagram :

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung instansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan

3. Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang

diterima. Pada umumnya *statechart* diagram menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart* diagram).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah

4. Activity Diagram

Activity diagram memodelkan alur kerja (*workflow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Diagram ini sangat mirip dengan sebuah *flowchart* karena kita dapat memodelkan sebuah alur kerja dari suatu aktifitas ke aktifitas lainnya atau dari suatu aktifitas ke dalam keadaan sesaat (*state*). Seringkali bermanfaat jika kita membuat *activity* diagram terlebih dahulu dalam memodelkan sebuah proses untuk membantu kita memahami proses secara keseluruhan.

Activity diagram juga sangat berguna ketika kita ingin menggambarkan perilaku paralel atau menjelaskan bagaimana perilaku dalam berbagai *use case* berinteraksi. Kita dapat menggunakan *statechart* diagram untuk memodelkan perilaku dinamis satu kelas dari objek. *Statechart* diagram memperlihatkan urutan keadaan sesaat (*state*) yang dilalui sebuah objek. Kejadian yang menyebabkan transisi dari

suatu state atau aktivitas kepada yang lainnya. Statechart diagram khususnya digunakan untuk memodelkan tarap-tarap diskrit dari sebuah siklus hidup objek, sedangkan *activity* diagram paling cocok digunakan untuk memodelkan urutan aktivitas dalam suatu proses.

5. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. *Sequence* diagram terdiri antar dimensi *vertikal* (waktu) dan dimensi *horizontal* (objek-objek yang terkait). *Sequence* diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai *response* dari sebuah *event* untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline vertikal*. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activition bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

6. Collaboration diagram

Collaboration diagram juga menggambarkan interaksi antar obyek seperti *sequence* diagram, tetapi lebih menekankan pada peran masing-masing objek dan

bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence* number, dimana *message* dari level tertinggi memiliki nomor 1.

7. Component diagram

Component diagram menggambarkan struktur dan hubungan antar komponen perangkat lunak, termasuk ketergantungan (*dependency*) diantaranya. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

8. Deployment diagram

Deployment/*physical* diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, server atau perangkat keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.

3.2.4. Pengujian Software

Software testing adalah investigasi yang dilakukan kepada software untuk mengetahui kualitas suatu produk. dilakukan untuk mengeliminasi defect yang terjadi setelah software diluncurkan, pengujian software sendiri tidak akan pernah secara penuh mengenali defect pada suatu software.black box testing dan white box testing adalah metode yang dikenal untuk menguji software.

Black box testing mengambil perspektif dari luar untuk melakukan testing, tes ini bisa berupa fungsional test atau non-fungsional, dilakukan dengan cara memasukkan input dengan benar maupun salah, kemudian menguji apakah outputnya sesuai dengan spesifikasi atau tidak, hal ini dilakukan tanpa harus tahu apa yang terjadi didalam mengolah input untuk mengasilkan output. black box testing popular karena kesederhanaan metodenya.

White box testing, atau biasa dikenal juga dengan nama glass box testing menguji software dari dalam, dengan melihat struktur internal software. untuk itu diperlukan kemampuan programing untuk mengidentifikasi bagian dalam suatu software.

Metode pengujian adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan

Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian black box merupakan metode peracangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat

lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan.

Pengujian black box merupakan pendekatan komplementer dari teknik white box, karena pengujian *black box* diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik white box. Pengujian *black box* berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.

Pengujian *black box* berusaha menemukan kesalahan dalam kategori:

(1) fungsi-fungsi yang tidak benar atau hilang

(2) kesalahan interface

(3) kesalahan dalam struktur data atau akses database eksternal

(4) kesalahan kinerja,

(5) inisialisasi dan kesalahan terminasi.