BAB II

TINJAUAN PUSTAKA

2.1 Sejarah Instansi/Perusahaan

Sejarah instansi memaparkan tentang sejarah perusahaan, struktur organisaisi dan kegiatan bisnis yang dikerjakan oleh perusahaan.

2.1.1 Sejarah PT. KBM

PT. Kencana Berlian Makmur (KBM) didirikan pada tanggal 03 Maret 2003 di Bandung dengan Surat Keputusan Departemen Kehakiman RI No.C-07100.HT.01.01 Tahun 2003. Pada hari Senin, tanggal 05 Januari 2004 pukul 12.00 WIB PT. Kencana Berlian Makmur disahkan sebagai perseroan atas dasar putusan kehakiman No.48/CN/2003/PN.BDG.P.ST.

PT. KBM berlokasi di Jalan dr. Junjunan No.119 (Pasteur), Bandung. PT. KBM merupakan perusahaan yang bergerak di bidang penjualan kendaraan berjenis sepeda motor dari produk YAMAHA. PT. KBM mempunyai sebelas kantor cabang berupa *dealer-dealer* resmi motor YAMAHA yang tersebar di daerah Jawa Barat. Pada perusahaan ini kegiatan yang dilakukan selain kegiatan penjualan, perusahaan ini juga menyediakan pelayanan perawatan kendaraan motor YAMAHA dan menerima pemesanan semua jenis suku cadang untuk motor-motor besar produk YAMAHA.

2.1.2 Kegiatan dan *Platform* Usaha PT. KBM

Perusahaan ini sebagai *main dealer* produk sepeda motor YAMAHA mempunyai kegiatan memasarkan dan mendistribusikan sepeda motor YAMAHA ke wilayah Jawa Barat.

PT. Kencana Berlian Makmur memasarkan dan mendistribusikan produk sepeda motor (kendaraan roda dua) merk YAMAHA. Adapun produk-produk yang dipasarkan oleh PT. KBM meliputi:

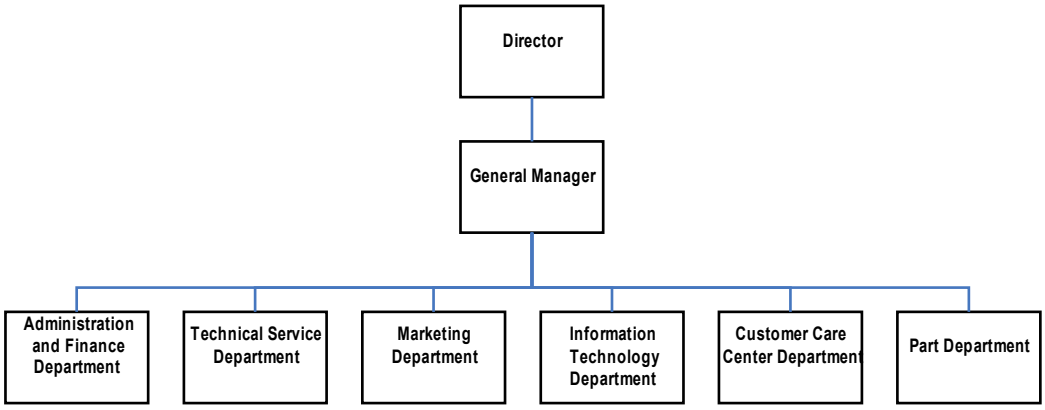
a. Sepeda motor YAMAHA MIO
b. Sepeda motor YAMAHA MIO CW
c. Sepeda motor YAMAHA MIO SOUL
d. Sepeda motor YAMAHA VEGA ZR DB
e. Sepeda motor YAMAHA VEGA ZR DRUM
f. Sepeda motor YAMAHA JUPITER Z NEW
g. Sepeda motor YAMAHA JUPITER Z CW
h. Sepeda motor YAMAHA JUPITER MX CW
i. Sepeda motor YAMAHA JUPITER MX AT CW
j. Sepeda motor YAMAHA VIXION
k. Sepeda motor YAMAHA SCORPIO Z CW

2.1.3 Bagan Struktur Organisasi PT. KBM

PT. KBM mengorganisasikan dengan baik karyawan dalam bagian-bagian pekerjaan menurut jenis pekerjaannya, struktur organisasi perusahaan seperti terlihat pada gambar II.1 sampai dengan gambar II.7.

2.1.3.1 Struktur Organisasi PT.KBM

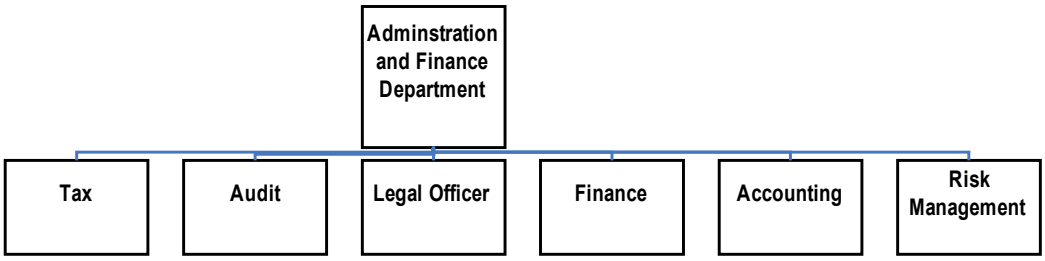
Struktur organisasi PT. KBM seperti terlihat pada gambar II.1.



Gambar II.1 Struktur Organisasi PT. KBM

2.1.3.2 Struktur *Administration and Finance Department*

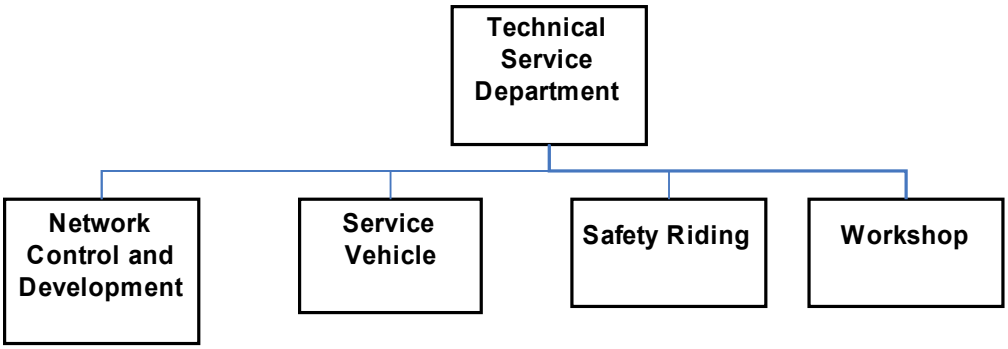
Struktur *Administration and Finance Department* PT. KBM seperti terlihat pada gambar II.2.



Gambar II.2 Struktur *Administration and Finance Department* PT. KBM

2.1.3.3 Struktur *Technical Service Department*

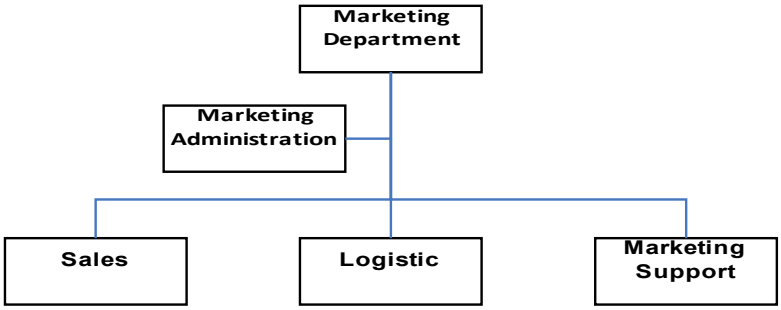
Struktur *Technical Service Department* PT. KBM seperti terlihat pada gambar II.3.



Gambar II.3 Struktur *Technical Service Department* PT. KBM

2.1.3.4 Struktur *Marketing Department*

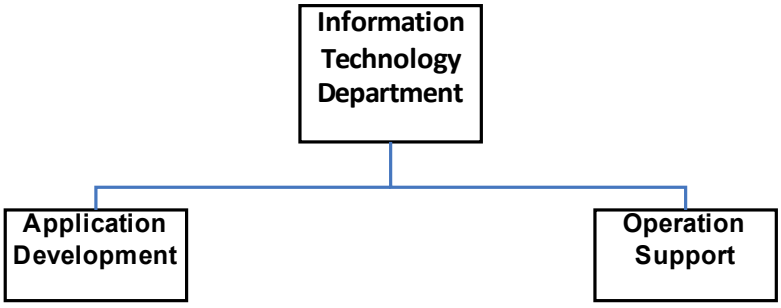
Struktur *Marketing Department* PT. KBM seperti terlihat pada gambar II.4.



Gambar II.4 Struktur *Marketing Department* PT.KBM

2.1.3.5 Struktur *Information Technology Department*

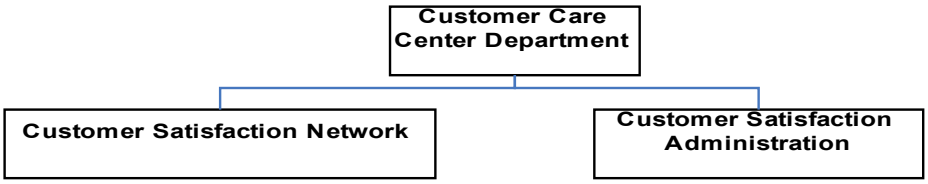
Struktur *Information Technology Department* PT. KBM seperti terlihat pada gambar II.5.



Gambar II.5 Struktur *Information Technology Department* PT. KBM

2.1.3.6 Struktur *Customer Care Center Department*

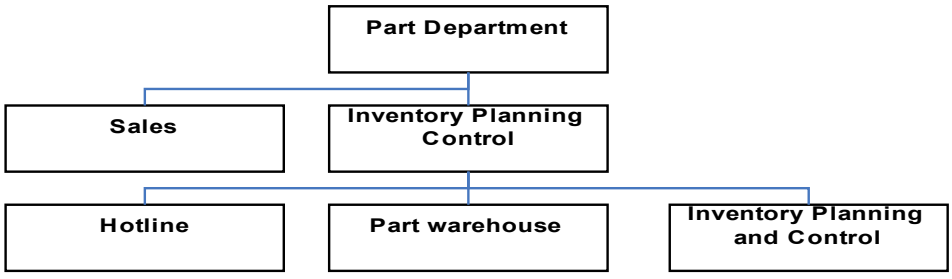
Struktur *Customer Care Center Department* PT. KBM seperti terlihat pada gambar II.6.



Gambar II.6 Struktur *Customer Care Center Department* PT. KBM

2.1.3.7 Struktur *Part Department*

Struktur *Part Department* PT. KBM seperti terlihat pada gambar II.7.



Gambar II.7 Struktur *Part Department* PT. KBM

2.1.4 Visi/Misi/Tujuan dan Strategi PT. KBM

PT. KBM mempunyai visi, misi, tujuan dan strategi demi kelancaran berbisnis dan meningkatkan prestasi pada perusahaan.

2.1.4.1 Visi PT. KBM

Menjadi pemimpin pasar sepeda motor YAMAHA melalui pelayanan terbaik dan dengan tata kelola perusahaan yang baik *(good corporate governance).*

2.1.4.2 Misi PT. KBM

Misi dari PT. KBM adalah sebagai berikut:

1) Mengembangkan sumber daya manusia secara konsisten agar senantiasa cakap, handal dan terpercaya dalam menjalankan kegiatan perusahaan.
2) Menjunjung tinggi integritas dan mau melayani lebih *(care)* dari seluruh karyawan.
3) Meyakini dan melaksanakan pelayanan terbaik bagi konsumen berikutnya sebagai keberhasilan perusahaan.
4) Memiliki sistem yang rapi dan terbuka sehingga tercipta tata kelola perusahaan.
5) Menghargai terobosan dan senantiasa menciptakan nilai tambah operasi perusahaan agar senantiasa terdepan dalam persaingan.
6) Efesiensi dan efektifitas bagian dari penentuan dan dasar kebijakan perusahaan.

2.1.4.3 *Goal and Strategy* PT. KBM

PT.KBM memiliki tujuan yang ingin dicapai dan strategi untuk mencapai tujuannya tersebut, strateginya terletak pada:

1) Penjualan Unit

Memberikan tantangan untuk penjualan mencapai target penjualan unit sepeda motor. Memberikan pengaruh yang agresif untuk aktivitas penjualan di persaingan tantangan teratas.

2) Market Share

Mempertahankan wilayah yang penguasaan pasarnya tinggi, dan fokus terhadap wilayah yang penguasaan pasarnya lemah.

3) Penjualan kredit

Mempertahankan bantuan penjualan kredit dimana sebagai strategi utama pemasaran. Memperkuat hubungan dengan setiap perusahaan pemberi kredit melalui pertemuan rutin/ pertemuan bisnis.

2.2 LANDASAN TEORI

Landasan teori menjelaskan beberapa teori yang berkaitan dengan permasalahan yang dibahas sebagai dasar pemahaman dalam sebuah sistem serta metode yang dipakai untuk kegiatan pengembangan terhadap sistem sendiri.

2.2.1 Sistem Informasi

Secara umum dalam melakukan proses mengubah data menjadi informasi membutuhkan sebuah sistem yang dinamakan sistem informasi.

2.2.1.1 Konsep Sistem Informasi

Sistem adalah sebuah tatanan (keterpaduan) yang terdiri atas sebuah komponen fungsional (dengan satu fungsi/tugas khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses/pekerjaan tertentu. [2].

Informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambian keputusan saat ini atau saat mendatang. [2].

Sistem informasi adalah suatu sistem buatan manusia yang secara umum terdiri atas sekumpulan komponen berbasis komputer dan manual yang dibuat untuk menghimpun, menyimpan, dan mengelola data serta menyediakan informasi keluaran kepada para pemakai. [2].

2.2.1.2 Aktivitas Sistem Informasi

Sistem informasi memiliki beberapa aktivitas utama, yaitu:

1. Melaksanakan komputasi numerik bervolume besar dengan kecepatan tinggi.
2. Menyediakan komunikasi dalam organisasi atau antar organisasi yang murah, akurat dan cepat.
3. Menyimpan informasi dalam jumlah yang besar dalam dalam ruang yang kecil tetapi mudah diakses.
4. Memungkinkan pengaksesan informasi yang sangat banyak di seluruh dunia dengan cepat dan murah.

5. Meningkatkan efektifitas dan efesiensi orang-orang yang bekerja dalam kelompok, dalam suatu tempat, atau pada beberapa lokasi.
6. Menyajikan informasi dengan jelas yang menggugah pikiran manusia.
7. Mengotomisasi proses-proses bisnis yang semi otomatis dan tugas-tugas yang dikerjakan secara manual.
8. Mempercepat pengetikan dan penyuntingan.
9. Pembiayaan yang jauh lebih murah dari pada pengerjaan secara manual.

Pada suatu sistem informasi terdapat komponen-komponen seperti perangkat keras *(hardware),* perangkat lunak *(software)* atau program, aturan atau prosedur, orang, basis data *(database),* serta jaringan komputer dan komunikasi data. Pada praktiknya, tidak semua sistem informasi mencakup keseluruhan komponen tersebut. Komponen-komponen yang terlibat tergantung pada bentuk dari sistem informasi yang digunakan, yaitu sederhana atau kompleks.

2.2.1.3 **Perancangan Sistem Informasi**

Perancangan sistem informasi merupakan pengembangan sistem baru dari sistem lama yang ada, dimana masalah-masalah yang terjadi pada sistem lama diharapkan sudah teratasi pada sistem yang baru. Siklus hidup pengembangan sistem informasi atau *System Development Life Cycles (SDLC)* secara konseptual dalam pengembangan sebuah sistem informasi adalah sebagai berikut :

a. Analisis sistem: menganalisis dan mendefinisikan masalah dan kemungkinan solusinya untuk sistem informasi dan proses organisasi.

b. Perancangan sistem: merancang *output, input*, struktur file, program, prosedur, perangkat keras dan perangkat lunak yang diperlukan untuk mendukung sistem informasi.
c. Pembangunan dan testing sistem: membangun perangkat lunak yang diperlukan untuk mendukung sistem dan melakukan testing secara akurat. Melakukan instalasi dan testing terhadap perangkat keras dan mengoperasikan perangkat lunak.
d. Implementasi sistem: beralih dari sistem lama ke sistem baru, melakukan pelatihan dan panduan seperlunya.
e. Operasi dan perawatan: mendukung operasi sistem informasi dan melakukan perubahan atau tambahan fasilitas.
f. Evaluasi sistem: mengevaluasi sejauh mana sistem telah dibangun dan seberapa bagus sistem telah dioperasikan.

2.2.2 Customer Relationship Management (CRM)

CRM adalah strategi bisnis dengan menerapkan teknologi informasi pada perusahaan secara komprehensif, handal, dan terintegrasi berdasarkan sudut pandang pelanggan, dengan begitu semua proses dan interaksi saling membantu mempertahankan dan mengembangkan hubungan yang bermanfaat. [4].

CRM merupakan suatu strategi bisnis yang diterapkan untuk mempelajari kebutuhan dan perilaku pelanggan agar perusahaan dapat mempertahankan pelanggan tersebut dengan cara membangun hubungan yang kuat antara perusahaan dengan pelanggan. *customer relationship management* (CRM)

bukanlah semata-mata teknologi, meskipun dalam pelaksanaannya harus didukung oleh serangkaian teknologi. CRM juga dapat didefinisikan sebagai sebuah proses yang menggabungkan berbagai potongan informasi tentang pelanggan, penjualan, efektifitas pemasaran, tingkat respon pasar, dan tren pasar.

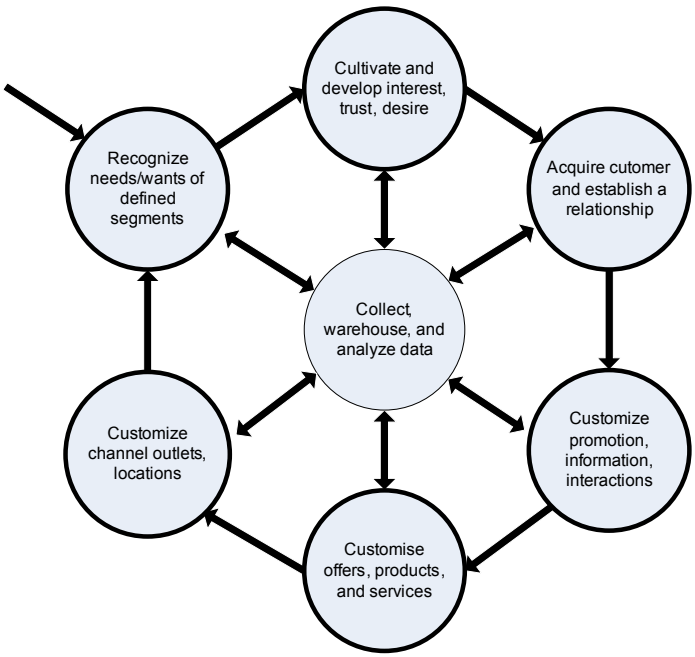
Ide CRM adalah untuk membantu pelaku bisnis menggunakan teknologi dan sumber daya manusia untuk mendapatkan informasi mengenai perilaku pelanggan dan nilai pelanggan tersebut. Manfaat yang diperoleh dari penerapan CRM yang baik adalah:

a. memberikan layanan konsumen yang lebih baik.
b. membuat *'call center'* menjadi lebih efisien.
c. membuat proses *'cross sell'* produk menjadi lebih baik.
d. membantu staf penjualan bernegosiasi lebih cepat.
e. menyederhanakan proses pemasaran dan penjualan.
f. mengidentifikasi peluang konsumen baru.
g. meningkatkan pendapatan dari konsumen.

Tujuan dari sistem CRM adalah meningkatkan pelayanan pada pelanggan, menambah kepuasan pelanggan, dan memastikan ingatan pelanggan dengan meluruskan cara berbisnis dengan integrasi teknologi. Selain itu, tujuan utama dari CRM adalah mengidentifikasi, mengantisipasi, mempertahankan, memahami kebutuhan pelanggan baik pelanggan saat ini maupun pihak yang berpotensi untuk menjadi pelanggan dengan cara mengumpulkan informasi tentang pelanggan, penjualan dan keefektivan pemasaran dan tren pasar, supaya CRM dapat berhasil dengan baik ialah organisasi harus mengetahui secara tepat jenis informasi apa

saja yang diinginkan dan didapatkan pelanggannya dan apa yang akan dilakukan dengan informasi yang diketahui tersebut.

Prinsip *customer relationship management* (CRM) diantaranya memberdayakan hubungan yang telah terbentuk untuk memaksimalkan pendapatan, memberdayakan pengetahuan tentang pelanggan untuk memberikan pelayanan yang terbaik, membentuk proses penjualan yang dapat dilakukan berulang-ulang, dan membangun loyalitas pelanggan. Menerapkan strategi CRM berarti menigkatkan pelayanan kepada pelanggan dan menciptakan interaksi yang konsisten, dapat diandalkan dan mudah diakses di setiap interaksi antara pelanggan dan perusahaan. Sasaran yang ingin dicapai dari proses sistem CRM adalah optimalisasi dari keuntungan, dan hal ini dapat dicapai dengan memahami bahwa tidak semua pelanggan diciptakan serupa satu sama lain. Hanya perusahaan yang mamahami hal ini dan memahami bagaimana keinginan tiap pelanggan dalam berinteraksi dengan perusahaan, akan mampu meraih kesetiaan pelanggan dan kemajuan proses bisnis yang menguntungkan. Proses pada CRM seperti terlihat pada gambar II.8.



(Sumber: Zikmund, McLeod dan Gilbert, 2003)

Gambar II.8 Proses pada CRM

2.2.2.1 Tahapan dalam CRM

Pengimplentasian konsep CRM yang baik adalah mengorganisasikan proses CRM di sekitar pelanggan dan tidak hanya pada fungsi internal perusahaan. [4]. Ada tiga tahapan dalam CRM, yaitu:

a. Acquire

Acquire merupakan suatu tahap mengakuisisi pelanggan. Melalui tahap ini dapat memberikan sebuah penilaian kepada perusahaan dengan menilai kesan pertama terhadap perusahaan. Apabila pelanggan memiliki penilaian yang baik terhadap perusahaan, maka transaksi akan dapat berjalan dengan baik. Mengakuisisi pelanggan baru dengan melakukan promosi atas produk dan jasa yang ditawarkan.

b. Enhance

Enhance merupakan suatu proses dimana perusahaan membangun hubungan yang baik dengan pelanggan. Dengan mendengarkan keluhan dari pelanggan dan menerima saran dari pelanggan merupakan salah satu cara yang efektif dalam membangun hubungan dengan pelanggan. Dalam tahap ini ada dua hal yang harus diperhatikan:

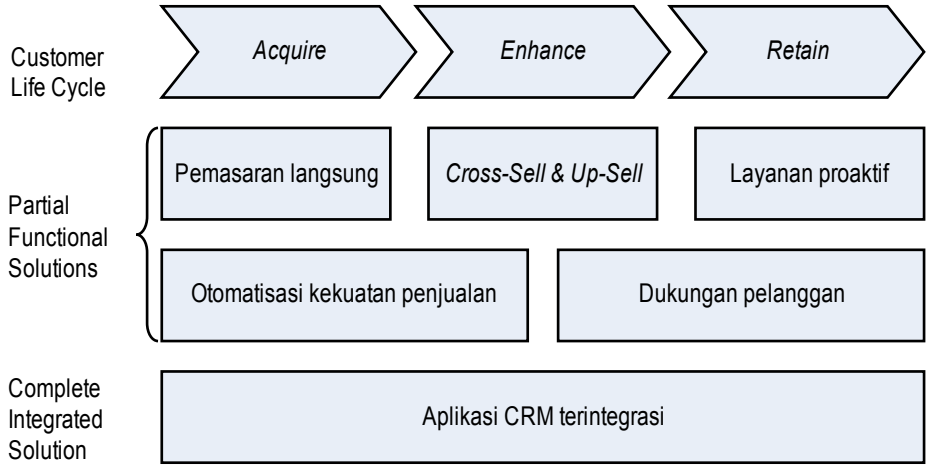
1) *Cross selling*, yaitu merupakan salah satu strategi dalam mengidentifikasi pelanggan guna manambah penjualan. Misalnya, menawarkan produk pelengkap kepada pelanggan.
2) *Up Selling*, yaitu merupakan strategi meningkatkan penjualan produk melalui penambahan kuantitas produk. Misalnya, menawarkan produk yang sama dengan kuantitas yang lebih besar.

c. Retain

Retain merupakan tahap dimana perusahaan berusaha mampertahankan hubungan dengan pelanggan, dengan kata lain mendapatkan pelanggan dan kemudian memelihara hubungan dengan mereka. Agar dapat menjalin hubungan dengan terus-menerus dan mempertahankan hubungan denga pelanggan, maka perusahaan harus menggunakan pandangan yang menggunakan pelanggan dengan memberikan pelayanan dan aplikasi pendukung.

Perusahaan yang ideal untuk melakukan CRM adalah perusahaan yang ingin mengoptimalkan keuntungan dengan cara mengorganisasikan proses CRM

di sekitar pelanggan dan tidak hanya pada fungsi internal perusahaan agar dapat menjalin hubungan lebih dekat dan harmonis dengan pelanggannya. Dengan demikian nilai tambah CRM dapat dioptimalkan dalam mengefektifkan dan mengefesiensikan biaya untuk mengakuisisi pelanggan baru dan mempertahankan pelanggan lama sehingga biaya dan usaha pemasaran tidak sia-sia. Skemanya seperti terlihat pada gambar II.9.



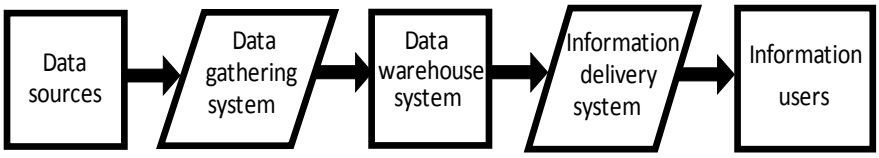
(Sumber: Zikmund, McLeod dan Gilbert, 2003)

Gambar II.9 Skema solusi dalam CRM

2.2.2.2 Arsitektur CRM

Pada arsitektur CRM, *data warehouse* adalah elemen pusat dalam sistem CRM, *data sources*, keduanya berada di dalam dan di luar perusahaan, menyediakan uraian data hubungan dengan pelanggan. Pada *data gathering system* mengubah data yang masuk ke alat elektronik jika belum siap diformat. *Data warehouse system* mempersiapkan untuk penyimpanan, menyimpan data, uraikan data sehingga kemudian data bisa didapatkan kembali, dan melakukan pengaturan dan berfungsi mengontrol data dalam data *warehouse system*. *Information delivery system* membuat isi dari *data warehouse* tersedia untuk

information users dalam bentuk elektronik atau dicetak sebagai laporan dan perlihatkan. Untuk bentuk arsitekturnya terlihat pada gambar II.10



(Sumber: Zikmund, McLeod dan Gilbert,2003)

Gambar II.10 *A Basic CRM Model*

Arsitektur CRM memudahkan pertemuan data, menyimpan data, mengubah data menjadi informasi dan memberikan informasi kepada pengguna. Oleh karena itu, sistem mengubah data menjadi informasi. Data terdiri dari fakta dan figur-figur yang bersifat sulit digunakan karena jumlahnya. Informasi terdiri atas kumpulan dan ringkasan berarti dari data yang memberitahukan kepada pengguna sesuatu yang telah pengguna tidak ketahui. [4].

2.2.2.3 Jenis CRM

Jenis dari *Customer Relationship Management (CRM)* adalah *Operational CRM, Analytical CRM dan Collaborative CRM / Full Integrated CRM.*

a. ***Operational CRM***

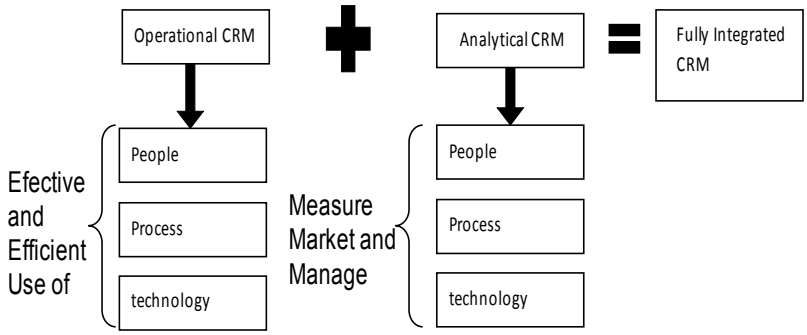
Pengelolaan secara otomatisasi dari proses bisnis secara integrasi dan horizontal.

b. Analytical CRM

Analisis data yang diperoleh dari operasional CRM dengan memanfaatkan *tools* dan *software* untuk mendapatkan pemahaman yang lebih baik dari pelanggan.

c. ***Collaborative CRM / Full Integrated CRM***

Seperangkat aplikasi dari pelayanan kolaborasi yang dirancang untuk memfasilitasi antara pelanggan dengan perusahaan.



(Sumber: Anton dan Pettouhoff, 2002)

Gambar II.11 *Fully Integrated CRM*

Dari gambar di atas hubungan antara manusia, proses dan teknologi dapat diartikan sebagai berikut :

1) Manusia merupakan faktor utama, CRM sebenarnya adalah bagaimana mengelola hubungan antar manusia sehingga diperlukan sentuhan dari manusia.
2) Proses merupakan sistem atau prosedur yang membantu manusia untuk lebih mengenali dan menjalin hubungan secara dekat dengan pelanggan.

3) Teknologi dilakukan setelah manusia dan proses telah dipersiapkan dengan baik, maka langkah selanjutnya adalah mempersiapkan teknologi agar dapat menunjang manusia dalam melaksanakan proses.

2.2.3 Perawatan Kendaraan (*Service Vehicle*)

Perawatan adalah kegiatan yang dilakukan dalam bentuk pemeliharaan terhadap suatu barang. Kendaraan adalah benda yang dipergunakan manusia untuk berbagai kepentingan. Misalnya, kendaraan bermotor roda dua/roda empat, sepeda dan lain-lain. Perawatan kendaraan dapat diartikan sebagai kegiatan yang dilakukan dalam bentuk pemeliharaan terhadap sebuah kendaraan yang dilakukan oleh manusia dengan tujuan untuk menjadikan kendaraan tetap dalam keadaan baik.

2.2.4 Basis Data

Basis data adalah kumpulan *file-file* yang mempunyai kaitan antara satu file dengan file lain sehingga membentuk satu bangunan data untuk menginformasikan satu perusahaan, instansi dalam batasan tertentu. [2]. Basis data relasional adalah basis data yang mengorganisasikan data dalam bentuk tabel. Tabel dibentuk dengan mengelompokan data yang mempunyai subjek yang sama.

Basis data pada prinsipnya ditujukan untuk pengaturan data agar terdapat kemudahan dalam pengambilan kembali data tersebut. Berikut ini terdapat beberapa tujuan dari basis data diantaranya yaitu:

1. kecepatan dan kemudahan *(speed)*.
2. efesiensi ruang penyimpanan *(space)*.
3. keakuratan *(accuracy)*.
4. ketersediaan *(avaibility)*.
5. kelengkapan *(completeness)*.
6. keamanan *(security)*.
7. kebersamaan *(sharability)*.

Kumpulan file yang saling berkaitan bersama dengan program untuk pengelolaannya disebut sebagai DBMS *(Database Management System)*. *Database* adalah kumpulan datanya, sedangkan program pengelolaannya berdiri sendiri dalam satu paket program yang komersial untuk membaca data, mengisi data, menghapus data, melaporkan data dalam *database*.

Beberapa definisi dalam perencanaan *database* yang perlu untuk diketahui adalah sebagai berikut:

a	*Entity*	:	Orang, tempat kejadian, kosep yang informasinya direkam.
b	*Attribute*	:	Setiap *entity* mempunyai atribut untuk mewakili satu *entity*/ atribut disebut juga sebagai data elemen, data field atau data item.
c	*Data Value*	:	Data aktual atau informasi yang disimpan pada tiap data elemen.
d	*Record/tufle*	:	Kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang *entity* secara lengkap.
e	*File*	:	Kumpulan *record-record* sejenis yang mempunyai panjang elemen yang sama, atribut yang sama namun berbeda *data value*nya.

2.2.5 *SQL (Structured Query Language)*

SQL (Structured Query Language) merupakan standar bahasa yang digunakan untuk memanipulasi/*Data Manipulation Language (DML)* dan mendefinisikan/*Data Definition Language (DDL)* objek-objek basis data sesuai dengan permintaan *(request)* dari pengguna sistem. Dimana *DML (Data Manipulatoin Language)* adalah sekumpulan perintah-perintah *query* yang digunakan untuk melakukan manipulasi data seperti penyimpanan data ke suatu tabel, lalu kemudian mengubahnya atau menghapusnya atau hanya sekedar menampilkan saja, sedang *DDL (Data Definition Language)* merujuk pada kumpulan perintah *query* yang digunakan untuk mendefinisikan objek-objek basis data seperti pembuatan tabel basis data atau indeks primer/sekunder dan lain-lain.

2.2.6 **Prosedur Tersimpan *(Stored Procedure)***

Prosedur tersimpan *(store procedure)* adalah perintah yang diberi nama dan disimpan atau tersimpan di basis data *server*. *Store procedure* biasanya berisi perintah-perintah umum yang berhubungan dengan basis data pada *server* dan menghasilkan sekumpulan hasil *(record). Store procedure* ditulis dalam bentuk *script.*

2.2.7 Arsitektur Sistem Basis Data

Arsitektur sistem merujuk pada konfigurasi sistem secara keseluruhan yang akan menjadi tempat hidup dari DBMS basis data dan aplikasi yang memanfaatkannya. Bagaimana tempat wujud dari tempat hidup tersebut yang juga

akan menentukan bagaimana para pemakai sistem berinteraksi dengannya, sudah seharusnya ditetapkan sejak memulai perancangan basis data atau paling tidak sebelum melaksanakan tahap implementasi basis data.

Dari waktu ke waktu, sejalan dengan kemajuan teknologi, arsitektur yang dapat kita pilih semakin beragam atau semakin banyak variannya, yang berubah juga keunggulannya pada tiap jenis arsitektur tersebut. Arsitektur saat ini yang dikatakan paling baik dan banyak ditempatkan di berbagai tempat dan di masa yang akan datang dapat beralih ke arsitektur lain yang memang sama sekali baru atau merupakan pengembangan dari arsitektur yang lama, yang menjadi utama adalah menentukan arsitektur yang cocok yang tentu saja bukan hanya keunggulan teknologinya tetapi disamping itu biaya yang sesuai dengan anggaran organisasi yang akan mengaplikasikannya.

2.2.8.1 Sistem Tunggal

Pada arsitektur ini *Database Management System (DBMS),* basis data dan aplikasi basis data ditempatkan pada mesin (komputer) yang sama. Dengan demikian, pemakai yang dapat menggunakannya setiap saat juga hanya satu orang *(single user).*

2.2.7.2 Sistem Tersentralisasi

Arsitektur ini terdiri dari sebuah mesin *server* dan sejumlah terminal (yang menjadi tempat *user* berinteraksi dengan sistem), yang tersentralisasi dalam

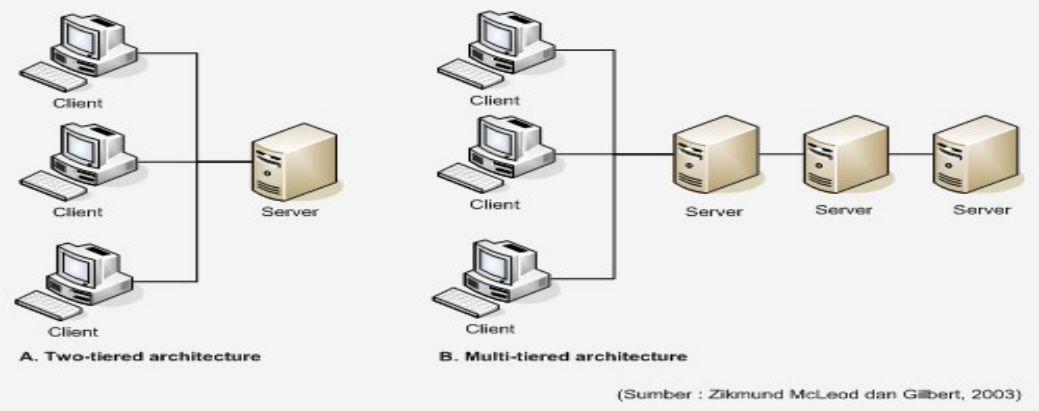
arsitektur ini dapat mencakup basis data, DBMS, dan aplikasi basis data atau basis data saja.

2.2.7.3 Sistem *Client-Server*

Arsitektur *client server* dapat dikatakan arsitektur yang paling baik dari paling banyak dipakai di berbagai tempat. Pada arsitektur ini ada dua komponen utama yaitu *client* merupakan terminal yang menjalankan *(request)* basis data dan *server* merupakan terminal memberikan pelayanan *(response)* terhadap permintaan *(request)* yang dilakukan oleh terminal *client* atas proses tertentu. Dimana terletak *client* berisi aplikasi basis data dan *server* berisi DBMS dan basis data.

Setiap aktivitas yang dikehendaki oleh pemakai akan lebih dulu ditangani oleh *client*. *Client* selanjutnya mengupayakan agar semua proses sebisa mungkin ditangani sendiri. Jika ada proses yang harus melibatkan data yang tersimpan pada basis data yang tersimpan di *server* seperti mengakses *stored procedure* atau lainya, barulah *client* melakukan kontak dengan s*erver*. Contohnya ada kebutuhan untuk mendapatkan informasi yang dikirim yang diwakili oleh baris-baris *(row)* dengan kriteria-kriteria tertentu *(*proses *query, stored procedure)* dari tabel yang diinginkan.

Untuk lebih jelasnya, arsitektur dari sistem *client-server*, seperti pada gambar II.12.



Gambar II.12 Arsitektur *Client-Server*

2.2.8 Protokol

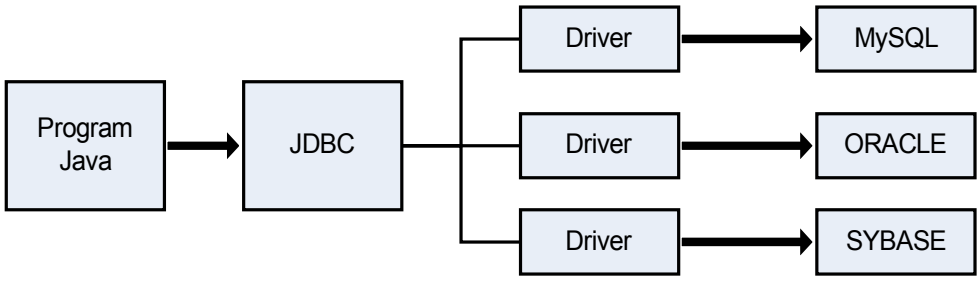
Protokol adalah perangkat lunak yang memberikan satu kesamaan bahasa agar antar terminal dapat berkomunikasi atau saling ada pengertian antara satu terminal dengan terminal yang lain. Protokol yang dipergunakan dalam aplikasi sistem adalah protokol TCP/IP *(Transmision Control Protocol/ internet protocol)* dimana protokol ini menjadi standar protokol karena beberapa *platform* lainnya mempunyai protocol ini. Protokol TCP/IP menggunakan alamat komputer dalam pengalamatan komputer yang disebut *IP Address* agar dapat diketahui komputer pengirim atau komputer yang akan dituju.

2.2.9 *Java Database Connectivity (JDBC)*

Java Database Connectivity (JDBC) adalah *Application Programming Interface (API)* yang dikembangkan oleh *Sun Microsystem* untuk menyediakan akses data universal dalam bahasa pemrograman JAVA. JDBC merupakan bagian dari Java 2 (dua) SDK *Standard Edition*. Inti JDBC adalah *package.sql* dan

javax.sql (bagian dari J2EE). JDBC menyediakan mekanisme untuk koneksi *database*, sintaks untuk mengirimkan *query* dan melakukan transaksi serta struktur data yang mewakili hasil *query*.

Diagram JDBC seperti terlihat pada Gambar II.13 di bawah ini.



Gambar II.13 Diagram JDBC

JDBC merupakan koleksi dari API yang terdiri dari sekumpulan *class* dan *interface* yang ditulis dalam pemrograman Java sebagai *standart API* untuk *developer database* maupun *software database* sehingga memungkinkan pembuatan aplikasi *database* yang *portable* dengan bahasa Java (Rikyanto, 2002).

Dengan JDBC kita dapat membuat program aplikasi *database* yang dapat berganti *server* tanpa mengubah banyak kode dari program yang sudah ada sehingga memberikan portabilitas tinggi bagi aplikasi yang telah dibuat.

Dalam JDBC terdapat tujuh langkah standar untuk menangani *database*:

1) Memanggil/*load driver JDBC*.
2) Mendefinisikan koneksi *database*.
3) Melakukan koneksi *database*.
4) Membuat objek *statement*.
5) Melakukan *query dan update*.
6) Memproses hasil.

7) Menutup koneksi.

2.2.10 UML

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis *object-oriented*. UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem *software*. [6].

Untuk dapat memahami UML membutuhkan bentuk konsep dari sebuah bahasa model, dan mempelajari 3 (tiga) elemen utama dari UML seperti *building block*, aturan-aturan yang menyatakan bagaimana *building block* diletakkan secara bersamaan, dan beberapa mekanisme umum (*common*). Tiga macam yang terdapat dalam *building block* adalah katagori benda/*things*, hubungan, dan diagram. Benda/*things* adalah abstraksi yang pertama dalam sebuah model, hubungan sebagai alat komunikasi dari benda-benda, dan diagram sebagai kumpulan / group dari benda-benda/*things.*

1) Benda/*Things*

Benda adalah hal yang sangat mendasar dalam model UML, juga merupakan bagian paling statik dari sebuah model, serta menjelaskan elemen-elemen lainnya dari sebuah konsep dan atau fisik. Bentuk dari beberapa benda/*thing*s adalah sebagai berikut:

Pertama, adalah sebuah kelas yang diuraikan sebagai sekelompok dari *object* yang mempunyai atribut, operasi, hubungan yang semantik. Sebuah kelas mengimplementasikan satu atau lebih *interfaces*. Sebuah kelas dapat digambarkan sebagai sebuah persegi panjang, yang mempunyai sebuah nama, atribut, dan metoda pengoperasiannya.

Kedua, yang menggambarkan *'interface'* merupakan sebuah antar-muka yang menghubungkan dan melayani antar kelas dan atau elemen. *'Interface'*/antar-muka mendefinisikan sebuah set/kelompok dari spesifikasi pengoperasian, umumnya digambarkan dengan sebuah lingkaran yang disertai dengan namanya. Sebuah antar-muka berdiri sendiri dan umumnya merupakan pelengkap dari kelas atau komponen.

Ketiga, adalah *collaboration* yang didefinisikan dengan interaksi dan sebuah kumpulan/kelompok dari kelas-kelas/elemen-elemen yang bekerja secara bersama-sama. C*ollaborations* mempunyai struktur dan dimensi. Pemberian sebuah kelas memungkinkan berpartisipasi didalam beberapa *collaborations* dan digambarkan dengan sebuah '*elips*' dengan garis terpotong-potong.

Keempat, sebuah '*use case*' adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. '*use case*' digunakan untuk membentuk tingkah-laku benda/*things* dalam sebuah model serta direalisasikan oleh sebuah collaboration. Umumnya *'use case'* digambarkan dengan sebuah *'elips'* dengan garis yang solid, biasanya mengandung nama.

Kelima, sebuah node merupakan fisik dari elemen-elemen yang ada pada saat dijalankannya sebuah sistem, contohnya adalah sebuah komputer, umumnya mempunyai sedikitnya *memory* dan *processor*. Sekelompok komponen mungkin terletak pada sebuah *node* dan juga mungkin akan berpindah dari *node* satu ke *node* lainnya. Umumnya *node* ini digambarkan seperti kubus serta hanya mengandung namanya.

2) Hubungan / *Relationship*

Terdapat empat macam hubungan didalam penggunaan UML, yaitu; *dependency, association, generalization*, dan *realization.*

Pertama, sebuah *dependency* adalah hubungan semantik antara dua benda/things yang mana sebuah benda berubah mengakibatkan benda satunya akan berubah pula. Umumnya sebuah *dependency* digambarkan sebuah panah dengan garis terputus-putus.

Kedua, sebuah *association* adalah hubungan antar benda struktural yang terhubung diantara obyek. Kesatuan obyek yang terhubung merupakan hubungan khusus, yang menggambarkan sebuah hubungan struktural diantara seluruh atau sebagian. Umumnya *assosiation* digambarkan dengan sebuah garis yang dilengkapi dengan sebuah label, nama, dan status hubungannya.

Ketiga, sebuah *generalization* adalah menggambarkan hubungan khusus dalam obyek anak/*child* yang menggantikan obyek *parent* / induk . Dalam hal ini, obyek anak memberikan pengaruhnya dalam hal struktur dan tingkah lakunya kepada obyek induk.

Keempat, sebuah *realization* merupakan hubungan semantik antara pengelompokkan yang menjamin adanya ikatan diantaranya. Hubungan ini dapat diwujudkan diantara *interface* dan kelas atau *elements*, serta antara *use cases* dan *collaborations*.

3) Diagram

UML terdiri atas pengelompokkan diagram-diagram sistem menurut aspek atau sudut pandang tertentu. Diagram adalah yang menggambarkan permasalahan maupun solusi dari permasalahan suatu model.

Diagram-diagram dari UML adalah *use-case, class, object, state, sequence, collaboration, activity, component,* dan *deployment diagram.*

1) Use Case Diagram

Use case adalah suatu diagram yang membantu pengembangan sistem bekerja dengan user untuk menentukan kegunaan sistem. Koleksi dari *use case* melukiskan apa yang diinginkan *user* terhadap sebuah sistem. *Use case* bertujuan untuk menentukan bagaimana *actors* berinteraksi dengan sebuah sistem. [6].

a. *Actor*

Proses pembuatan *Use case* diawali dengan mengidentifikasi aktor pada sistem. Aktor adalah segala sesuatu yang berhubungan dengan sistem sebagai pelaksana atau pemucu timbulnya *use case*. Aktor dapat berupa manusia, perangkat lunak, perangkat keras *data store*, ataupun jaringan. Setiap aktor mempunyai peranan tertentu.

b. Precondition

Precondition adalah sebuah kondisi awal yang harus dimiliki aktor untuk dapat masuk kedalam sistem. *Precondition* memberitahukan bahwa sistem harus berada dalam kondisi apa pada awal *use case*.

c. Postcondition

Postcondition adalah sebuah kondisi akhir dari proses *use case* yang telah dilakukan atau hasil akhir yang diterima oleh *actor. Postcondition* memberitahukan bahwa sistem harus berada dalam kondisi apa pada akhir *use case*.

d. Flow of event

Flow of event adalah sebuah skenario yang ditulis dengan anggapan bahwa segala sesuatunya berjalan dengan lancar pada sebuah proses. Skenario ditulis dari sudut pandang aktor. Setiap skenario merupakan serangkaian kejadian yang mendeskripsikan kegunaan dari *use case*.

e. Alternative flow

Alternative flow adalah sebuah skenario yang memberikan serangkaian kejadian yang berbeda dengan yang digunakan pada *flow of event*. Di sini, seorang aktor mungkin memilih salah satu dari beberapa hal yang dapat dilakukan pada point yang sama dalam *use case*.

2) Statechart diagram

Statechart diagram adalah bagian dari UML yang menggambarkan tingkah laku yang umum dari sebuah objek didalam sebuah class yang spesifik dan berisi *states* dan transisi diantaranya. [6].

Statechart sangat penting karena dapat membantu analisis sistem, *designer* dan pengembangan dalam memahami *behavior* dari objek dalam suatu sistem.

3) Activity Diagram

Activity diagram adalah bagian daru UML yang digunakan untuk menggambarkan tahapan dari setiap proses bisnis yang ada agar lebih mudah memahami proses bisinis yang terjadi. [6].

Dalam *activity diagram* tiap aktivitas direpresentasikan dengan *rounded rectangle* yang dihubungkan dengan anak panah untuk menggambarkan transisi dari satu aktivitas ke aktivitas lain. *Activity diagram* dimulai dari *initial state* dan diakhiri dengan *final state*.

4) Class Diagram

Class diagram adalah bagian dari UML yang menggambarkan sebuah kumpulan dari kelas-kelas yang ada dan hubungan diantara kelas tersebut dimana setiap kelas mempunyai *attributes* dan *operations*. [6].

Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Selain itu, c*lass diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

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 atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons 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*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

6) Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek 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. *Messages* dari level yang sama memiliki prefiks yang sama.

7) Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. 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 komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. 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 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 menyebarkan komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

9) Object Diagram

Object diagram adalah diagram yang memberikan gambaran model *instance-instance* dari sebuah *class*. Diagram ini digunakan untuk menggambarkan sebuah sistem pada sebuah sudut pandang waktu tertentu. Dengan menggunakan diagram ini anda dapat memeriksa keabsahan kelas-kelas diagram berikut aturan-aturan multiplisitasnya dengan "*real data"* dan mengujinya dengan skenario-skenario tertentu.