BAB III

OBJEK DAN METODE PENELITIAN

3.1. OBJEK PENELITIAN

Penulis melalakukan penelitian di CV. CITRA TIARA TRANS atau yang dikenal sebagai CITI TRANS yang berlokasi di Jl. Dipati Ukur 53 Bandung 40132.

3.1.1. Sejarah Perusahaan

Berawal dari ingin mengakomodir kawan sejawat untuk pulang di masa liburan sekelompok mahasiswa memulai suatu bisnis jasa angkutan yang semakin lama semakin menjanjikan, hingga pada akhirnya sebuah perusahaan jasa travel yang di beri nama CV. CITRA TIARA TRANS resmi berdiri pada tahun 2005 pada bulan Agustus dimana lokasi awal perusahaan ini di Jakarta dan pada tahun 2006 berpindah ke Bandung tepatnya di Jl. Dipati Ukur No. 53, lokasi yang sangat strategis dimana lokasi tersebut merupakan wilayah perkampusan dan termasuk wilayah yang sangat strategis diwilayah bandung dalam menjalankan bisnis jasa travel.

CV. CITRA TIARA TRANS atau yang lebih dikenal dengan nama CITI TRANS mempunyai karakter perusahaan jasa yang memberikan pelayanan yang berbeda yaitu pelayanan yang eksklusif dalam artian kelas menengah keatas dan mempunyai beberapa kelebihan dibandingkan perusahaan jasa travel yang lainnya,

beberapa kelebihan yang dimiliki CITI TRANS adalah merupakan pelopor travel dengan susunan tempat duduk single sheet, Unit GPS pada setiap kendaraan agar setiap kendaraan dapat terpantau segala informasinya dengan jelas serta mempunyai komitmen dalam hal ganti rugi apabila terdapat keluhan.

Setelah dibukanya jalan tersebut maka akses antara kota Bandung dan Jakarta menjadi sangatlah terbuka, hal tersebut juga sangat mendukung bergairahnya berbagia bisnis terutama di kota bandung yang merupakan tujuan perjalanan utama jakarta. Hal ini juga dilihat makin banyak dan berkembangnya pusat-pusat perbelanjaan, *took* souvenir dan lain-lain. Keadaan demikian dimanfaatkan juga oleh para pelaku bisnis transportasi yang ada, dengan melihat terbukanya akses jalan tol tersebut maka diapandang jasa transportasi merupakan salah atu bisnis potensial yang dapat dikembangkan

CV CITRA TIARA TRANSPORT atau lebih dikenal CITI TRANS merupakan salah satu travel yang cukup besar di kota Bandung. CITI TRANS menawarkan berbagai macam jasa selain jasa travel, yaitu jasa pengiriman barang, dokumen atau lebih dikenal dengan sebutan pengiriman paket.

CITI TRANS perusahaan jasa transportasi Bandung-Jakarta yang dipimpin oleh Andrew Arristanto diresmikan pada tanggal 11 September 2005 dan mulai beroperasi 7 Oktober 2005 dengan kantor pusat di Bandung yakni Jln. Imam Bonjol no.18, pada awal beroperasinya Citi Trans melayani jalur Bandung-Semanggi PP setiap 2 jam sekali.

Pada tahun 2006 Citi Trans memindahkan kantor pusatnya ke Jln. Dipatiukur no. 53 dan menambah jalur baru ke Fatmawati, Bintaro yang kemudian pada tahun berikutnya di susul oleh jalur yang terbaru yakni Kelapa Gading. Dan hingga sekarang Citi Trans melayani jalur PP setiap 1 jam sekali.

Untuk mempertahankan posisi dan memenangkan persaingan, setiap perusahaan mempunyai target tertentu yang berupa visi dan misi. Pencapaian visi dan misi tersebut perlu direalisasikan melalui tindakan efektif dan efisien yang mempunyai arah pasti menuju sasaran yang ingin dicapai.

3.1.2. Visi dan Misi

CITI TRANS merupakan perusahaan yang bergerak di bidang jasa transportasi khususnya travel Bandung-Jakarta PP. CITI TRANS memiliki visi, misi serta motto perusahaan yang menjadi dasar berjalannya perusahaan ini, visi, misi dan motto CITI TRANS yaitu :

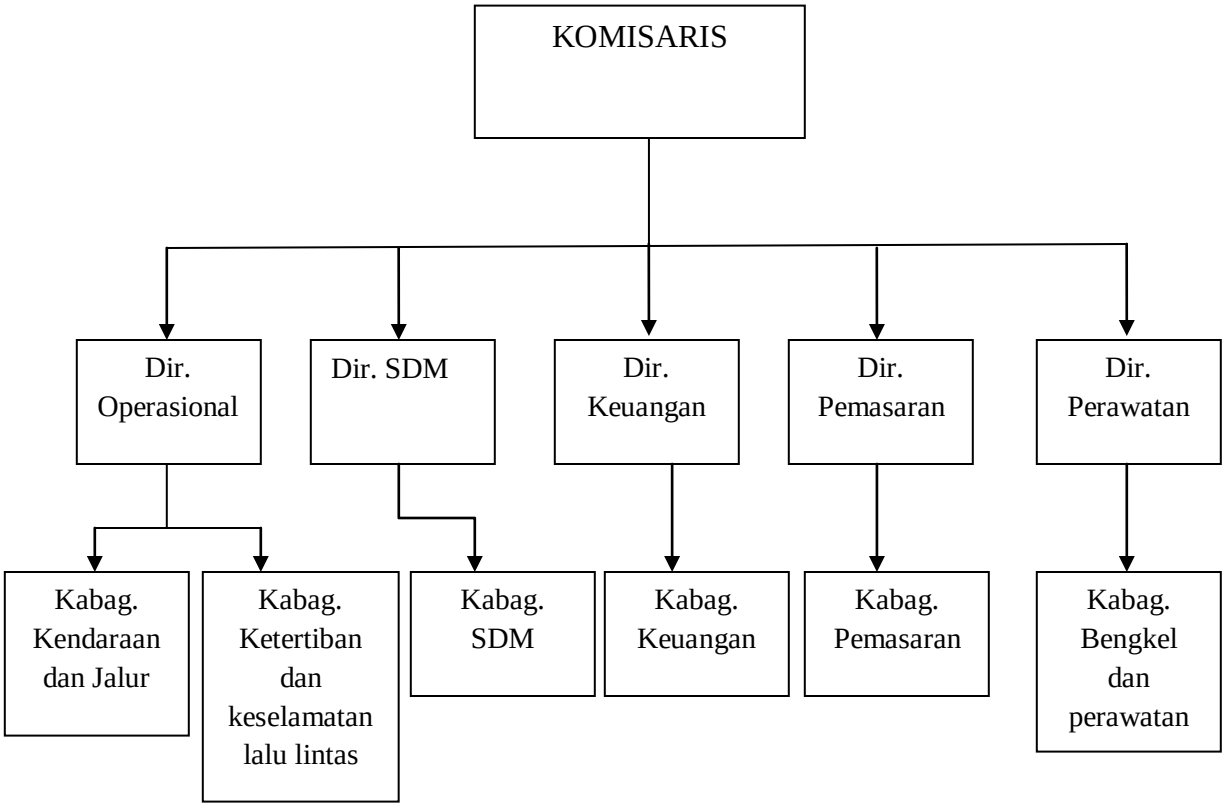
a. VISI

Berperan serta dalam dunia transportasi sebagai penyedia jasa transportasi yang berkualitas yang memberikan segala kemudahan serta kenyamanan dalam perjalanan dan terus menerus melayani permintaan konsumen.

b. MISI

Merubah, menambah kendaraan dan fasilitas yang telah ada untuk meningkatkan jumlah pelanggan.

3.1.3. Struktur Organisasi



Gambar 3.1 Struktur Organisasi

(Sumber: Citi Trans Travel)

3.1.4. Deskripsi Kerja

1. Komisaris

Merupakan pengawas sekaligus pemilik dari CV. CITI TIARA TRANS yang terdiri dari beberapa orang yag bertugas sebagai pengawas serta pengambil kebijakan dalam perusahaan ini serta pemberi nasehat kepada tiap-tiap direktur.

2. Direktur Operasional

Bertugas membawahi dua Kepala Bagian kendaraan dan jalur serta ketertiban dan keselamatan lalulintas. Bertugas mengambil kebijakan dalam hal pengusulan penambahan kendaraan serta penambahan jalur dan mengawasi ketertiban serta keselamatan lalulintas yang ditekankan kepada para Driver.

3. Direktur Sumber Daya Manusia (SDM)

Membawahi Kepala Bagian SDM, bertugas untuk merekrut pegawai baru serta melakukan training kepada karyawan secara rutin dan terprogram, juga bertugas sebagai bagian yang sangat peka terhadap masukan dari para pelanggan yang nantinya akan di beritahukan dan diarahkan kepada para karyawan (rapat rutin) sehingga terjadinya hubungan relasi yang baik antara pelanggan dengan perusahaan.

4. Direktur Keuangan

Membawahi Kepala Bagian Keuangan, bertugas mengatur masuk keluarnya uang pada perusahaan serta memberikan segala laporan kepada Komisaris tentang perkembangan keuangan perusahaan.

5. Direktur Pemasaran

Membawahi Kepala Bagian Pemasaran, bertugas melakukan promosi kepada para pelanggan, melakukan hal-hal kreatif dan inovatif dalam menarik perhatian pelnggan serta menjalin hubungan kerjasama antara perusahaan dalam program diskon.

6. Direktur Perawatan

Membawahi Kepala Bengkel dan Perawatan bertugas untuk melakukan perawatan kendaraan secara rutin, memeriksa segala kelengkapan kendaraan serta mengecek kelayakan kendaraan sebelum keberangkatan.

7. Kepala Bagian Kendaraan dan Jalur

Membantu Direktur Operetional dalam bidang kendaraan dan jalur.

8. Kepala Bagian Ketertiban dan Keselamatan Lalulintas

Membantu Direktur Operetional dalam bidang ketertiban dan keselamatan lalulintas.

9. Kepala Bagian Keuangan

Membantu Direktur Keuangan dalam melaksanakan tugasnya.

10. Kepala Bagian Keuangan

Membantu Direktur Keuangan dalam melaksanakan tugasnya.

11. Kepala Bagian Marketing

Membantu Direktur Marketing dalam melaksanakan tugasnya.

12. Kepala Bengkel Perawatan

Membantu Direktur Perawatan dalam melaksanakan tugasnya.

3.2. Metode Penelitian

Metode merupakan cara bagaimana secara teknis membangun perangkat lunak yang harus berada pada komitmen dasar menuju kualitas. Suatu penelitian tidak akan berjalan dengan baik apabila tidak dilakukan dalam suatu proses yang teratur

dan terarah. Oleh karena itu, diperlukan suatu metode pendekatan/penyelesaian untuk melaksanakan suatu penelitian.

3.2.1. Desain Penelitian

Desain penelitian adalah keseluruhan dari perencanaan untuk menjawab pertanyaan penelitian dan mengantisipasi beberapa kesulitan yang mungkin timbul selama proses penelitian, hal ini penting karena desain penelitian merupakan strategi untuk mendapatkan data yang dibutuhkan untuk keperluan pengujian hipotesis atau untuk menjawab pertanyaan penelitian dan sebagai alat untuk mengontrol variabel yang berpengaruh dalam penelitian. Desain penelitian yang baik dapat memudahkan kita dalam melakukan penelitian dan sesuai dengan tujuan yang telah ditetapkan sebelumnya.

Sesuai dengan tujuan yang ingin dicapai dalam penelitian ini, penulis mengumpulkan data sebagai bahan laporan dengan menggunakan metode deskriptif yaitu suatu penelitian yang dilakukan dengan tujuan utama untuk memberikan gambaran atau Deskripsi tentang suatu keadaan secara objektif sehingga dapat diperoleh suatu kesimpulan.

3.2.2. Jenis dan Metode Pengumpulan Data

Untuk memperoleh data dan keterangan yang diperlukan, dalam hal ini penulis menggunakan cara sebagai berikut :

3.2.2.1. Sumber Data Primer

Jenis pengumpulan data primer merupakan penelitian yang mengumpulkan data langsung dari lapangan penelitian atau tempat penelitian untuk mengetahui

keadaan penelitian yang akan dijalankan. Metode yang dipakai dibagi ada beberapa cara sebagai berikut :

a. Wawancara

Wawancara, yaitu teknik pengumpulan data dengan cara tanya jawab secara langsung dengan para pegawai yang berhubungan dengan penelitian ini. Wawancara dilakukan dengan pimpinan yang dapat memberikan data yang relevan dengan masalah yang diteliti oleh penulis.

b. Observasi

Observasi merupakan proses untuk mendapatkan data dengan mengadakan pengamatan ditempat lokasi penelitian secara langsung yaitu pada Travel CITI TRANS Bandung, kemudian hasil dari pengamatan tersebut dicatat dan dianalisis lebih lebih lanjut.

3.2.1.3. Sumber Data Sekunder

Data sekunder merupakan data yang di peroleh secara tidak langsung yaitu berupa referensi dari buku, makalah atau dari berupa sumber literature internet dan artikel - artikel yang berhubungan dengan tema penelitian yang sedang dilakukan dan dapat dikaji sebagai bahan rujukan dan landasan teoritis dalam pemecahan masalah.

3.2.3. Metode Pendekatan dan Pengembangan Sistem

Metode pendekatan dan pengembangan sistem adalah beberapa metode yang digunakan pada penelitian ini, diantaranya metode pengembangan sistem dan metode pendekatan sistem dan alat bantu analisis dan perancangan, Dibawah ini akan diuraikan tentang pemahaman dari metode tersebut.

3.2.3.1. Metode Pendekatan Sistem

Metode pendekatan yang digunakan oleh penulis adalah dengan menggunakan pendekatan sistem berorientasi objek. Menurut Bambang Hariyanto (2004 : 86) keunggulan pendekatan berorientasi objek adalah sebagai berikut :

1. Bekerja yang mendekati kognisi manusia.

2. Menghasilkan sistem yang dibangun di atas bentuk-bentuk antara yang stabil dan dengan demikian lebih mampu untuk mengikuti perubahan.

3. Dapat digunakan tidak hanya pada perancangan perangkat lunak tapi juga seluruh proses pengembangan perangkat lunak.

4. Mereduksi resiko pengembangan sistem-sistem kompleks terutama karena pemaduan terjadi menyebar pada siklus kehidupan tidak terjadi dalam sekejap sebagaimana kejadian ledakan besar (*big bang*).

5. Pendekatan ini membantu mengeksploitasi keampuhan bahasa pemrograman berbasis objek.

Pendekatan berorientasi objek adalah salah satu faktor untuk menuntun menuju ke solusi perangkat lunak, untuk itu kadang harus menyeimbangkan dengan pertimbangan- pertimbangan pendekatan yang lain. Menurut Booch, prinsip-prinsip yang mendasari pengembangan berorientasi objek adalah :

1. Abstraksi
2. Pengkapsulan
3. Modularitas
4. Hirarki
5. *Typing*
6. Kongkurensi
7. *Persistence*

Prinsip- prinsip ini bekerja/diterapkan secara sinergi (bersama-sama bekerja sama). Prinsip-prinsip tersebut merupakan prinsip-prinsip utama, dimana keempat prinsip pertama adalah wajib sementara tiga prinsip berikutnya adalah *optional* (boleh digunakan ataupun tidak).

3.2.2.2. Metode Pengembangan Sistem

Metode pengembangan sistem menggunakan metode *prototype* yang merupakan metode dalam pengembangan sistem yang menggunakan pendekatan untuk membuat suatu program dengan cepat dan bertahap sehingga segera dapat dievaluasi oleh pemakai. Kadang-kadang pemakai atau user hanya memberikan beberapa kebutuhan umum *software* tanpa detail input, proses atau *detail output*.

Di lain waktu mungkin dimana tim pembangun (*developer*) tidak yakin terhadap efisiensi dari algoritma yang digunakan, tingkat adaptasi terhadap sistem operasi atau rancangan *form user interface*. Ketika situasi seperti ini terjadi model *prototype* sangat membantu proses pembangunan *software.*

Seperti pada semua metode, *prototype* dimulai dari pengumpulan kebutuhan. Dengan perencanaan yang cepat akan dibentuk konstruksi dari prototypenya. *Prototype* ini dievaluasi oleh pelanggan dan digunakan untuk mengelola kembali kebutuhan dari perangkat lunak yang dikembangkan. Suatu proses iterasi terjadi, setelah *prototype* disesuaikan dengan kebutuhan pelanggan, sementara pihak pengembang makin mengerti keinginan pemakai.

Tahapan proses model *prototype*, bisa dijelaskan sebagai berikut:

1. Pengumpulan kebutuhan : *developer* dan pelanggan bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya. Detail kebutuhan mungkin tidak dibicarakan disini, pada awal pengumpulan kebutuhan.
2. Perancangan : perancangan dilakukan cepat dan rancangan mewakili semua aspek *software* yang diketahui, dan rancangan ini menjadi dasar pembuatan *prototype.*
3. Evaluasi *prototype* : pelanggan mengevaluasi prototype yang dibuat dan digunakan untuk memperjelas kebutuhan *software*.

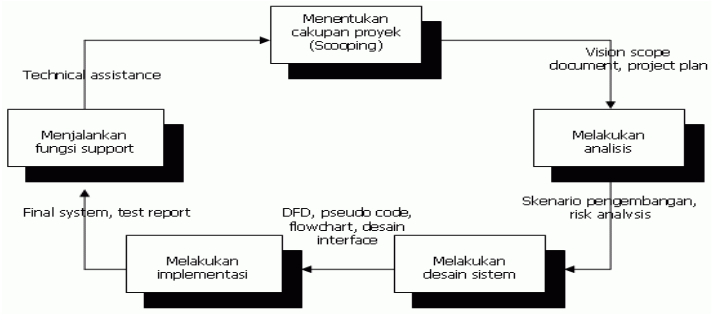
Perulangan ketiga proses ini terus berlangsung hingga semua kebutuhan terpenuhi. *Prototype-prototype* dibuat untuk memuaskan kebutuhan pelanggan dan untuk memahami kebutuhan pelanggan lebih baik.

Seluruh metode pengembangan sistem memiliki kelebihan dan kekurangan. Berikut adalah kelebihan dan kekurangan dari metode *prototype*. Abdul Kadir (2003 : 418) :

1. Kelebihan
   a. Pendefinisian kebutuhan pemakai menjadi lebih baik karena keterlibatan pemakai yang lebih intensif.
   b. Meningkatkan kepuasan pemakai dan mengurangi resiko pemakai tidak menggunakan sistem mengingat keterlibatan mereka yang sangat tinggi sehingga sistem memenuhi kebutuhan mereka dengan lebih baik.
   c. Memperkecil kesalahan disebabkan pada setiap versi *prototype* kesalahan segera terdeteksi oleh pemakai.
   d. Pemakai memiliki kesempatan yang lebih banyak dalam meminta perubahan-perubahan.
   e. Mempersingkat waktu dalam mengembangkan sistem secara keseluruhan.
   f. Menghemat biaya (menurut penelitian, biaya penghematan dapat mencapai 10% hingga 20% dibandingkan dengan menggunakan metode SDLC tradisional).
2. Kekurangan
   a. Sistem akan baik jika pemakai sungguh-sungguh meluangkan waktunya untuk menggarap *prototype*.
   b. Dokumentasi sering terabaikan karena pengembang lebih berkonsentrasi pada tahap pengujian dan pembuatan *prototype.*
   c. Waktu yang singkat dapat menghasilkan sistem yang tidak lengkap dan kurang teruji.

d. Jika terlau banyak proses pengulangan dalam membuat *prototype*, ada kemungkinan pemakai menjadi jenuh dan memberikan respon yang negatif.

e. Apabila *prototype* tidak dikelola dengan baik dapat mengakibatkan *prototype* tidak pernah berakhir karena usulan perubahan terlalu sering dan mudah untuk dipenuhi.



Gambar 3.2 Proses *Prototype*

(Sumber: http://www.ristinet.com/2010/04/16)

3.2.3.3. Alat Bantu Analisis & Perancangan

Menurut Bambang Hariyanto (2004 : 252) Analisis Sistem menggunakan pemodelan agar :

1. Fokus pada fitur-fitur sistem penting sambil meremehkan fitur-fitur kurang penting
2. Membahas perubahan-perubahan dan koreksi- koreksi kebutuhan- kebutuhan pembeli dengan ongkos yang murah dan resiko minimal.
3. Memferivikasi pemahaman analis sistem terhadap lingkungan pemakai dan didoekumnetasikan dengan cara yang diperlukan oleh perancangan dan pemograman membangun sistem.

Analisis berorientasi objek (OOA) adalah metode analisis untuk memeriksa kebutuhan-kebutuhan dari perspektif kelas-kelas dan objek-objek yang ditemukan di domain masalah.

Selama analisis dan tahap perancangan, pengembangan mempunyai dua tugas, yaitu :

1. Identifikasi kelas-kelas dan objek-objek pembentuk pada domain masalah, (disebut abstraksi masalah).
2. Menemukan struktur dimana himpunan-himpunan objek yang bekerja sama menyediakan perilaku-perilaku yang memenuhi kebuthuan-kebutuhan masalah (disebut mekanisme implementasi).

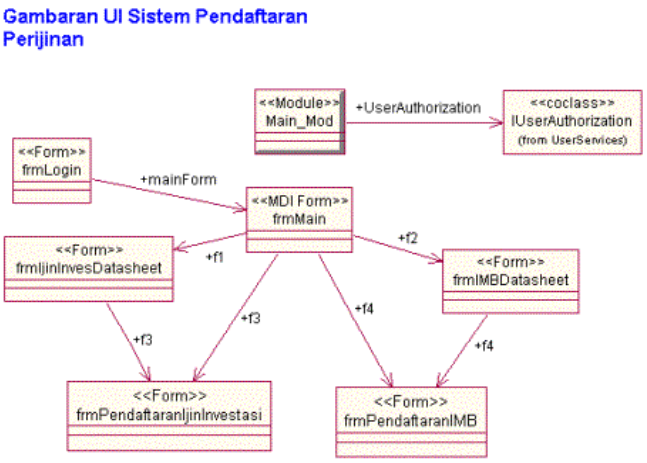
Dua bagian penting dari pengertian perancangan berorientasi objek adalah:

1. Menuntut menuju dekomposisi berorientasi objek
2. Menggunakan notasi-notasi berbeda untuk mengekspresikan model-model perancangan logik berbeda (struktur kelas dan objek) dan perancangan fisik (arsitektur modul dan proses), serta juga model statis dan dinamis sistem yang sedang dirancang.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi pendukung OOP (*Object Oriented Programming*) mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya dan pendekatan OOP lebih mudah

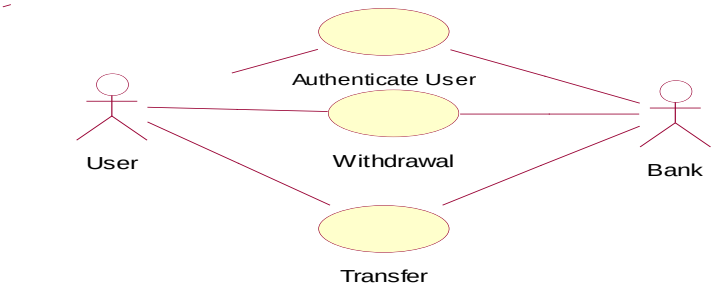
dikembangkan dan dirawat. Dalam UML (*Unified Modelling Language*) yang dapat dikelompokan berdasarkan sifatnya yaitu statis atau dinamis jenis diagramnya adalah sebagai berikut :

1. Diagram Kelas (*Class Diagram*) bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi serta relasi-relasi. Diagram ini umum dijumpai pada pemodelan berorientsi objek.



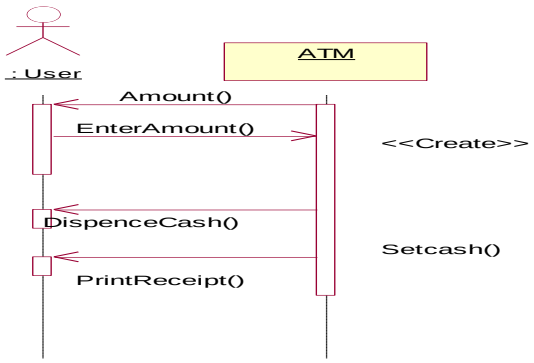
Gambar 3.3 : *Class Diagram* (*Modul Unaki, 2003, hal: 53*)

2. Diagram *Use-Case (Use-Case Diagram)* bersifat statis. Diagram ini memperlihatkan himpunan *use case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna.



Gambar 3.4 : *Use Case Diagram* (*Modul Unaki, 2003, hal :32)*

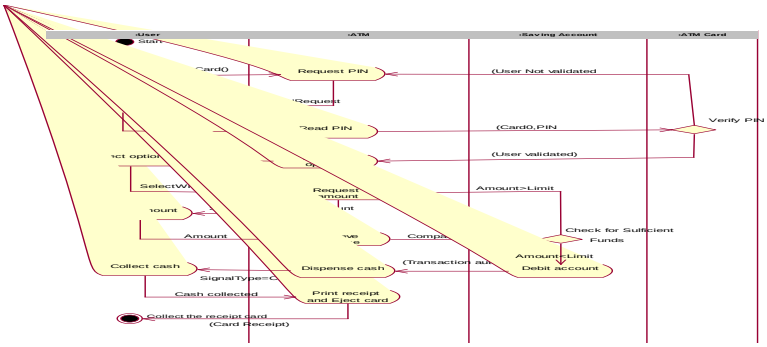
3. Diagram Sekuen (*Sequence Diagram*) bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan (*message*) dalam suatu waktu tertentu.



Gambar 3.5 : *Sequence Diagram* (*Modul Unaki, 2003, hal: 39)*

4. Diagram aktivitas, (*Activity Diagram*) bersifat dinamis. Diagram ini adalah diagram tipe khusus dari *digram state* yang memperlihatkan aliran dari suatu aktifitas ke aktifitas yan lainnya dalam suatu sistem. Diagram

ini penting dalam pemodelan fungsi-fungsi dalam satu sistem dan memberi tekanan pada aliran kendali antar objek.



Gambar 3.6 : *Activity Diagram* (*Modul Unaki, 2003, hal: 62)*

3.3. Evaluasi *Software*

3.3.1. *Black Box*

Pengujian *Software* adalah suatu cara atau teknik untuk menguji perangkat lunak yang digunakan secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan.

Pengujian *black-box* sering disebut juga dengan pengujian tingkah laku (*behavior testing*) yang lebih terfokus kepada kebutuhan fungsional dari perangkat lunak, pengujian *black-box* memungkinkan pembuat perangkat lunak untuk menentukan kondisi yang terjadi untuk suatu masukan yang akan menjalankan semua kebutuhan fungsional (Pressman 2002:551).

Jadi dapat penulis simpulkan bahwa Pengujian *Black Box* adalah pengujian yang dilakukan untuk antarmuka Perangkat Lunak. Pengujian ini dilakukan untuk

memperlihatkan demo bahwa fungsi-fungsi Perangkat Lunak bekerja dengan baik dalam arti semua masukan diterima dengan benar dan keluaran yang dihasilkan benar-benar tepat dan pengintegrasian dari eksternal data berjalan dengan baik.

Pengujian *black-box* dilakukan untuk menentukan beberapa macam kesalahan, yaitu :

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau *database* eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

Pengujian ini dilakukan pada tahap akhir dalam membuat perangkat lunak tidak seperti pengujian *white-box* yang dilakukan di awal pembuatan, hal tersebut dikarenakan pengujian *black-box* dengan sengaja menghiraukan struktur kendali dimana perhatian lebih di utamakan pada domain informasi.

Teknik pengujian konvensional yang termasuk pengujian Kotak Hitam (*Black Box*) adalah sebagai berikut :

1. *Graph based testing* (pengujian yang didasarkan pada grafik)
2. *Equivalence partitioning* (menyekat persamaan)
3. *Comparison testing* (pengujian perbandingan)
4. *Orthogonal testing* (pengujian orthogonal)

Pada pengujian Kotak Hitam (*Black Box*), kita mencoba beragam masukan dan memeriksa keluaran yang dihasilkan. Kita dapat mempelajari apa yang dilakukan kotak, tapi tidak mengetahui sama sekali mengenai cara konversi dilakukan.

Black-Box Testing terfokus pada spesifikasi fungsional dari perangkat lunak *Teste*r dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. *Black Box Testing* bukanlah solusi alternatif dari *White-Box* Testing tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh *White-Box Testing.*

Black-Box Testing cenderung untuk menemukan hal-hal berikut:

- Fungsi yang tidak benar atau tidak ada
- Kesalahan antarmuka (*interface errors*)
- Kesalahan pada struktur data dan akses basis data
- Kesalahan performansi (*performance errors*)
- Kesalahan inisialisasi dan terminasi.

Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:

- Bagaimana fungsi-fungsi diuji agar dapat dinyatakan *valid*?
- Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
- Apakah sistem sensitif pada input-input tertentu?
- Bagaimana sekumpulan data dapat diisolasi?
- Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?

- Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

Equivalence Partitioning

Membagi input menjadi kelas-kelas data yang dapat digunakan untuk menggenerasi kasus uji. Bertujuan untuk menemukan kelas-kelas kesalahan. Berdasarkan pada kesamaan kelas-kelas kondisi input. Sebuah kelas yang ekivalen merepresentasikan kumpulan status/kondisi yang *valid* atau tidak *valid* Sebuah kondisi *input* dapat berupa nilai numerik yang spesifik, rentang nilai, kumpulan nilai yang berkaitan, atau kondisi boolean. Kelas ekivalen dapat didefinisikan dengan kondisi berikut:

- Jika kondisi input mensyaratkan rentang nilai atau nilai yang spesifik, maka sebuah kelas ekivalen yang valid dan dua buah kelas ekivalen yang tidak *valid* akan terbentuk
- Jika sebuah kondisi input mensyaratkan sebuah boolean atau anggota dari sebuah himpunan, maka sebuah kelas ekivalen yang valid dan sebuah kelas ekivalen yang tidak valid akan terbentuk.
- Kasus uji untuk setiap domain input item data harus dikembangkan dan dieksekusi.

Aturan Kelas Ekivalen

- Jika sebuah kondisi mengacu pada nilai masukan yang memiliki rentang maka nilai yang ada dalam rentang masuk menjadi sebuah kelas ekivalen *valid,* dan

nilai masukan diluar kedua sisi ambang batas menjadi dua buah kelas ekivalen yang tidak *valid.*

- Jika sebuah kondisi mengacu pada satu atau sejumlah nilai masukan dengan nilai tertentu maka akan menjadi sebuah kelas ekivalen yang *valid* yang berisi satu atau sejumlah nilai yang valid dan dua buah kelas ekivalen yang tidak valid yaitu kelas yang berisi masukan kosong dan masukan diluar nilai yang dispesifikasikan.
- Jika sebuah kondisi mengacu pada himpunan nilai masukan tertentu maka akan menjadi sejumlah anggota himpunan kelas ekivalen yang *valid* dan sebuah kelas ekivalen yang tidak valid dimana jika masukan memiliki nilai diluar anggota himpunan.
- Jika sebuah kondisi mengekspresikan sebuah kalimat yang berisi "harus" misal, awal masukan harus berupa karakter maka akan menjadi sebuah kelas ekivalen yang valid yaitu semua masukan yang diawal karakter dan sebuah kelas ekivalen yang tidak valid yaitu masukan yang tidak diawali karakter.
- Jika ada alasan yang menyatakan bahwa elemen di dalam sebuah kelas ekivalen diproses dengan cara yang berbeda maka kelas ekivalen itu harus dibagi-bagi lagi menjadi kelas-kelas ekivalen yang lebih kecil berdasarkan pendekatan lojik dari sub-program pada level algoritma.
- Jika sub-program memiliki beberapa masukan yang independen, maka kelas ekivalen atau kasus uji harus dibuat berdasarkan permasukan dan kombinasi

kemungkinan per masukan dimana permasukan dibuat dua buah kelas ekivalen yang valid dan sebuah kelas ekivalen yang tidak *valid*.

- Tulis kasus uji untuk setiap kelas ekivalen yang tidak valid

  Contoh:

  Sebuah spesifikasi menyatakan untuk memasukkan masukan berupa tipe dan nomor mobil.

- Untuk menguji sebuah sub-program yang memiliki beberapa masukan yang saling bergantung maka kelas ekivalen harus dibuat berdasarkan kombinasi masukan.

Boundary Value Analysis / Limit Testing

Banyak kesalahan terjadi pada kesalahan masukan. BVA mengijinkan untuk menyeleksi kasus uji yang menguji batasan nilai input. BVA merupakan komplemen dari equivalence partitioning. Lebih pada memilih elemen-elemen di dalam kelas ekivalen pada bagian sisi batas dari kelas. Contoh:

- Untuk rentang yang dibatasi a dan b maka uji (a-1), a, (a+1), (b-1), b, (b+1).
- Jika kondisi input mensyaratkan sejumlah n nilai maka uji dengan sejumlah (n-1), n dan (n+1) nilai.
- Aplikasikan dua aturan sebelumnya pada kondisi *output* (buat *table* pengujian hasil outputnya untuk nilai maksimal dan minimal).
- Jika struktur data *internal* dari program memiliki cakupan (misal: ukuran *buffer*, batas array) gunakan data input yang menguji batas cakupan.

Comparison Testing

Pada beberapa aplikasi *reliability* dari sebuah perangkat lunak sangat penting. Redundansi perangkat keras dan perangkat lunak mungkin digunakan untuk meminimalisir kesalahan (*error*). Untuk redundansi perangkat lunak, gunakan tim yang terpisah untuk mengembangkan setiap versi perangkat lunak yang independen. Uji setiap versi dengan data yang sama untuk memastikan semua versi menghasilkan keluaran yang sama. Jalankan semua versi dengan paralel dan perbandingan keluaran secara *real-time*. Walau hanya dijalankan sebuah versi pada akhirnya, untuk beberapa aplikasi yang penting dapat mengembangkan versi independen dan menggunakan *comparison testing* atau *back-to-back testing*. Ketika *output* dari versi berbeda, maka setiap versi diinvestigasi jika ada kemungkinan *defect*. Metode ini tidak untuk menemukan kesalahan dari spesifikasi.

Sample and Robustness Testing

- *Sample Testing* melibatkan beberapa nilai yang terpilih dari sebuah kelas ekivalen, mengintegrasikan nilai pada kasus uji, nilai-nilai yang terpilih mungkin dipilih dengan urutan tertentu atau interval tertentu.
- *Robustness Testing* data input dipilih diluar spesifikasi yang telah didefinisikan Tujuan dari pengujian ini adalah membuktikan bahwa tidak ada kesalahan jika masukan tidak valid.

Behavior Testing dan Performance Testing

- *Behavior Testing*

Hasil uji tidak dapat dievaluasi jika hanya melakukan pengujian sekali, tapi dapat dievaluasi jika pengujian dilakukan beberapa kali, misalnya pada pengujian struktur data stack.

- *Performance Testing*

Mengevaluasi kemampuan program untuk beroperasi dengan benar dipandang dari sisi acuan kebutuhan misalnya: aliran data, ukuran pemakaian memori, kecepatan eksekusi, dll, untuk mencari tahu beban kerja atau kondisi konfigurasi program, spesifikasi mengenai performansi didefinisikan pada saat tahap spesifikasi atau desain, dapat digunakan untuk menguji batasan lingkungan program.

Requirement Testing

Spesifikasi kebutuhan yang terasosiasi dengan perangkat lunak (input/output/fungsi/performansi) diidentifikasi pada tahap spesifikasi kebutuhan dan desain. Requirement testing melibatkan pembuatan kasus uji untuk setiap spesifikasi kebutuhan yang terkait dengan program. Untuk memfasilitasinya, setiap spesifikasi kebutuhan bias ditelusuri dengan kasus uji dengan menggunakan traceability matrix.