BAB III

TINJAUAN UMUM PT. SANBE FARMA

3.1 Objek Penelitian

Dalam sub bab ini, penulis akan menguraikan mengenai objek penelitian yang meliputi sejarah PT Sanbe Farma, struktur organisasi yang akan diuraikan sebagai berikut :

3.1.1 Sejarah dan Perkembangan PT. Sanbe Farma

PT. Sanbe Farma secara resmi didirikan pada tanggal 28 Juni 1974 oleh Drs. Jahja Santoso, Apt., seorang farmasis yang lulus dengan predikat *cum laude* dari ITB. Pada tahun 1975 pabrik pertama di Jl. Kejaksaan no.35 Bandung mulai melakukan produksi sebagai industri rumah tangga (*home industry)* dengan jumlah karyawan hanya 4 orang termasuk Bapak Jahja Santoso. Produk pertama yang diproduksi adalah Kapsul Colsancetine®.

Pada tahun 1980, PT. Sanbe Farma berpindah lokasi ke Jl. Industri 1 no.9 Cimahi dengan luas bangunan 8000 m2 dan luas lahan 10.000 m2. Hal ini disebakan karena adanya larangan Pemda tentang lokasi industri di pusat kota dan di tengah pemukiman penduduk. Bangunan ini dikenal dengan PT. Sanbe Farma Unit I dan mulai memproduksi produk non penisilin, non sefalosporin, hormon, dan obat hewan (*veterinary*) pada tahun 1982.

PT. Sanbe Farma mulai berkembang dengan memproduksi produk *OTC* (*Over The Counter*) pada tahun 1992. Salah satu produk *OTC* yang memenangkan *grand*

price "The Most Popular Brand " di Indonesia tahun 1997 dan 1999 adalah Sanaflu®, non- drowsy cold and flu. Selain itu, pada tahun 2000 PT. Sanbe Farma memproduksi Poldan Mig®, obat migrain non resep pertama di Indonesia.

Divisi *veterinary* telah memasarkan sekitar 64 produk kesehatan hewan unggas dan peternakan termasuk vaksin pada tahun 1985. Pada tahun 200*5 veterinary* mulai memproduksi beberapa obat hewan untuk ikan dan udang (*aquaculture*). Divisi ini telah memperoleh sertifikat *GMP* (*Good Manufacturing Practices)* untuk 7 jenis produk. Aktivitas di unit I terus berkembang sampai saat ini.

Pada tahun 1996 bangunan PT. Sanbe Farma unit II didirikan untuk memenuhi tuntutan produksi yang semakin besar dan sesuai dengan CPOB, dimana bangunan untuk produk penisilin dan sefalosporin harus diproduksi di bangunan terpisah. Luas bangunan unit II adalah 5.600 m2 (5 lantai) dan luas lahan 4.900 m2. Unit II memproduksi khusus produk beta laktam dan sefalosporin dengan berbagai macam bentuk sediaan. *San-Qlin-Eq* juga terdapat di unit II, sebagai laboratorium studi bioavabilitas dan bioekivalensi untuk pengujian klinik dan preklinik produkproduk PT. Sanbe Farma dan juga menerima order dari perusahan lain.

Gedung obat jadi (*Finished Good Warehouse*) dengan luas bangunan 6.160 m2 (3 lantai) dan luas lahan 5.980 m2 dibangun pada tahun 2003. Gedung Obat Jadi (GOJ) adalah tempat menyimpan obat jadi hasil pengemasan dari unit I, II, dan III. GOJ dilengkapi dengan *cool storage* untuk penyimpanan vaksin dan lemari khusus untuk penyimpanan obat psikotropika. Bangunan unit III dan Caprifarmindo Laboratories mulai difungsikan pada tahun 2005 dengan luas bangunan 29.000m2

dan luas lahan ±200.000 m2. Pembangunan unit III mengacu pada CPOB Australia, yang menjadikannya industry farmasi pertama yang dikendalikan oleh *SCADA* (*Supervisory Computer Automatization Data Acquisition*). Di unit III juga terdapat *WWTP* (*Water Waste Treatment Plant*) juga untuk pengolahan limbah dari unit I dan II.

3.1.2 Visi dan Misi

Visi :

Menjadi penyedia produk generik dan OTC yang dikenali diseluruh dunia.

Misi :

1. Integrity

   Sebagai penyedia produk dan pelayanan yang berkualitas

2. Highest Regard for People

   Orang merupakan pondasi kesuksesan Sanbe, dan Sanbe mengkaji, memotivasi, dan memakai orang dengan kemampuan dan kompetensi yang baik.

3. Kepuasan Konsumen

   Sanbe akan bergerak cepat untuk memenuhi kebutuhan konsumen tepat waktu dengan kemampuan terbaik Sanbe.

4. Komunitas

Sanbe akan mengirim produknya yang berkualitas untuk meningkatkan pelayanan kesehatan.

5. Inovasi

   Sanbe akan selalu melakukan inovasi

6. Team Work

   Saling bekerja sama dalam satu team dan saling percaya.

7. Performance

   Sanbe selalu menetapkan standar yang tinggi dan mencapai sukses hari demi hari melebihi standar yang ditetapkan dengan rasa tidak ketidak puasan untuk mendapatkan kemenangan

8. Leadership

   Sanbe akan memimpin apapun yang kami lakukan dengan cara kami yang unik dan memotivasi tidak hanya anggota Sanbe saja tetapi juga di dalam industri.

3.1.3 Lokasi dan Sarana Produksi

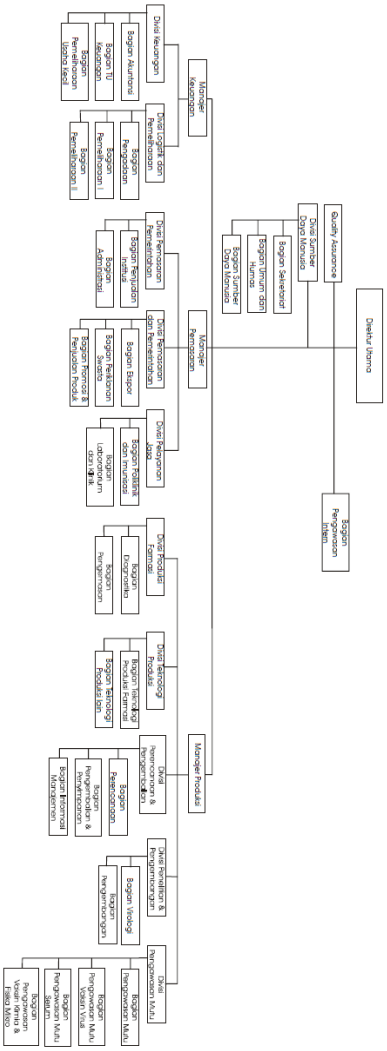
PT. Sanbe Farma mempunyai beberapa tempat untuk melakukan aktivitas produksi yaitu:

a. PT. Sanbe Farma unit I di Jl. Industri 1 no.9 Cimahi memproduksi produk non beta laktam , non sefalosporin , dan obat hewan (*veterinary*)
b. PT. Sanbe Farma unit II di Jl. Leuwigajah no.162 Cimahi memproduksi produk betalaktam (lantai 2) dan sefalosporin (lantai 4)

c. PT. Sanbe Farma unit III Jl. Industri Cimareme no.8 Padalarang memproduksi *SVP* (*Small Volume Parenteral* = injeksi volume kecil), *LVP* (*Large Volume Parenteral* = injeksi volume besar), tetes mata, sediaan steril semisolid, sediaan serbuk injeksi steril.
d. Gudang bahan baku (GBB) berada di masing-masing unit, gudang bahan pengemas berada di Jl. Leuwigajah no.174 Cimahi, dan Gudang obat jadi (GOJ) di Jl. Leuwigajah no.184 Cimahi. Jadi, Total luas bangunan PT. Sanbe Farma (unit I,II, dan III, GBB, GOJ, dan Caprifarmindo) 52.360 m2 dan luas lahan ± 241.880 m2.

3.1.4 Struktur Organisasi

Struktur organisasi dari PT.Sanbe Farma :



Gambar 3.1 Struktur Organisasi Sanbe Farma

3.1.5 Deskripsi Kerja

Berdasarkan struktur organisasi di atas, maka dapat diuraikan mengenai fungsi dan tugas masing-masing bagian. Berikut adalah fungsi dan tugas dari masing-masing bagian yang terlibat pada PT. Sanbe Farma :

1. Direktur Utama

   Tugas dan fungsi Direktur Utama, yaitu:

   a. Bertanggung jawab sepenuhnya terhadap seluruh kegiatan perusahaan.
   b. Membentuk susunan organisasi dan menunjuk karyawan.
   c. Melakukan pengolahan data, analisis dan evaluasi hasil usaha dalam rangka meningkatkan daya guna dan hasil guna milik perusahaan.
   d. Mengangkat dan memberhentikan karyawan.
   e. Mengadakan hubungan dengan pihak luar terutama mengenai hal-hal yang menyangkut kebijaksanaan pengembangan perusahaan dan kebijaksanaan yang bersifat umum.

2. Quality Assurance

   Tugas dan fungsi dari Quality Assurance, yaitu :

   a. Mengusulkan pelatihan-pelatihan system manajemen mutu.
   b. Mengkoordinir penyusunan dokumen sistem mutu.
   c. Meninjau seluruh dokumen sistem mutu.

d. Mengkoordinir pelaksanaan kampanye mutu/kesadaran tentang sistem manajemen mutu.
e. Meninjau dokumen sistem mutu minimal 1 tahun sekali dan merencanakan penyempurnaan secara bertahap.
f. Mengusulkan jadwal rapat tinjauan manajemen kepada kepala direktur, menetapkan agenda-agenda rapat dan bertindak sebagai notulis pada rapat tinjauan manajemen.
g. Menyusun jadwal audit mutu internal dan mengatur penugasan tim audit (audit mutu internal dilaksanakan minimal 6 bulan sekali atau lebih sering sesuai dengan tipe aktivitas yang harus diaudit dan dengan mempertimbangkan hasil audit sebelumnya).
h. Menganalisis rangkuman hasil audit dan melakukan tindak lanjut agar pada periode audit berikutnya dicapai implementasi sistem yang lebih efektif.
i. Menganalis rangkuman dari hasil pengukuran quality objektif dari seluruh unit kerja dan mengusulkan tindakan perbaikan dan pencegahan.
j. Melakukan tindak lanjut terhadap hasil audit oleh lembaga sertifikasi mutu setiap 6 bulan.

3. Divisi Sumber Daya Manusia

Tugas dan fungsi Divisi Sumber Daya Manusia, yaitu :

a. Mengkoordinasikan perumusan perencanaan dan pemmberdayaan pegawai (man Power Planning), sesuai kebutuhan Perusahaan.
b. Mengkoordinasikan perumusan sistem pengadaan, penempatan dan pengembangan pegawai.
c. Menkoordinasikan perumusan sistem dan kebijakan imbal jasa pegawai dengan mempertimbangkan “internal / external equity”.
d. Menyelenggarakan Sistem Informasi SDM dalam suatu database kepegawaian.
e. Melakukan kajian dan evaluasi terhadap efektifitas program dan kontribusi peraturan bagi perkembangan perusahaan.
f. Mengevaluasi hasil penilaian kinerja seluruh pegawai yang telah dilaksanakan bersama para atasan langsung.
g. Menyelenggarakan kegiatan pemeliharaan dan kebersihan gedung, halaman, taman, dan lapangan parkir kantor pusat serta kantor divisi keamanan.
h. Menyelenggarakan adminsitrasi, penempatan, penyimpanan, dan penggunaan peralatan, inventaris, fasilitas kantor.

i. Menyediakan perlengkapan dan peralatan kerja yang diperlukan kantor pusat dan atau perusahaan (computer, kendaraan dinas, kendaraan operasional).

4. Bagian Sekretariat

Tugas dan fungsi dari bagian secretariat, yaitu :

a. Pelaksanaan koordinasi perencanaan program, anggaran dan laporan perusahaan.
b. Pelaksanaan pembinaan organisasi dan ketatalaksanaan.
c. Pengelolaan administrasi kepegawaian.
d. Pengelolaan surat menyurat, dokumentasi, rumah tangga perusahaan, kearsipan dan perpustakaan.
e. Pemeliharaan rutin gedung dan perlengkapan/peralatan kantor
f. Pelaksanaan hubungan masyarakat dan keprotokolan.
g. Pengelolaan survey kesehatan daerah skala kota.
h. Penyelenggaraan bimbingan dan pengendalian operasionalisasi bidang kesehatan.
i. Pelaksanaan administrasi perizinan/rekomendasi.

5. Bagian umum dan Humas

Tugas dan fungsi dari bagian umum dan humas, yaitu:

a. Pelaksanaan urusan kepegawaian, organisasi, dan tata laksana.
b. Pelaksanaan urusan tata usaha, perusahaan dan perlengkapan.

c. Pelaksanaan urusan hubungan masyarakat dan hubungan antar lembaga.

6. Satuan pengawasan intern

Tugas dan fungsi dari satuan pengawasan intern yaitu:

a. Menjabarkan visi dan misi perusahaan kedalam konsep audit internal dan pengawasan manajemen pada tingkat korporasi.
b. Mengendalikan kegiatan pemeriksaan internal agar mencapai sasaran dan sesuai dengan standar profesi audit internal.
c. Melakukan review atas kebijakan perusahaan sebagai akibat perubahan lingkungan dan mengusulkan perbaikan kebijakan yang dinilai tidak serasi lagi dengan pola kerja perusahaan.
d. Memantau dan melakukan pengujian terhadap implementasi dari kebijakan perusahaan.
e. Memberikan peringatan dini/awal kepada direktur utama atas potensi masalah yang mungkin akan timbul akibat tidak dilaksanakannya kebijakan perusahaan.
f. Membina hubungan dengan instansi luar perusahaan agar diperoleh informasi lebih dini tentang perkembangan Internal Audit Standard yang berlaku.

7. Manajer Keuangan

Tugas dan fungsi dari manajer keuangan, yaitu :

a. Mengurus keuangan dalam perusahaan.

b. Mengeluarkan dana untuk keprluan perusahaan.
c. Memeriksa laporan tahunan anggaran keuangan perusahaan.
d. Melaksanakan kegiatan dan pengawasan administrasi dan keuangan.
e. Merencanakan dan mengusulkan kebutuhan pengembangan personil secara professional.
f. Melakukan evaluasi secara berkala terhadap perkerjaan setiap bagian bidang usaha yang dipimpin.
g. Melaporkan kegiatan dan hasil kerja secara berkala dan memberikan usul kepada pihak terkait.

8. Manajer Perusahaan

Tugas dan fungsi dari manajer perusahaan, yaitu:

a. Melaksanakan kegiatan pemasaran terhadap produk yang ada di perusahaan.
b. Menghimpun dan mengolah data tentang bannyaknya konsumen perusahaan sesuai dengan kebijakan perusahaan.
c. Melaporkan kegiatan da hasil kerja secara berkala dan memberikan usul kepada pihak terkait.

9. Manajer Produksi

Tugas dan fungsi dari manajer produksi, yaitu:

a. Membuat rencana kegiatan produksi sesuai dengan informasi dari manajer pemasaran.

b. Melakukan evaluasi secara berkala terhadap hasil pekerjaan.

10. Kepala Bagian Sekuriti

Tugas dan fungsi kepala bagian sekuriti, yaitu:

a. Menjaga dan mengawasi seluruh keamanan di perusahaan.
b. Memberi perintah kepada pegawai sekuriti.
c. Menerima laporan keamanan dari pegawai sekuriti dibawahnya.

11. Pegawai Bagian Sekuriti

Tugas dan fungsi dari Pegawai bagian sekuriti, yaitu :

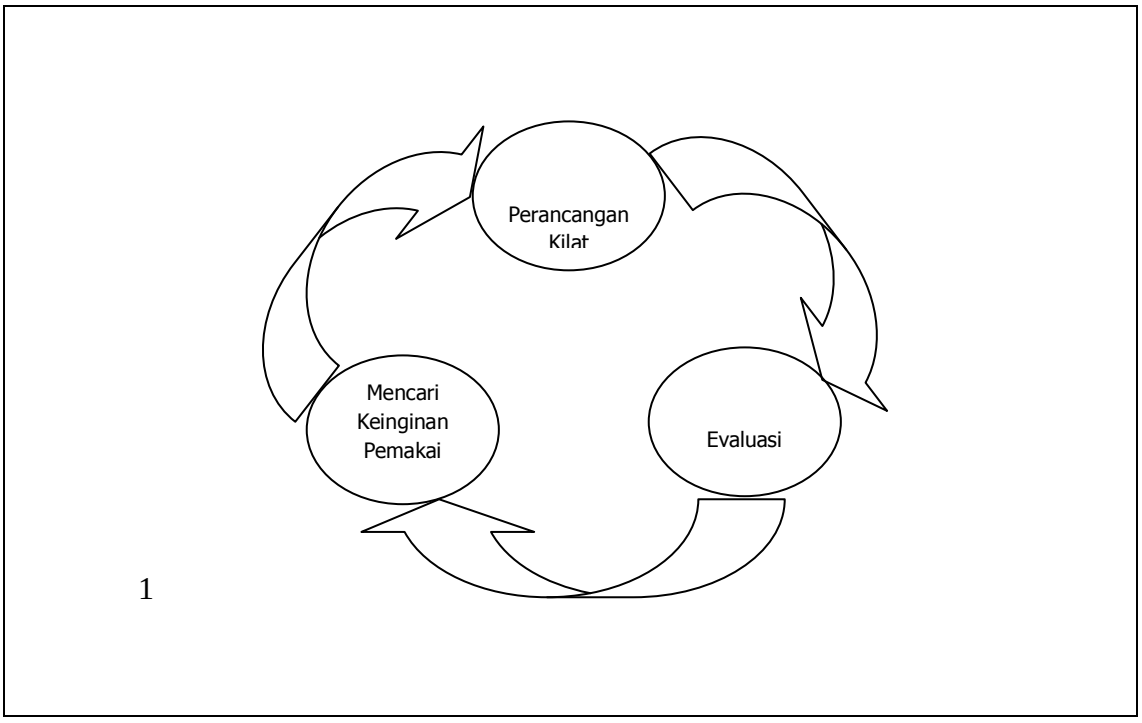
a. Bertugas menjaga keamanan di perusahaan.
b. Menerima perintah dari kepala keamanan.
c. Membuat laporan keamanan untuk kepala keamanan.

3.2 Metode Pengembangan Sistem

Metode yang digunakan dalam membangun perangkat lunak dalam penyusunan skripsi ini adalah **Prototipe** karena tidak melakukan identifikasi kebutuhan output, pemrosesan ataupun input detail, tidak memiliki kepastian efisiensi algoritma, tidak memiliki kemampuan penyesuaian dari sebuah sistem operasi, atau bentuk - bentuk yang harus dilakukan oleh interaksi manusia dan mesin.

Prototipe paradigma dimulai dengan pengumpulan kebutuhan, pengembang dan pelanggan bertemu dan mendefinisikan obyektif keseluruhan dari perangkat lunak, mengidentifikasikan segala kebutuhan yang diketahui, dan area garis besar dimana definisi labih jauh merupakan keharusan kemudian dilakukan ‘perancangan kilat’.

Perancangan kilat membawa kepada konstruksi sebuah prototipe. Prototipe tersebut dievaluasi dan dipakai untuk menyaring kebutuhan pengembangan perangkat lunak. literasi terjadi pada saat prototipe disetel untuk memenuhi kebutuhan pelanggan, dan pada saat yang sama memungkinkan pengembang untuk secara lebih baik memahami apa yang harus dilakukannya.



Gambar 3.2 prototipe Model

[Sumber:"Software Engineering Laboratory" Modul Praktikum Rekayasa Perangkat Lunak. 2004]

Prototipe dimulai dengan pengumpulan kebutuhan, pengembang dan pelanggan bertemu dan mendefinisikan obyektif keseluruhan dari perangkat lunak, mengidentifikasikan segala kebutuhan yang diketahui, dan area garis besar dengan definisi yang jelas merupakan suatu keharusan, kemudian dilakukan 'perancangan

kilat'. Perancangan kilat membawa kepada konstruksi sebuah prototipe. Prototipe tersebut dievaluasi dan dipakai untuk membagi kebutuhan pengembangan perangkat lunak.

Secara ideal prototipe berfungsi sebagai sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Bila prototipe yang sedang bekerja dibangun pengembang harus mempergunakan fragmen-fragmen program yang ada yang memungkinkan program yang bekerja untuk dimunculkan secara cepat.

Pemrograman berorientasi objek sebagai berikut :

"Pemrograman berorientasi objek atau OOP (Object Oriented Programing) adalah suatu cara baru dalam berpikir serta berlogika dalam menghadapi masalah-masalah yang akan dihadapi dengan bantuan komputer."

Seiring dengan munculnya OOP (*Object Oriented Programing*) maka telah banyak bermunculan metodologi pemodelan berorientasi objek, karena banyaknya metodologi yang ada maka tiga metodologis terkenal yaitu Grady Booch, James Rumbaugh, dan Ivar Jacobson mengkombinasikan metode-metode mereka untuk memperoleh notas-notasi yang dapat digunakan seluruh metodologi berorientasi objek yang dikemukakan para metodologis. Mereka membentuk notasi yang dapat digunakan bersama yaitu : *Unified Modeling Language (UML)*.

Pengertian UML sebagai berikut :

"UML adalah salah satu bentuk notasi atau bahasa yang sama yang digunakan oleh professional dibidang software untuk menggambarkan atau memodelkan sebuah sistem software."

UML telah disahkan oleh OMG (*Object Management Group*) pada tahun 1997 sebagai standar *de-facto* dari pemodelan OOP dan diterima sebagai salah satu standar industri. Secara garis besar UML merupakan standard bahasa pemodelan untuk pembuatan object-oriented software dan merupakan kombinasi dari:

1. Konsep Pemodelan Data (*Entity Relationship Diagrams*)
2. Pemodelan Bisinis (*Work Flow*)
3. Pemodelan *Object*,
4. Pemodelan Komponen

Spesifikasi UML mendefinisikan sekumpulan diagram grafis sebagai tampilan dari beberapa level abstraksi dan UML dapat digunakan bersama oleh semua proses pada keseluruhan tahap siklus-hidup *(life-cycle)* pengembangan *software* serta pada implementasi ke beberapa teknologi yang berbeda.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram perangkat lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya, yaitu: Grady Booch OOD *(Object-Oriented Design)*, Jim Rumbaugh OMT *(Object Modeling Technique)*, dan Ivar Jacobson OOSE *(Object-Oriented Software Engineering)*.

3.3 *Object Management Group* (OMG)

OMG adalah konsorsium yang beranggotakan lebih dari 850 perusahaan untuk mendefinisikan standar-standar teknilogi objek termasuk CORBA (*Common Object Request Broker Architecture*) yang dibentuk pada tahun 1989 yang ditujukan untuk mempromosikan penggunaan teknologi berarah obyek pada aplikasi software.

Unified Modeling Language (UML) merupakan sistem arsitektur yang bekerja dalam OOAD (*Object-Oriented Analysis/Design*) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, mengkontruksi, dan mendokumentasikan *artifact* (sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa software, dapat berupa model, deskripsi, atau software) yang terdapat dalam sistem software. UML merupakan bahasa pemodelan yang paling sukses dari tiga metode OO yang telah ada sebelumnya, yaitu Booch, OMT (*Object Modeling Technique*), dan OOSE (*Object-Oriented Software Engineering*).

UML merupakan kesatuan dari dari ketiga pemodelan tersebut dan ditambah kemampuan lebih karena mengandung metode tambahan untuk mengatasi masalah pemodelan yang tidak dapat ditangani ketiga metode tersebut. UML dikeluarkan oleh OMG (Object Management Group, Inc) yaitu organisasi internasional yang dibentuk pada 1989, terdiri dari perusahaan sistem informasi, *software developer*, dan para *user* sistem komputer.

3.4 Artifact UML

Diagram – diagram yang digunakan untuk mendefinisikan *UML* adalah sebagai berikut:

a. Use Case Diagram

Sebuah *use case* menggambarkan suatu urutan interaksi antara satu atau lebih aktor dan sistem. Dalam fase *requirements*, model *use case* mengambarkan sistem sebagai sebuah kotak hitam dan interaksi antara aktor dan sistem dalam suatu bentuk naratif, yang terdiri dari input user dan respon-respon sistem. Setiap *use case* menggambarkan perilaku sejumlah aspek sistem, tanpa mengurangi struktur internalnya. Selama pembuatan model *use case* secara pararel juga harus ditetapkan objek-objek yang terlibat dalam setiap *use case*.

Use case Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskripsi fungsi yang disediakan oleh sistem kedalam bentuk teks sebagai dokumentasi dari *use case symbol* namun dapat juga dilakukan dalam *activity diagrams*.

Use case digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat user) dan bukan bagaimana fungsi yang ada di dalam sistem. Simbol yang digunakan yaitu:

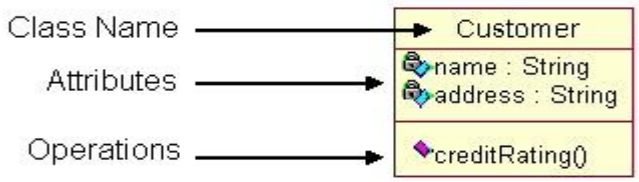


Gambar 3.3 *Use Case* Model

b. Class Diagram

Menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara: *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainnya) atau *package* (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class diagram.*

Suatu *class* biasanya terdiri dari 3 bagian, yaitu nama, atribut, dan operasi. Berikut adalah contoh dari suatu *class*:



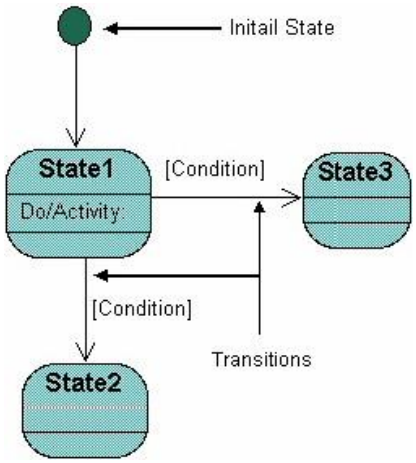
Gambar 3.4 Bagian-bagian dari c*lass*

c. Statechart Diagram

Statechart Diagram merupakan 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 dengan 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 titik akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.



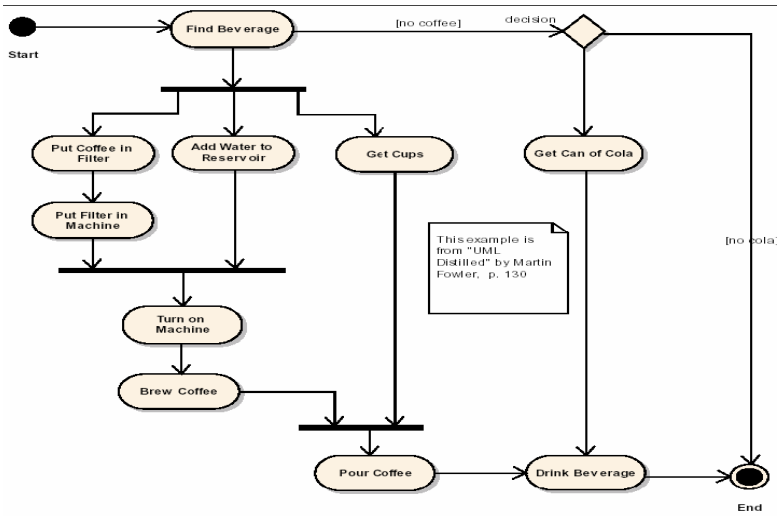
Gambar 3.5 *State Diagram*

d. Activity Diagram

Activity Diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing - masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana aktivitas itu berakhir. *Activity diagram* juga dapat menggambarkan proses pararel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus,dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di *triger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *Activity diagram* tidak menggambarkan

behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak , tetapi lebih menggambarkan proses - proses dan jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama dengan *state*, standar *UML* menggunakan segi empat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses pararel (*fork on join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

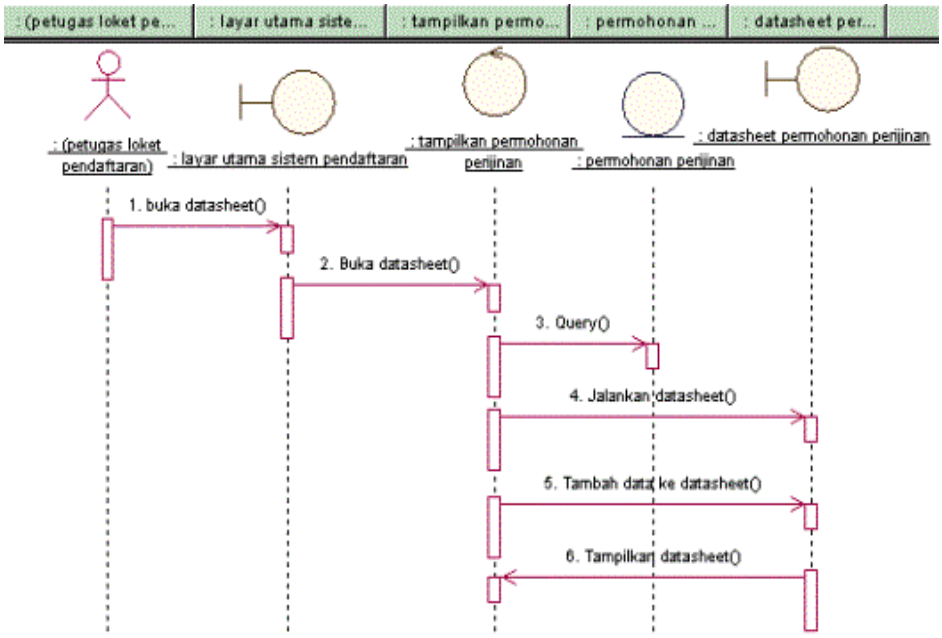


Gambar 3.6 *Activity Diagram*

e. *Sequence Diagram*

Menggambarkan kolaborasi dinamis antara sejumlah *object*. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

Komponen utama *sequence diagram* terdiri atas objek yang dituliskan dengan kotak segiempat bernama pesan diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan proses vertikal.



Gambar 3.7 *Sequence Diagram*

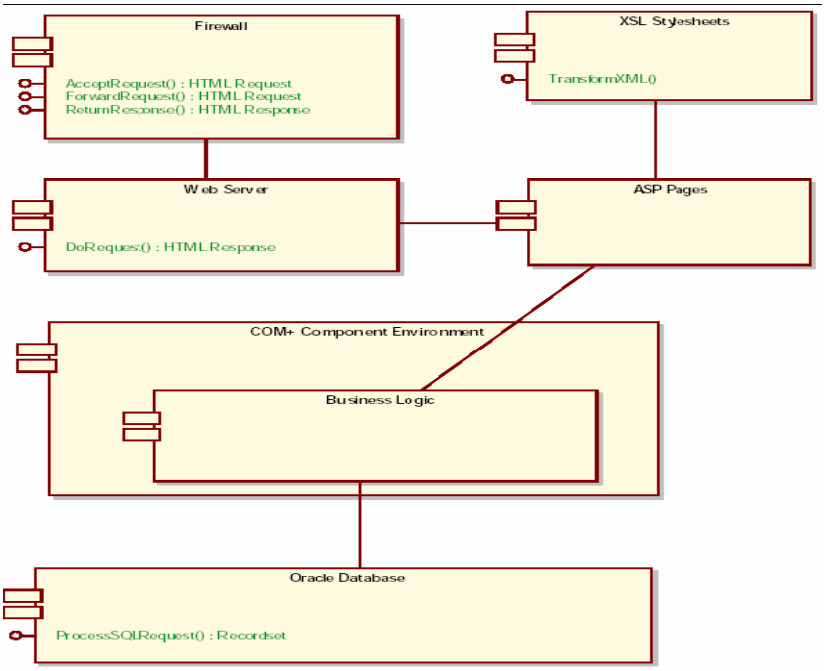
f. *Colaboration Diagram*

Menggambarkan kolaborasi dinamis seperti *sequence diagrams*. Dalam menunjukkan pertukaran pesan, *collaboration diagrams* menggambarkan object dan

hubungannya (mengacu ke konteks). Jika penekanannya pada waktu atau urutan maka gunakan *sequence diagrams*, tapi jika penekanannya pada konteks gunakan *collaboration diagrams*.

g. Component Diagram

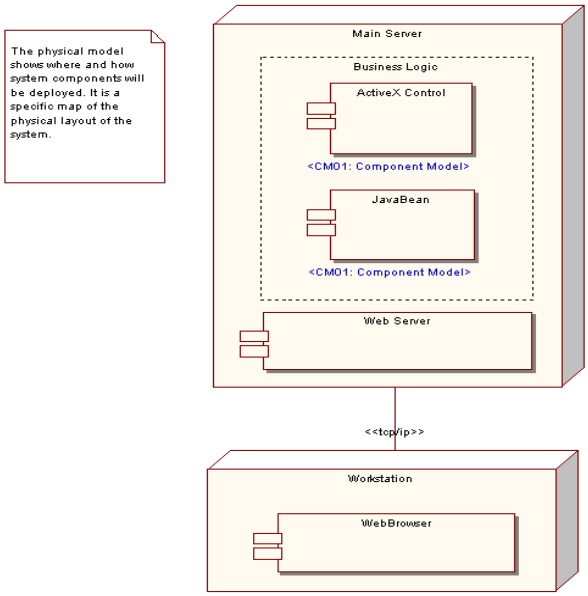
Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency)* diantaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time, link time*, maupun *run time.* Umumnya komponent terbentuk dari beberapa *class* atau *package* tapi juga dari komponen - komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.



Gambar 3.8 *Component Diagram*

h. Deployment Diagram

Deployment / phisycal diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal - hal lain yang bersifat fisikal. Sebuah *node* adalah server, *workstation* atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.



Gambar 3.9 *Deployment Diagram*

3.5 Semantik Dalam UML

Seperti bahasa - bahasa lainnya, *UML* mendefinisikan notasi dan *syntax* / semantik. Notasi *UML* merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan *UML syntax* mendefinisikan bagaimana bentuk - bentuk tersebut dapat dikombinasikan. Notasi *UML* terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu *Grady Booch OOD (Object-Oriented Design)*, *Jim Rumbaugh OMT (Object Modeling Technique)*, dan *Ivar Jacobson OOSE (Object-Oriented Software Engineering)*.

OMG telah menetapkan semantik (makna istilah) semua notasi *UML* dalam *Model Struktural* dan *Model Behavioral*.

1. *Model Struktural* atau model statis, menekankan pada struktur objek dalam sebuah sistem, menyangkut kelas - kelas, *interface*, *atribute* dan hubungan antar komponen.
2. *Model Behavioral* atau model dinamis, menekankan pada perilaku objek dalam sebuah sistem, menyangkut metode, interaksi, kolaborasi dan *state history*.

3.6 Tujuan UML

Tujuan utama dari perancangan UML adalah :

1. Menyediakan bahasa pemodelan visual yang ekspresif dan siap pakai untuk mengembangkan dan pertukaran model-model yang berarti.
2. Menyediakan mekanisme perluasan dan spesialisasi untuk memperluas konsep-konsep ini.
3. Mendukung spesifikasi independen bahasa pemrograman dan proses pengembangan tertentu.
4. Menyediakan basis formal untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar kakas berorientasi objek.
6. Mendukung konsep-konsep pengembangan level lebih tinggi seperti komponen, kolaborasi, *framework* dan *pattern*.

3.7 Cakupan UML

Model *UML* dapat mencakup banyak perbendaharaan. Diantaranya adalah sebagai berikut :

1. *Things*
   a. *Structural*:
      1. *Use case* : deskripsi interaksi dengan *external actor.*
      2. *Class* : deskripsi untuk objek – objek.
      3. *Interface* : kumpulan operasi yang memberikan service tertentu untuk *class / component.*
      4. *Component* : bagian sistem yang dapat diganti *(replaceable)* dan realisasinya sesuai dengan interface.
   b. *Behaviour*
      1. Interaksi *(message sequence chart)*: pertukaran messages antar objek.
      2. *State machine*: urutan state dari objek dalam berinteraksi dengan objek lain.
   c. *Grouping*

      Package: mekanisme untuk mengumpulkan elemen ke dalam satu set (group).
   d. *Anotasi*

      Catatan atau keterangan (teks) sebagai dokumentasi.
2. *Relationship*
   a. *Dependency*

Hubungan antar element dimana perubahan pada elemen yang satu dapat mempengaruhi elemen yang lain *(dependent)*.

b. Association

Hubungan struktur antara elemen dan bertindak sebagai *link.*

c. Generalization

Hubungan dimana elemen yang special *(child)* mewarisi elemen yang umum *(parent).*

d. Realization

1. Hubungan (semantik) antara 2 elemen, dimana elemen yang satu memberikan kontrak dan elemen yang lain menjamin realisasi kontrak tersebut.
2. Dimana elemen yang special *(child)* mewarisi elemen yang umum *(parent).*

3. *Diagram*: *use case diagram, class diagram, msc diagram*, dan lain – lain.

Selain cakupan – cakupan di atas, ada tiga aspek utama dalam pemodelan sistem yang mampu didukung oleh *UML*:

a. *Functional Model*, untuk menunjukkan fungsionalitas dari suatu sistem dari sudut pandang user atau pengguna. Ini dicapai dengan menggunakan *Use Case Diagram.*

b. *Object Model,* untuk menunjukkan struktur dan substruktur dari suatu sistem dengan menggunakan object, atribut, operasi dan juga asosiasi. Ini dicapai dengan menggunakan *Class Diagram*.

c. *Dynamic Model*, menunjukkan *internal behavior* dan suatu sistem. Ini dicapai dengan menggunakan *Sequence Diagram, Activity Diagram* dan juga *Statechart Diagram*.

3.8 Perancangan Basis Data

Sebelum kita membuat basis data, terlebih dahulu harus dilakukan suatu perancangan. Proses perancangan ini bersifat konseptual. Kita belum menentukan DBMS apa yang akan kita gunakan untuk mengimplementasikan rancangan basis data yang dibuat. Tujuan perancangan basis data adalah untuk mendapatkan skema basis data yang meminimasi terjadinya redudansi dan duplikasi data serta menjaga integritas data. Kebanyakan metode perancangan yang ada berbasis pada model basis data relasional, yaitu struktur data diatur melalui pembuatan tabel-tabel dan keterkaitan antar tabel satu dengan yang lainnya *(*relasi*)*.

a. *Multiplicity*

Multiplicity pada kasus asosiasi menunjukkan bahwa ada sejumlah objek pada sebuah *class* yang berhubungan dengan sebuah objek pada sebuah asosiasi *class*, ada banyak multiplicity yang mungkin untuk dipakai. Sebuah class dapat berhubungan dengan yang lain dalam satu ke satu *(one-to-one)*, satu ke banyak (*one-to-many*), satu ke satu atau lebih (*one-to-one or more*), satu ke nol atau satu, satu ke sebuah interval

tertentu (contoh satu ke lima sampai dengan sepuluh), satu ke sejumlah n atau satu ke serangkaian pilihan (contoh satu ke sembilan atau sepuluh). Pada notasi UML, untuk menampilkan lebih atau banyak digunakan (*). Untuk menunjukkan ‘atau’ digunakan titik dua (..) seperti contohnya 1..* (berarti satu atau lebih). Notasi ‘atau’ bisa juga digunakan tanda koma (,).

b. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut / properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda / fungsi).

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

Class memiliki tiga area pokok :

1. Nama (dan *stereotype)*
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut:

1. *Private,* tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected,* hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public,* dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Diagram demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Sesuai dengan perkembangan *class* model, *class* dapat dikelompokan menjadi *package*. *Class* dapat membuat diagram yang terdiri atas *Package.*

Hubungan antar Class :

1. Asosiasi, yaitu hubungan statis antar class. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukan arah query antar *class.*
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas").
3. Pewarisan, yaitu hubungan hirarkis antar *class. Class* dapat diturunkan dari class lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di *passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

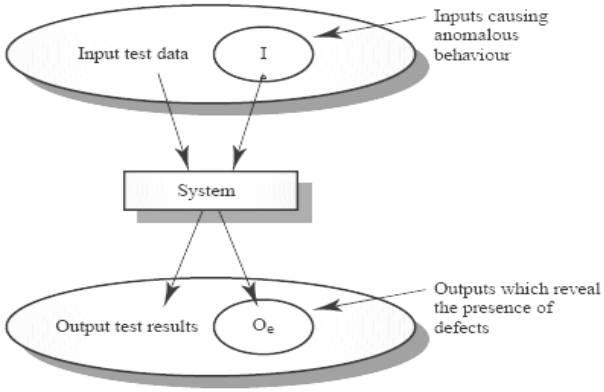
3.9 Pengujian *Black Box*

Pengujian *black-box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black-box* memungkinkan perekayasa perangkat lunak

mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black-box* bukan merupakan alternatif dari teknik *white-box*, tetapi merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalahan daripada metode *white-box*.

Pengujian *black-box* berusaha menemukan kesalahan dalam kategori sebagai berikut :

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.



Gambar 3.10 *Metode Pengujian Black-Box*