# BAB II

# TINJAUAN PUSTAKA

Tinjauan Pustaka memiliki dua sub bab, yaitu tinjauan perusahaan yang merupakan tempat penelitian aplikasi ini dibangun, dan landasan teori yang berkaitan dengan perangkat lunak untuk membangun suatu aplikasi.

## 2.1 Tinjauan Perusahaan

Tinjauan perusahaan ini adalah sejarah tentang perusahaan, visi dan misi, dan struktur organisasi tempat dilakukannya penelitian aplikasi yang akan dibangun.

### 2.1.1 Sejarah Perusahaan

Dalam perjalanan sejarahnya, CV.Trendy House dibangun bersama-sama oleh 2 (dua) orang pimpinan yang dulu namanya CV.Trendy Digital Photo yang bergerak dibidang *digital photo*. Namun dengan berjalannya waktu dan dinamika yang harus diadaptasi, seperti perkembangan teknologi, regulasi, dan pasar, maka CV.Trendy House telah mengalami berbagai perubahan dan perkembangannya. CV.Trendy House yang sekarang ini tidak bergerak dibidang *digital photo* saja, tetapi sudah dibangunnya dibidang industri pakaian dan dibidang *counter* pulsa. CV.Trendy House yang terletak di Komplek Perumdam Lebong Siarang Blok.F9 Palembang mulai berdiri pada tanggal 15 April 2007.

Dilihat dari prospek masa depannya CV.Trendy House merupakan salah satu untuk memenuhi kebutuhan masyarakat untuk keperluan bahan jadi. Untuk

pembelian barang ke supplier biasanya perusahaan membeli barang dengan jumlah yang lebih besar agar pengeluaran perusahaan sedikit. Sedangkan penjualan barang ke konsumen, perusahaan memberikan potongan harga/diskon jika pembeliannya dalam jumlah yang lebih besar yang telah ditentukan oleh perusahaan.

### 2.1.2 Visi dan Misi Perusahaan

#### 2.1.2.1 Visi

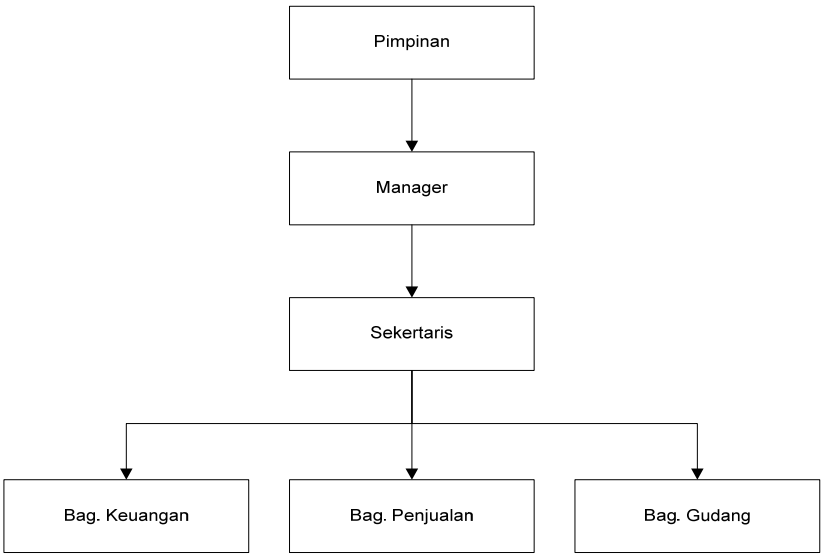
Visi perusahaan CV.Trendy House adalah menjadikan unit pelayanan industri yang ramah lingkungan dan menjadi pilihan utama bagi masyarakat.

#### 2.1.2.2 Misi

Sedangkan misi perusahaan CV.Trendy House itu sendiri adalah sebagai penggerak utama dibidang industri dan sebagai komponen kegiatan pembangunan yang berkelanjutan dan berwawasan lingkungan.

### 2.1.3 Struktur Organisasi

Struktur organisasi CV.Trendy House Palembang, dapat dilihat pada gambar berikut ini :




Gambar 2.1 Struktur Organisasi CV.Trendy House Palembang

Deskripsi pekerjaan :

1. Pimpinan, bertugas mengawasi kinerja seluruh karyawan, berkewajiban mengambil keputusan, serta bertanggung jawab terhadap segala kegiatan di perusahaan.
2. Manager, bertugas mengarahkan dan membina kinerja seluruh karyawan, serta penanggung jawab atas pemasukan dan pengeluaran kebutuhan *inventory* perusahaan.
3. Sekertaris, bertugas membuat dan mencatat surat keluar masuknya surat dan mendokumentasikan data-data perusahaan.
4. Bagian Keuangan, bertugas mengatur, mengamankan dan bertanggung jawab atas keuangan kas perusahaan beserta surat-suratnya.

5. Bagian Penjualan, bertugas melakukan transaksi peenjualan, mengatur perkembangan perusahaan untuk kemajuan perusahaan dan mencari alternatif-alternatif baru dalam perusahaan untuk kepentingan perusahaan.
6. Bagian Gudang, bertugas mengatur dan mencatat pengeluaran dan pemasukan suatu barang dari gudang ke gudang.

## 2.2 Landasan Teori

### 2.2.1 Pengertian Sistem

Sistem dapat didefinisikan menurut dua kelompok pendekatan sistem, yaitu yang melakukan pada prosedurnya dan yang menekankan pada komponen atau elemen. Definisi sistem berdasarkan pendekatan yang menekankan pada prosedur adalah sebagai berikut :

> “Suatu sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu”.[1]

Pendekatan sistem yang lebih menekankan pada elemen atau komponennya mendefinisikan sistem sebagai berikut :

> “Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai tujuan tertentu”.[1]

#### 2.2.1.1 Karakteristik Sistem

Suatu sistem memiliki karakteristik atau sifat-sifat tertentu, yaitu memiliki komponen-komponen *(components)*, batas sistem *(boundary)*, lingkungan luar sistem *(environment)*, penghubung *(interprest)*, masukan *(input)*, keluaran *(output)*, pengolah *(process)* dan sasaran *(objective)* dan tujuan *(goal)*.[1]

1. Komponen Sistem (*System Components*)

   Komponen sistem atau elemen-elemen sistem dapat berupa suatu sub sistem atau bagian-bagian dari sistem. Setiap sistem baik besar maupun kecil, selalu mengandung komponen-komponen atau subsistem-subsistem. Setiap subsistem mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai suatu yang lebih besar yang disebut *supra system.*

2. Batas Sistem (*System Boundary*)

   Batas sistem merupakan daerah-daerah yang membatasi antara satu sistem dengan sistem lainnya dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan. Batas suatu sistem menujukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem (*System Environment*)

   Lingkungan luar sistem dari suatu sistem adalah apapun di luar batas dari sistem yang mempengaruhi operasi sistem yang dapat bersifat menguntungkan dan dapat pula merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi dari sistem yang harus dijaga dan

dipelihara. Sedangkan yang merugikan harus ditahan dan dikendalikan, karena akan mengganggu kelangsungan hidup sistem.

4. Penghubung Sistem *(System Interprest)*

   Penghubung sistem merupakan media penghubung antara satu sub sistem dengan subsistem yang lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem yang lainnya. Dengan penghubung akan terjadi interaksi antar subsistem, sehingga membentuk satu kesatuan.

5. Masukan Sistem *(System Input)*

   Masukan adalah suatu energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). *Maintenance input* adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Contoh *maintenance input* di dalam sistem komputer adalah program, yang digunakan untuk mengoperasikan komputer. Sedangkan *signal input* adalah energi yang diproses untuk mendapatkan keluaran. Contoh *signal input* di dalam sistem komputer adalah data, yang dapat diolah menjadi informasi.

6. Keluaran Sistem *(System Output)*

   Keluaran (*Output*) merupakan hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna.

7. Pengolah Sistem (*System Process*)

   Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya, yang bertugas untuk merubah masukan menjadi keluaran.

8. Sasaran Sistem (*System Objective*)

   Suatu sistem pasti memiliki tujuan (*goal*) atau sasaran (*objective*). Suatu operasi sistem akan berguna dan berhasil apabila mencapai sasaran atau tujuannya. Sasaran sistem sangat menentukan masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem.

#### 2.2.1.2 Klasifikasi Sistem

Sistem dapat diklasifikasikan dari beberapa sudut pandang, diantaranya sebagai berikut ini :[1]

1. Sistem diklasifikasikan sebagai sistem abstrak (*abstract system*) dan sistem fisik (*physical system*).

   Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik. Misalnya sistem teologia, yaitu sistem yang berupa pemikiran-pemikiran hubungan antara manusia dengan Tuhan. Sistem fisik merupakan sistem yang ada secara fisik. Misalnya sistem komputer, sistem akuntansi, sistem produksi dan lain sebagainya.

2. Sistem diklasifikasikan sebagai sistem alamiah (*natural system*) dan system buatan manusia (*human made system*).

   Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat manusia. Misalnya sistem perputaran bumi. Sistem buatan manusia adalah sistem yang dirancang oleh manusia. Sistem buatan manusia yang melibatkan interaksi antara manusia dengan mesin disebut dengan *human-machine system* atau ada yang menyebut dengan *man-machine system*. Sistem informasi merupakan contoh *man-machine system*, karena menyangkut penggunaan komputer yang berinteraksi dengan manusia.

3. Sistem diklasifikasikan sebagai sistem tertentu (*deterministic system*) dan sistem tak tentu (*probabilistic system*).

   Sistem tertentu beroperasi dengan tingkah laku yang sudah dapat diprediksi. Interaksi diantara bagian-bagiannya dapat dideteksi dengan pasti, sehingga keluaran dari sistem dapat diramalkan. Sistem komputer adalah contoh dari sistem tertentu yang tingkah lakunya dapat dipastikan berdasarkan program-program yang dijalankan. Sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

4. Sistem diklasifikasikan sebagai sistem tertutup (*closed system*) dan sistem terbuka (*open system*).

   Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa adanya turut campur tangan dari pihak diluarnya. Secara teoritis sistem

tertutup ini ada, tetapi kenyataannya tidak ada sistem yang benar-benar tertutup, yang ada hanyalah *relatively closed system* (secara relatif tertutup, tidak benar-benar tertutup). Sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk lingkungan luar atau subsistem yang lainnya. Karena sistem sifatnya terbuka dan terpengaruh oleh lingkungan luarnya, maka suatu sistem harus mempunyai suatu sistem pengendalian yang baik. Sistem yang baik harus dirancang sedemikian rupa, sehingga secara relatif tertutup karena sistem tertutup akan bekerja secara otomatis dan terbuka hanya untuk pengaruh yang baik saja.

### 2.2.2 Pengertian Informasi

Informasi merupakan hasil pengolahan dari sebuah model, formasi, organisasi, ataupun suatu perubahan bentuk dari data yang memiliki nilai tertentu, dan bisa digunakan untuk menambah pengetahuan bagi yang menerimanya. Dalam hal ini, data bisa dianggap sebagai obyek dan informasi adalah suatu subyek yang bermanfaat bagi penerimanya. Informasi juga bisa disebut sebagai hasil pengolahan atau pemrosesan data. [1]

Sumber dari informasi adalah data. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata, tetapi data merupakan bentuk yang masih mentah dan belum dapat memberikan arti banyak bagi pemakai,

sehingga perlu diolah lebih lanjut untuk menghasilkan informasi yang dibutuhkan. Dengan kata lain informasi adalah hasil dari pengolahan data.

#### 2.2.2.1 Kualitas Informasi

Kualitas informasi sangat dipengaruhi atau ditentukan 3 hal, yaitu : [1]

a. Relevan (*relevancy*)

   Relevan berarti informasi harus memberikan manfaat bagi pemakainya. Relevansi informasi berbeda-beda untuk tiap-tiap orang satu dengan yang lainnya.

b. Akurat (*accuracy*)

   Informasi harus bebas dari kesalahan-kesalahan dan tidak bisa atau menyesatkan, dan harus jelas mencerminkan maksudnya. Ketidakakuratan dapat terjadi karena sumber informasi (data) mengalami gangguan atau kesengajaan sehingga merusak atau merubah data-data asli tersebut. Komponen akurat : lengkap (*completeness*), benar (*correctness*), aman (*security*).

c. Tepat waktu (*timeliness*)

   Informasi yang dihasilkan atau dibutuhkan tidak boleh terlambat (usang). Informasi yang using tidak mempunyai nilai yang baik, sehingga kalau digunakan sebagai dasar dalam pengambilan keputusan akan berakibat fatal atau kesalahan dalam keputusan dan tindakan. Kondisi demikian menyebabkan mahalnya nilai suatu infomasi, sehingga kecepatan untuk

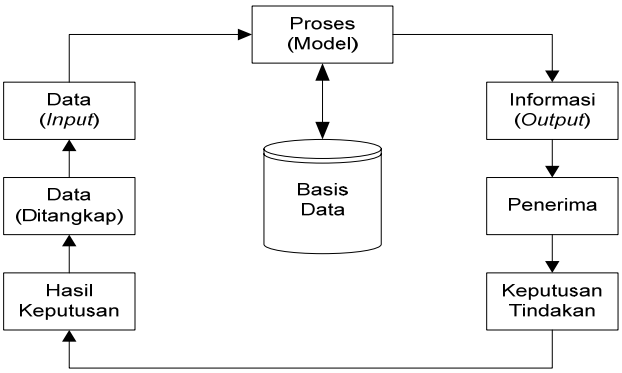
mendapatkan, mengolah, dan mengirimkannya memerlukan teknologi-teknologi terbaru.

#### 2.2.2.2 Nilai Informasi

Nilai dari informasi *(value of information)* ditentukan dari dua hal, yaitu manfaat dan biaya mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya. Kegunaan informasi adalah untuk mengurangi hal ketidakpastian di dalam proses pengambilan keputusan tentang suatu keadaan.

#### 2.2.2.3 Siklus Informasi

Data agar menjadi lebih berarti dan berguna dalam bentuk informasi, maka perlu diolah melalui suatu model tertentu. Data yang telah diolah tersebut kemudian diterima oleh penerima, lalu penerima membuat suatu keputusan dan melakukan tindakan, yang berarti menghasilkan suatu tindakan yang lain yang akan membuat sejumlah data kembali. Data tersebut akan ditangkap sebagai *input*, dan diproses kembali lewat suatu model dan seterusnya sehingga membentuk suatu siklus. Siklus ini disebut dengan siklus informasi (*information cycle*) atau disebut pula siklus pengolahan data (*processing cycles*).




Gambar 2.2 Siklus Informasi

### 2.2.3 Pengertian Sistem Informasi

Sistem informasi dapat didefinisikan sebagai suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai suatu tujuan yaitu menyajikan informasi. Pengertian lain dari sistem informasi adalah sekumpulan prosedur organisasi yang pada saat dilaksanakan dan memberikan informasi bagi pengambil keputusan dan/atau untuk mengendalikan informasi.

#### 2.2.3.1 Konsep Dasar Sistem Informasi

Jhon Bruch dan Gary Grudnitski menerangkan bahwa sistem informasi terdiri dari komponen-komponen yang disebut dengan istilah blok bangunan (*building block*), yaitu : blok masukan (*input block*), blok basis data (*database block*), blok model (*model block*), blok keluaran (*output block*), blok teknologi (*technologi block*), dan blok kendali (*control block*). Sebagai suatu sistem, keenam blok tesebut

masing-masing berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasarannya, antara lain :[1]

1. Blok Masukan

   Input mewakili data yang masuk kedalam sistem informasi. Input disini termasuk metode-metode dan media untuk menangkap data yang akan dimasukan, yang dapat berupa dokumen-dokumen dasar.

2. Blok Model

   Blok ini terdiri dari kombinasi prosedur, logika dan model matematik yang akan memanipulasi data input dan data yang tersimpan di basis data dengan cata tertentu untuk menghasilkan keluaran yang diinginkan.

3. Blok Keluaran

   Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.

4. Blok Teknologi

   Teknologi digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem keseluruhan. Teknologi ini terdiri dari 3 bagian utama, yaitu teknisi (*humanware atau brainware*), perangkat lunak (*software*) dan perangkat keras (*hardware*). Teknisi dapat berupa orang-orang yang mengetahui teknologi dan membuatnya dapat beroperasi. Misalnya teknisis operator komputer, pemograman, operator pengolahan

kata, spesialis telekomunikasi, analisa sistem, penyimpanan data dan sebagainya.

5. Blok Basis Data

   Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat lunak untuk memanipulasinya yang disebut dengan DBMS (*Database Management System*). Data perlu disimpan dalam basis data untuk keperluan penyediaan informasi lebih lanjut. Data didalam basis data perlu di organisasikan sedemikian rupa, supaya informasi yang dihasilkan berkualitas. Organisasi basis data yang baik juga berguna untuk efisiensi kapasitas penyimpanannya.

6. Blok Kendali

   Terdapat banyak hal yang dapat merusak sebuah informasi, misalnya bencana alam, api, temperatur, air, debu, kecurangan-kecurangan, kegagalan-kegagalan sistem itu sendiri, kesalahan-kesalahan, ketidak efisienan, sabotase dan lain sebagainya. Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat langung cepat diatasi.

#### 2.2.3.2 Tujuan Pembangunan Sistem Informasi

Terdapat 3 (tiga) tujuan pembangunan sistem informasi, diantaranya : [1]

1. Integrasi sistem
    a. Menghubungkan sistem individu/kelompok
    b. Pengkolektifan data dan penyambungan secara otomatis
    c. Peningkatan koordinasi dan pencapaian sinergi
2. Efisiensi pengelolaan sistem
    a. Penggunaan basis data dalam upaya kesamaan pengadministrasian data
    b. Pengelolaan data berkaitan dengan karakteristik informasi
    c. Penggunaan dan pengambilan informasi
3. Dukungan keputusan untuk manajemen
    a. Melengkapi informasi guna kebutuhan proses pengambilan kebutuhan
    b. Akuisisi informasi eksternal melalui jaringan komunikasi
    c. Ekstraksi dari informasi internal yang terpadu.

#### 2.2.3.3 Manfaat Sistem Informasi

Sistem informasi memiliki banyak manfaat, diantaranya : [1]

a. Menghemat tenaga kerja
b. Peningkatan efisiensi
c. Mempercepat proses

d. Perbaikan dokumentasi

e. Pencapaian standar

f. Perbaikan keputusan

#### 2.2.3.4 Model Proses Sekuensial Linier atau *Waterfall*

Model proses untuk rekayasa perangkat lunak dipilih berdasarkan sifat aplikasi dan proyeknya, metode dan alat-alat bantu yang akan dipakai dan kontrol serta penyampaian yang dibutuhkan. Perkembangan perangkat lunak bisa dianggap sebagai lingkaran pemecahan masalah dimana terdapat 4 (empat) keadaan berbeda, yaitu status quo, definisi masalah, perkembangan teknis memecahkan masalah di keseluruhan aplikasi dari banyak aplikasi, dan integrasi pemecahan menyampaikan hasil kepada siapa yang membutuhkan pertama kali. Sekuensial linier mengusulkan sebuah pendekatan kepada perkembangan perangkat lunak yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. [8]

### 2.2.4 Pengertian Data

*"Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata serta merupakan suatu istilah jamak dari kata datum yang berarti fakta yang menghubungkan dengan kenyataan, simbol-simbol, ganbar-gambar, kata-kata, dan angka-angka yang menunjukkan suatu ide, obyek, kondisi, dan situasi"*.[3]

Maka dapat di artikan sebagai sumber informasi yang merupakan bahan mentah dari informasi yang masih membutuhkan suatu pengolahan dimana data yang telah diolah dapat digunakan sebagai suatu informasi yang dibutuhkan serta sebagai pertimbangan dalam melakukan aktifitas selanjutnya, sehingga data dapat berguna untuk bahan dasar yang objektif dalam proses penyusunan dan pengolahan keputusan bagi suatu organisasi.

### 2.2.5 Pengertian Basis Data

Basis Data (*Database*) dapat dibayangkan sebagai sebuah lemari arsip. Basis Data terdiri dari atas dua kata, yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan dan sebagainya, yang direkam dalam bertuk angka, huruf, symbol, teks, gambar, bunyi, atau kombinasinya.[4]

> "*Basis Data dapat diartikan sebagai himpunan atau sekumpulan data bisa berupa tabel atau file yang saling berhubungan dan disimpan dalam media penyimpanan elektronis tanpa pengulangan* (*redudansi).*"[4]

Setiap basis data umumnya dibuat untuk mewakili sebuah semesta data yang spesifik. Karena itu, operasi-operasi dasar yang dapat dilakukan berkenaan dengan basis data dapat meliputi : [4]

1. Pembuatan basis data baru (*create database*), yang identik dengan pembuatan lemari arsip yang baru.
2. Penghapusan basis data (*drop database*), yang identik dengan perusakan lemari arsip beserta isinya.
3. Pembuatan file/tabel baru ke suatu basis data (*create table*), yang identik dengan penambahan map/arsip baru ke sebuah lemari arsip yang telah ada.
4. Penghapusan file/tabel dari suatu basis data (*drop table),* yang identik dengan perusakan map arsip lama yang ada di sebuah lemari arsip.
5. Penambahan/ pengisian data baru kesebuah file/tabel disebuah basis data (*insert),* yang identik dengan penambahan lembaran arsip kesebuah map arsip.
6. Pengambilan data dari sebuah file/tabel (*retrieve/search),* yang identik dengan pencarian lembaran arsip dari sebiuah map arsip.
7. Pengubahan data dari sebuah file/tabel (*update),* yang identik dengan perbaikan isi lembar arsip yang ada di sebuah map arsip.
8. Penghapusan data dari sebuah file/tabel (*delete),* yang identik dengan penghapusan sebuah lembaran arsip yang ada di sebuah map arsip.

Tujuan dari basis data erat hubungannya dengan masalah-masalah yang timbul dalam *file* basis data, yaitu : [4]

1. Penyediaan akses yang *fleksibel*, yaitu untuk memberikan kemudahan dalam menampilkan semua informasi yang diperlukan.

2. Pemeliharaan integritas data, basis data berisi *file* yang saling berkaitan yaitu dengan adanya *field* kunci yang menghubungkan kedua *file* tersebut.
3. Keamanan data, basis data dapat mendefinisikan prosedur otoritas untuk memastikan bahwa data hanya dapat diakses oleh *user* yang berhak dan dapat mengijinkan *user* yang berbeda untuk melakukan tiap akses yang pada tahap data yang sama.
4. Menghilangkan ketergantungan data, program yang berinteraksi dengan DBMS (*Database Management System*) relatif mandiri terhadap data aktual dalam basis data. Hanya perubahan yang mungkin dilakukan terhadap struktur data tanpa membutuhkan perubahan terhadap program aplikasi yang telah ada.
5. Mengurangi kerangkapan data.
6. Penggunaan data bersama-sama, data yang sama dapat diakses atau digunakan oleh beberapa *user* pada saat bersamaan. Hal ini menggunakan sistem basis data, tiap aplikasi mempunyai *file* tersendiri, sehingga suatu data tunggal dalam basis data dapat digunakan untuk beberapa kegunaan.
7. Standarisasi data, yaitu fasilitas-fasilitas kamus data untuk mendefinisikan nama-nama data secara rinci.

### 2.2.6 Pengertian Sistem Basis Data

Sistem basis data adalah gabuangan antara basis data dan perangkat lunak DBMS termasuk programnya aplikasi yang dibuat dan bekerja dalam suatu sistem yang bertujuan untuk dapat memanipulasi data dari basis data sehingga diperoleh informasi yang diinginkan. [4]

### 2.2.7 *Object Oriented*

Secara spesifik, pengertian "berorientasi objek" berarti bahwa kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang sangat berbeda dengan pemrograman konvensional di mana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup 4 (empat) hal, yaitu dentifikasi, klasifikasi, *polymorphisme* dan *inheritance.*

Pendekatan berorientasi objek adalah cara memandang persoalan menggunakan model-model yang diorganisasikan seputar konsep objek yang mengkombinasikan struktur data dan perilaku suatu entitas. Pada pendekatan ini, organisasi perangkat lunak adalah sebagai kumpulan objek diskrit yang saling bekerja sama, berkomunikasi dan berinteraksi menuju sasaran tertentu.[5]

Pendekatan berorientasi objek mempunyai keunggulan sebagai berikut :[5]

1. Bekerja yang mendekati kognisi manusia.
2. Menghasilkan sistem yang dibangun diatas 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 pemanduan terjadi menyebar pada siklus kehidupan tidak terjadi dalam sekejab sebagaimana kejadian ledakan besar (*big bang*).
5. Pendekatan ini membantu mengeksploitasi keampuhan bahasa pemrograman berbasis objek (ADT, seperti Ada, Modula-2, Visual Basic sebelum versi 5.0) atau berorientasi objek.

#### 2.2.7.1 *Object Oriented* Analisis dan Design (OOAD)

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas. [5]

**A. Analisis Berorientasi Objek**

Analisis berorientasi objek mendefinisikan semua kelas yang relevan terhadap masalah beserta operasi-operasi dan atribut atribut yang diaosiasikan dengan kelas itu, keterhubungan di kelas-kelas dan perilaku yang dimilikinya. Sasaran

analisis berorientasi objek adalah untuk mengembangkan model yang mendeskripsikan perangkat lunak yang memenuhi sekelompok kebutuhan yang didefinisikan pemesan. Analisis berorientasi objek menggunakan sejumlah pemodelan untuk memenuhi sasaran. Model analisis akan mengekspresikan informasi, perilaku dan fungsi di dalam konteks model objek.

Analisis menghasilkan pandangan logik (*logical view*) terhadap sistem untuk mendeskripsikan keberadaan dan maksud abstraksi dan mekanisme kunci yang membentuk ruang masalah atau yang mendefinisikan arsitektur sistem. Tujuan dari analisis berorientasi objek adalah memodelkan sistem dunia eksternal sehingga dapat dipahami. Untuk melakukan ini harus memeriksa kebutuhan, menganalisis implikasinya dan menyatakannya kembali dengan lebih bagus menggunakan notasi/bahasa spesifikasi yang disepakati.

## B. Perancangan Berorientasi Objek

Perancangan berorientasi objek (OOD-*object Oriented Design*) merupakan disiplin yang lebih kompleks dibanding perancangan konvensional. Kompleksitas muncul antara lain dari keberagaman yang dibahas di perancangan berorientasi objek. Ragam objek dan kelas serta keterhubungan sungguh banyak. Kompleksitas ini menyebabkan OOD lebih sulit untuk dipelajari dibanding disiplin perancangan konvensional. Namun demikian, OOD sungguh memberi kekayaan dan daya ekspresi dibanding teknik konvensional.

OOD (*object Oriented Design*) mengidentifikasi dan mendefinisikan kelas-kelas dan objek-objek tambahan yang merefleksikan implementasi. Perancangan berorientasi onjek adalah proses disiplin untuk menghasilkan kumpulan model yang mendeskripsikan beragam aspek sistem perangkat lunak menggunakan notasi yang terdefinisi baik. Tujuan perancangan berorientasi objek adalah menghasilkan model atau representasi entitas yang akan dibangun. Proses pembangunan model mengkombinasikan antara intuisi dan pertimbangan didasarkan :[5]

1. Pengalaman dalam membangun entitas-entitas serupa.
2. Sekumpulan prinsip dan heuristik yang menuntun jalan dalam evolusi model.
3. Sekumpulan kriteria yang memungkinkan mempertimbangkan kualitas.
4. Proses iterasi yang menuntun ke representasi rancangan akhir.

#### 2.2.7.2 Objek

Rumbaugh mendefinisikan objek sebagai konsep, abstraksi atau sesuatu dengan batas dan arti yang jelas untuk suatu masalah yang dihadapi. Objek menjelaskan dua tujuan, yaitu pengertian tentang dunia nyata dan dilengkapi dengan dasar praktek untuk implementasi komputer. Dekomposisi dari suatu masalah ke dalam objek tergantung pada keputusan dan kenyataan dari masalah tersebut. Tak ada sesuatu penyajian yang sempurna. Dalam term OOP, object adalah sebuah struktur yang menggabungkan data dan prosedur untuk bekerja bersama-sama. Objek itu

secara mudahnya dapat dikatakan terdiri dari property dan metode. Konsep *object oriented* memiliki karakteristik utama yaitu : [5]

**1. Enkapsulasi**

Enkapsulasi adalah suatu mekanisme untuk menyembunyikan atau memproteksi suatu proses dari kemungkinan interferensi atau penyalahgunaan dari luar sistem sekaligus menyederhanakan penggunaan system itu sendiri. Akses ke internal sistem diatur sedemikian rupa melalui seperangkat interface. Contoh kasus sepeda motor, pada sistem pemindahan gigi transmisi, maka pengendara tidak perlu tahu detail dari bagaimana proses pemindahan gigi itu dilakukan oleh mesin, cukup tahu bagaimana menekan gigi transmisi itu. Pedal gigi transmisi yang diinjak pengendara itu merupakan interface (antar muka) pengendara dengan sistem transmisi sepeda motor.

**2. Pewarisan (*Inheritance*)**

Sebagai manusia kita sebenarnya terbiasa untuk melihat objek yang berada disekitar kita tersusun secara hierarki berdasarkan *class*-nya masing-masing. Dari sini kemudian timbul suatu konsep tentang pewarisan yang merupakan suatu proses dimana suatu *class* diturunkan dari class lainnya sehingga ia mendapatkkan ciri atau sifat dari *class* tersebut. Perhatikan contoh hirarki berikut ini:

Dari hirarki diatas dapat dilihat bahwa, semakin kebawah, *class* akan semakin bersifat spesifik. *Class* mamalia memiliki seluruh sifat yang dimiliki oleh binatang, demikian halnya juga Anjing, Kucing dan Monyet memiliki seluruh sifat yang diturunkan dari *class* mamalia. Dengan konsep ini, karakteristik yang dimiliki

oleh *class* binatang cukup didefinisikan didefinisikan dalam *class* binatang saja. *Class* mamalia tidak perlu mendefinisikan ulang apa yang telah dimiliki oleh class binatang, karena sebagai *class* turunannya, ia akan mendapatkan karakteristik dari *class* binatang secara otomatis. Demikian juga dengan *class* anjing, kucing dan monyet, hanya perlu mendefinisikan karakteristik yang spesifik dimiliki oleh *class*-nya masing-masing. Dengan memanfaatkan konsep pewarisan ini dalam pemrograman, maka hanya perlu mendefinisikan karakteristik yang lebih umum akan didapatkan dari *class* darimana ia diturunkan.

**3. *Polymorphism***

*Polymorphism* berasal dari bahasa Yunani yang berarti banyak bentuk. Dalam PBO, konsep ini memungkinkan digunakannya suatu *interface* yang sama untuk memerintah objek agar melakukan aksi atau tindakan yang mungkin secara prinsip sama namun secara proses berbeda. Dalam konsep yang lebih umum sering kali *polymorphism* disebut dalam istilah satu *interface* banyak aksi.

### 2.2.7.3 Kelas

Kelas merupakan pengkapsulan nilai-nilai atribut dan layanan-layanan eksklusifnya. Objek adalah instan dari kelas. Maka kelas merupakan deskripsi satu objek atau lebih dengan sekumpulan atribut dan layanan yang seragam, termasuk deskripsi penciptaan objek baru di kelas itu. Kelas adalah tipe data abstrak dilengkapi dengan implementasi parsial atau total. Kelas objek mendeskripsikan kelompok objek

dengan properti-properti (atribut-atribut) serupa,perilaku (operasi-operasi) yang *common*, keterhubungan terhadap himpunan objek-objek lain dan semantik yang *common*. [5]

Perbedaan kelas dan objek adalah objek merupakan entitas kongkret yang ada secara ruang dan waktu, sedangkan kelas hanya meruapakan representasi abstraksi. Objek adalah bukan kelas, meski kelas boleh jadi menjadi objek. Pendekatan berorientasi objek memerlukan identifikasi kelas dan objek untuk menyelaraskan representasi sistem lebih mendekati pandangan konseputal dunia eksternal. Mengabstraksikan domain masalah dengan kelas dan objek berdampak pada pemahaman dan komunikasi efektif. Pendefinisian objek dan kelas sebagai abstraksi dunia eksternal membantu meningkatkan pemahaman kita dan komunikasi mengnai domain masalah, dapat diterapkan untuk sistem yang dikaji dan dengan pandangan menuju ke penggunaan ulang hasil-hasil analisis, perancangan dan pemrograman. [5]

### 2.2.8 *Unified Modeling Language* (UML)

Pendefinisian *Unified Modeling Language* (UML) menurut para ahli adalah sebagai berikut : [5]

1. Menurut (Hend, 2006) "*Unified Modeling Language* (UML) adalah bahasa yang telah menjadi standard untuk visualisasi, menetapkan, membangun dan mendokumentasikan artifak suatu sistem perangkat lunak".

2. Menurut (Adi Nugroho : 2005). "*Unified Modeling Language* (UML) adalah alat bantu analisis serta perancangan perangkat lunak berbasis objek".
3. Menurut (Joomla dari http://soetrasoft.com : 2007). "*Unified Modeling Language* (UML) merupakan *standard modeling language* yang terdiri dari kumpulan-kumpulan diagram, dikembangkan untuk membantu para pengembang sistem dan *software* agar bisa menyelesaikan tugas-tugas seperti : Spesifikasi, Visualisasi, Desain Arsitektur, Konstruksi, Simulasi dan testing serta Dokumentasi".

Berdasarkan beberapa pendapat yang dikemukakan diatas dapat ditarik kesimpulan bahwa "*Unified Modeling Language* (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis OO (*Object Oriented*)". [5]

#### 2.2.8.1 Tujuan Penggunaan UML

Ada 3 tujuan penggunaan UML, antara lain : [5]

1. Menyediakan perangkat pemodelan visual yang ekspresif dan dimengerti secara umum.
2. Memberikan bahasa pemodelan yang tidak dipengaruhi bahasa pemrograman dan proses rekayasa apapun.

3. Menyatukan praktek-praktek yang terbaik yang terdapat dalam pemodelan yang sudah ada.

#### 2.2.8.2 Gambaran dari UML

Adapun gambaran dari UML, sebagai berikut : [5]

**A. UML Sebagai Bahasa Pemodelan**

UML fokus pada pemahaman subjek yang melalui formulasi model dari subjek (dan konteks yang terhubung). Model memuat pengetahuan pada subjek, dan aplikasi dari pengetahuan ini berkaitan dengan intelejensia.

**B. UML Sebagai Bahasa *Visualizing***

UML dapat digunakan untuk menjelaskan sistem secar visual sebelum direalisasikan.

**C. UML Sebagai Bahasa *Specifying***

UML dapat digunakan untuk mengkomunikasi “apa” yang diperlukan dari sistem dan “bagaimana” sistem dapat direalisasikan.

**D. UML Sebagai Bahasa *Constructing***

UML dapat digunakan untuk membangun sebuah perangkat lunak sistem intensif.

**E. UML Sebagai Bahasa *Documenting***

UML dapat digunakan untuk menangkap pengetahuan mengenai sistem pada seluruh siklus hidup.

#### 2.2.8.3 Bagian-bagian dari UML

Adapun bagian-bagian dari UML, sebagai berikut : [5]

**A. *View***

*View* digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis *view* dalam UML antara lain : *use case view*, *logical view*, *component view*, *concurrency view*, dan *deployment view*.

***Use case view***

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan external actors. *Actor* yang berinteraksi dengan sistem dapat berupa *user* atau sistem lainnya. *View* ini digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams*. *View* ini digunakan terutama untuk pelanggan, perancang (*designer*), pengembang (*developer*), dan penguji sistem (*tester*).

***Logical view***

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class*, *object*,dan *relationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam *class diagrams* untuk struktur statis dan dalam *state*, *sequence*, *collaboration,* dan *activity diagram* untuk model dinamisnya. *View* ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).

***Component view***

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi *administrative* lainnya. *View* ini digambarkan dalam *component view* dan digunakan untuk pengembang (*developer*).

***Concurrency view***

Membagi sistem ke dalam proses dan prosesor. *View* ini digambarkan dalam diagram dinamis (*state*, *sequence*, *collaboration*, dan *activity diagrams*) dan diagram implementasi (*component* dan *deployment diagrams*) serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

***Deployment view***

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya. *View* ini digambarkan dalam *deployment diagrams* dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

## B. Diagram

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis diagram antara lain :

***Use-Case Diagram***

*Use-case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use-case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use-case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

*Use-case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan *client*, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use-case* dapat meng-*include* fungsionalitas *use-case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use-case* yang di-*include* akan dipanggil setiap kali *use-case* yang meng-include dieksekusi secara normal.

Sebuah *use-case* dapat di-include oleh lebih dari satu *use-case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use-case* juga dapat meng-*extend use-case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use-case* menunjukkan bahwa *use-case* yang satu merupakan spesialisasi dari yang lain.

***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 3 (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*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

**Hubungan Antar *Class***

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* menunjukkan arah *query* antar *class*. 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.

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.

***Statechart Diagram***

*Statechart diagram* menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat

dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

***Activity Diagram***

*Activity diagrams* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

*Activity diagram* merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* 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-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 seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. Decision digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses *paralel* (*fork* dan *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.

***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.

*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*. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity*.

***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*, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki prefiks yang sama.

***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.

***Deployment Diagram***

*Deployment/physical diagram* menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana 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.

### 2.2.9 Sistem *Inventory Control*

*Inventory* merupakan kata lain dari persediaan, istilah persediaan disini maksudnya menunjukkan barang-barang yang dimiliki perusahaan. Persediaan dapat mengambil bentuk yang tergantung pada jenis usaha yang ditekuni oleh perusahaan

yang bersangkutan. Persediaan barang merupakan salah satu unsur yang paling efektif dalam operasional perusahaan, yang secara berkelanjutan digunakan dalam kegiatan barang harian yang dapat disajikan dalam bentuk laporan persediaan barang.

Menurut AGU “ditinjau dari segi neraca persediaan adalah barang-barang atau bahan-bahan yang masih tersedia pd tanggal neraca atau barang-barang yang akan segera dijual, digunakan atau diproses dalam periode normal perusahaan. Sifat dan wujud persediaan sangat bervariasi tergantung jenis, sifat dan bidang usaha perusahaan”.

Menurut RAN “persediaan merupakan bahan-bahan yang disediakan dan bahan-bahan dalam proses yang terdapat dalam perusahaan untuk produksi suatu barang-barang jadi atau produk yang disediakan untuk memenuhi permintaan dari konsumen atau langganan setiap periode”.

Laporan persediaan barang adalah suatu laporan yang menyajikan tentang data-data barang yang masuk dan data-data barang yang keluar dalam suatu perusahaan. Hal ini sangat penting sekali bagi perusahaan untuk melakukan pemeriksaan barang yang tersedia dan barang-barang yang habis persediaannya.

Menurut fungsi, persediaan dibagi menjadi 3 (tiga) jenis antara lain : [9]

1. *Batch stock/Lot size inventory*

   Persediaan yang diadakan karena membeli atau membuat bahan-bahan atau barang-barang dalam jumlah yang lebih besar dari jumlah yang dibutuhkan saat itu.

2. *Fluctuation*

   Persediaan yang diadakan untuk menghadapi fluktuasi permintaan yang tidak dapat diramalkan.

3. *Anticipation stock*

   Persediaan yang diadakan untuk menghadapi fluktuasi permintaan yang dapat diramalkan, berdasarkan pola musiman yang terdapat dalam satu tahun dan untuk menghadapi penggunaan atau penjualan atau permintaan yang meningkat.

Sistem *inventory control* adalah suatu paket perangkat lunak dan perangkat keras yang digunakan dalam operasi gudang, dan tempat-tempat lain, untuk memantau jumlah, lokasi dan status *inventory* serta terkait pengiriman, penerimaan, dan memilih *putaway* proses. Sistem *inventory control* tujuannya adalah untuk menjaga informasi tentang kegiatan-kegiatan dalam perusahaan yang memastikan penyampaian produk kepada pelanggan.

Secara umum model persediaan dapat dikelompokkan menjadi 2 (dua) model, yaitu antara lain : [9]

1. Model Deterministik, yakni model yang menganggap semua variabel telah diketahui dengan pasti.
2. Model Probabilistik, yakni model yang menganggap semua variabel mempunyai nilai-nilai yang tidak pasti dan satu atau lebih variabel tersebut merupakan variabel-variabel acak.

Adapun beberapa macam model persediaan deterministik yang biasanya digunakan, antara lain : [9]

1. Model persediaan deterministik dengan *backorder*

   Suatu *backorder* adalah permintaan yang tidak dapat dipenuhi pada saat sekarang, tetapi kemudian dipenuhi pada periode yang akan datang. Didalam situasi yang bersifat *backorder*, suatu perusahaan tidak kehilangan penjualan (pelanggan yang tidak terpenuhi) ketika *inventory*-nya habis. Sebagain ganti, karena kesetiaan pelanggan akan produk tertentu mau berada dalam masa penantian pelanggan untuk terpenuhi permintaan mereka ketika perusahaan menerima *order*/pesanan yang berikutnya.

2. Model persediaan deterministik dengan potongan harga/diskon

   Suatu potongan harga akan digunakan/diberikan oleh penjual kepada pembeli dengan syarat pembelian barang dalam jumlah yang lebih besar, karena sebagai suatu perangsang ekonomi kepada para pembeli.

3. Model persediaan deterministik dengan perubahan harga

   Kondisi menggunakan model perubahan harga lebih dikhususkan pada saat-saat tertentu saja. Perusahaan biasanya memberikan harga yang *special*, misalnya untuk merayakan hari ulang tahun perusahaan, hari kemerdekaan suatu negara atau bahkan hari-hari besar keagamaan. Hal ini dinamakan dengan model *special sale price*, yang biasanya harga tawaran produknya jauh lebih kecil dibandingkan dengan harga normal dan ini tidak berlaku untuk hari-hari yang lain.

4. Model persediaan deterministik dengan laju kedatangan *uniform*

   Model persediaan dengan laju produksi, sangat biasa terjadi pada *buffer stock* yang ada di tiap departemen atau mesin produksi. Persediaan yang ada di *bufferstock* akan menerima barang pesanan dari depertemen sebelumnya secara bertahap sehingga memiliki laju atau kecepatan kedatangan. Apabila pola kedatangan barang pesanan dari departemen sebelumnya adalah serentak, maka model ini akan kembali pada model persediaan sederhana tanpa adanya laju kedatangan.

5. Model persediaan deterministik dinamis

   Suatu dinamis adalah suatu permintaan yang bervariasi dari satu periode ke periode yang lain pada setiap awal periode. Pokok perencanaan hanya terbatas pada beberapa periode waktu saja. Apabila tidak ada pesanan pada bagian penjadwalan, maka jumlah permintaannya adalah nol. Biaya-biaya tambahan selalu dikurangi dengan penjadwalan pada permintaan periode berikutnya.

Model persediaan deterministik dengan potongan harga akan digunakan pada saat perusahaan membeli barang ke *supplier* dengan cara memesan dalam ukuran *lot* (kelompok) yang lebih besar. Manfaat dari model potongan harga bagi perusahaan adalah penjualan dalam jumlah yang lebih banyak akan mengurangi biaya produksi tiap unitnya.

Pada potongan harga terdapat 2 (dua) jenis umum karena kuantitas barang pesanan yang ditawarkan oleh para *supplier*, adalah sebagai berikut : [9]

1. Potongan *all-unit*, yakni pembelian dalam jumlah yang lebih besar mengakibatkan adanya suatu harga tiap satuan yang lebih rendah untuk keseluruhan paket pemesanan.
2. Potongan incremental, yakni menerapkan atau memberlakukan harga per satuan yang lebih rendah hanya untuk membeli unit diatas suatu kuantitas tertentu saja sesuai yang diterapkan perusahaan secara berjenjang.

Tujuan menggunakan model persediaan deterministik dengan potongan harga adalah sebagai berikut : [9]

1. Untuk dapat memenuhi kebutuhan atau permintaan konsumen dengan cepat (memuaskan konsumen).
2. Untuk menjaga kontinuitas produksi atau menjaga agar perusahaan tidak mengalami kehabisan persediaan yang mengakibatkan terhentinya proses produksi.
3. Untuk mempertahankan dan bila mungkin meningkatkan penjualan dan laba perusahaan.
4. Menjaga supaya penyimpanan dalam *emplacement* tidak besar-besaran, karena akan mengakibatkan biaya menjadi besar.

Jumlah potongan (*Quantity Discount*) seringkali diberikan oleh penjual kepada pembeli jika membeli dalam jumlah tertentu yang cukup besar. Pesanan pembelian optimal dapat dipengaruhi oleh adanya kebijakan jumlah potongan (*Quantity Discount*) ini, sehingga harga pembelian barang dipertimbangkan sebagai *variable input* dalam menghitung EOQ (*Economic Order Quantity*).

Untuk menentukan berapa persediaan yang harus dipesan setiap kali pesan, dapat dilakukan langkah-langkah berikut ini : [9]

1. Menghitung Q (jumlah pesanan per unit atau kuantitas per *range*) untuk setiap tingkat diskon.
2. Pada tingkat diskon masing-masing dipertimbangkan apakah perlu dilakukan penyesuaian jumlah atau tidak.
3. Menghitung biaya pembelian untuk setiap Q.

Rumus untuk menghitung Q (jumlah pesanan per unit), sebagai berikut : [9]

$$Q = \sqrt{\frac{2CR}{PF}}$$

Keterangan :

Q = jumlah pesanan per unit atau kuantitas per *range*

C = biaya pesan per unit

R = jumlah permintaan barang

P = harga per unit

F = biaya simpan dari harga beli per unit

EOQ (*Economic Order Quantity*) adalah jumlah kuantitas barang yang dapat diperoleh dengan biaya yang minimal/pembelian paling ekonomis. EOQ dipengaruhi oleh biaya persediaan (*Inventory Cost*). Biaya persediaan adalah biaya yang dikeluarkan dalam rangka pengadaan persediaan hingga digunakan untuk kegiatan operasional (selain harga barang/bahan). Biaya persediaan dapat berupa biaya tetap maupun variabel. Biaya persediaan variabel dibedakan menjadi 2 (dua), yaitu : [11]

1. Biaya pesan (*Procurement Cost/set-up cost)*, biaya yang berubah sesuai frekuensi pesan.
2. Biaya simpan (*Carrying Cost*), biaya yang berubah-ubah sesuai dengan perubahan persediaan.

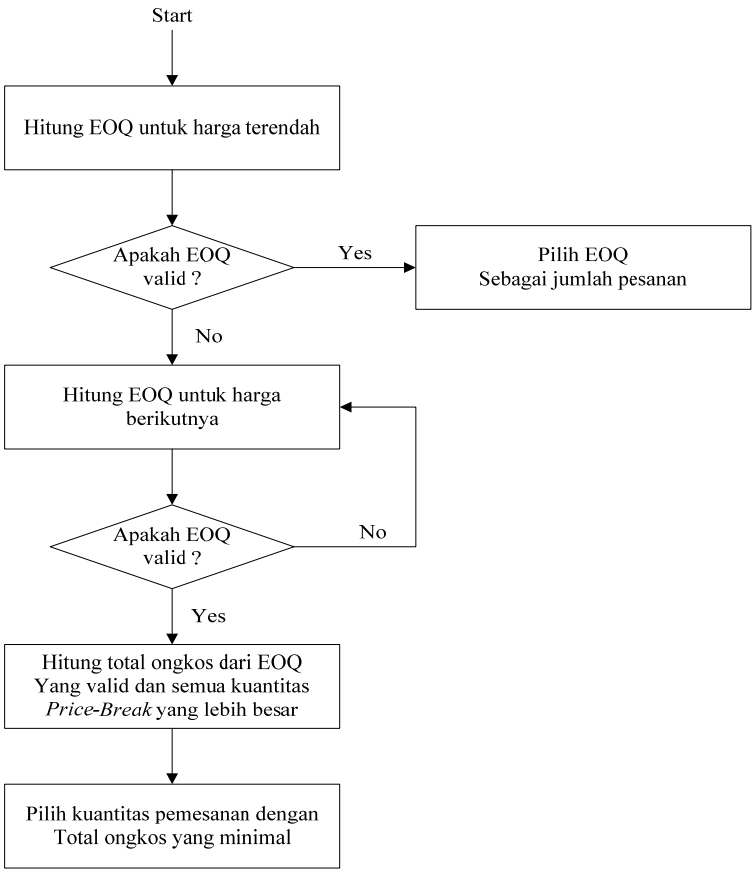
Setelah langkah-langkah untuk menentukan berapa persediaan yang harus dipesan setiap kali pesan maka diterapkan bagaimana cara memperoleh pemesanan kuantitas yang minimum ketika satu atau lebih potongan *all-units*, sebagai berikut : [9]

1. Mulai dengan biaya unit yang paling rendah, mengkalkulasi EOQ (*Economic Order Quantity*) pada biaya unit masing-masing sampai suatu EOQ (*Economic Order Quantity*) sah atau valid diperoleh.
2. Kalkulasi total biaya tahunan untuk EOQ (*Economic Order Quantity*) yang sah dan total biaya untuk setiap *price break* dimana untuk jumlah yang lebih besar dari EOQ (*Economic Order Quantity*) yang sah

tersebut (suatu kuantitas *price break* adalah kuantitas yang paling rendah dimana potongan harga ada tersedia).

3. Pilihlah kuantitas dengan total biaya yang paling rendah sesuai dari langkah 2 diatas.

Berikut gambaran dari langkah-langkah jumlah potongan *all-units,* seperti di bawah ini : [9]




Gambar 2.3 Jumlah Potongan *All-Units*

Biaya pembelian adalah harga per unit apabila unit dibeli dari pihak luar, atau biaya produksi per unit apabila di produksi dalam perusahaan. Penetapan dari biaya pembelian ini tergantung dari pihak penjualan barang atau bahan sehingga pihak pembeli hanya bisa mengikuti fluktuasi harga barang yang ditetapkan oleh pihak penjual. Namun karena biaya pembelian barang atau bahan per unit tidak terpengaruh pada keputusan apapun yang diambil, maka biaya pembelian barang atau bahan per unit tidak perlu dalam pengambilan keputusan. [11]

Rumus biaya pembelian per unit sebagai berikut : [11]

Biaya pembelian per unit = jumlah permintaan barang x harga beli per unit

Keterangan :

Jumlah permintaan barang : jumlah yang dipesan oleh pembeli kepada penjual.

Harga beli per unit : harga tiap unit berdasarkan tingkat diskon

Biaya pesan (C) adalah biaya yang dikeluarkan sehubungan dengan pemesanan barang ke *supplier*. Biaya pesan meliputi : [11]

1. Biaya selama proses persiapan : biaya untuk persiapan pesanan, penentuan kuantitas pesanan
2. Biaya pengiriman pesanan
3. Biaya penerimaan barang yang dipesan : pembongkaran dan pemasukan ke gudang, pemeriksaan, laporan penerimaan, dan mencatat

4. Biaya proses pembayaran : pengecekan keabsahan, persiapan pembuatan cek, pengiriman cek dan auditingnya.

Besar kecilnya biaya pesan sangat tergantung pada frekuensi pesanan, semakin sering memesan barang maka biaya yang dikeluarkan akan semakin besar dan sebaliknya. Biaya pesan dapat pula diartikan yaitu biaya yang diperlukan pada saat pengiriman barang atau biaya yang diperlukan untuk memesan barang setiap kali pengiriman barang, semua biaya yang timbul akan ditanggung oleh perusahaan pemesanan biaya.

Rumus biaya pesan per tahun sebagai berikut : [9]

$$\text{Biaya pesan per tahun} = \frac{\text{jumlah permintaan barang}}{Q} \text{ x biaya pesan}$$

Keterangan :

Q = jumlah pesanan per unit atau kuantitas per *range*

Biaya simpan (F) adalah biaya yang dikeluarkan atas investasi dalam persediaan dan pemeliharaan maupun investasi sarana fisik untuk menyimpan persediaan, atau dapat pula diartikan bahwa biaya simpan semua biaya yang timbul akibat penyimpanan barang maupun bahan. Biaya simpan meliputi : [11]

1. Biaya sewa gudang
2. Biaya pemeliharaan material
3. Biaya menghitung/menimbang barang

4. Biaya asuransi
5. Biaya modal
6. Pajak barang di gudang

Besar kecilnya biaya simpan sangat tergantung pada jumlah rata-rata barang yang disimpan di gudang. Semakin banyak rata-rata persediaan, maka biaya simpan juga akan besar dan sebaliknya.

Rumus biaya simpan per tahun sebagai berikut : [11]

$$\text{Biaya simpan per tahun} = \frac{\text{Q x harga beli per unit x biaya simpan}}{2}$$

Keterangan :

Q = jumlah pesanan per unit atau kuantitas per *range*

Total biaya ditentukan sebagai jumlah dari ketiga biaya yang diubah sebagai fungsi dari variabel keputusan Q (jumlah pesanan per unit), yaitu jumlah dari biaya pembelian per unit, jumlah dari biaya pesan per tahun dan jumlah dari biaya simpan per tahun. Dengan rumus sebagai berikut : [9]

Total biaya = Biaya pembelian per unit + Biaya pesan per tahun +
Biaya simpan per tahun

### 2.2.10 Sistem *Client-Server*

*Client-server* menjelaskan aplikasi arsitektur dimana permintaan *client* tindakan atau layanan dari penyedia layanan, *server*. Mempertimbangkan sebuah *web browser* dan *server web*. Bila Anda alamat URL dalam jendela *browser*, ia (*client*) meminta halaman web dari *server*. *Server* kembali ke halaman html sebuah *client*, yang mem-parsing halaman (data) dan menampilkan pada komputer Anda.

Sistem *client-server* atau disebut juga sistem tersentralisasi diterapkan pada sebuah jaringan. Sistem *client-server* ini ditujukan untuk mengatasi kelemahan-kelemahan yang terdapat pada sistem sebelumnya. Sistem *client-server* terdiri dari 2 (dua) komponen utama yaitu *client* dan *server*. *Client* berisi aplikasi basis data dan *server* berisi DBMS dan basis data. Setiap aktifitas yang dikehendaki para pemakai akan lebih dulu ditangani oleh *client*. *Client* selanjutnya mengupayakan agar semua proses “sebisa mungkin” ditangani sendiri. Bila ada proses yang harus melibatkan data yang tersimpan pada basis data barulah *client* mengadakan hubungan dengan *server*.

Pada sistem *client-server* untuk memenuhi kebutuhan, *client* akan mengirimkan *message* (pesan/perintah) *query* pengambilan data. Selanjutnya, *server* yang menerima *message* tersebut akan menjalankan *query* tersebut dan hasilnya akan dikirimkan kembali ke *client*. Dengan begitu transfer datanya jauh lebih efisien. Adapun bentuk dari sistem *client-server* yang sederhana. Disamping bentuk *client-server* sederhana terdapat pula bentuk *client-server* yang lebih komplek.

### 2.2.11 Borland Delphi 7.0

Borland Delphi, adalah suatu aplikasi pengembangan perangkat lunak yang berjalan di lingkungan Windows. Borland Delphi menggunakan bahasa pemrograman Pascal. Borland Delphi sendiri merupakan hasil pengembangan bahasa Pascal yang bersifat visual, yang dikembangkan oleh Borland International. Aplikasi yang dapat dikembangkan dengan Borland Delphi berupa *game*, *database*, multimedia dan sebagainya. Setelah muncul versi pertama dari Borland Delphi, setiap *release* Borland Delphi selalu disertai fitur baru dan perbaikan dari versi sebelumnya. [6]

Secara umum, kemampuan Delphi adalah menyediakan komponen-komponen dan bahasa pemrograman yang handal, sehingga memungkinkan untuk membuat program aplikasi sesuai keinginan, dengan tampilan dan kemampuan yang canggih. Khusus untuk pemrograman database, Delphi menyediakan objek yang sangat kuat, canggih dan lengkap, sehingga memudahkan pemrogram dalam merancang, membuat dan menyelesaikan aplikasi database yang diinginkan. Selain itu, Delphi juga dapat menangani data dalam berbagai format database, misalnya format *Microsoft Access*, SyBase, Oracle, FoxPro, Informix, DB2 dan lain-lain. Format database yang dianggap asli dari Delphi adalah Paradox dan dBase. [6]

### 2.2.12 *Microsoft Access*

*Microsoft Access* adalah sebuah program aplikasi basis data komputer relasional yang ditujukan untuk kalangan rumahan dan perusahaan kecil hingga menengah. Aplikasi ini merupakan anggota dari beberapa aplikasi *Microsoft Office*, selain tentunya *Microsoft Word*, *Microsoft Excel*, dan *Microsoft PowerPoint*. Aplikasi ini menggunakan mesin basis data *Microsoft Jet Database Engine*, dan juga menggunakan tampilan grafis yang intuitif sehingga memudahkan pengguna. Versi terakhir adalah *Microsoft Office Access* 2007 yang termasuk ke dalam *Microsoft Office System* 2007. [10]

*Microsoft Access* dapat menggunakan data yang disimpan di dalam format *Microsoft Access*, *Microsoft Jet Database Engine*, *Microsoft SQL Server*, *Oracle Database*, atau semua kontainer basis data yang mendukung standar ODBC. Para pengguna/*programmer* yang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang kompleks, sementara para programmer yang kurang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang sederhana. *Access* juga mendukung teknik-teknik pemrograman berorientasi objek, tetapi tidak dapat digolongkan ke dalam perangkat bantu pemrograman berorientasi objek. [10]

Dalam *Microsoft Access* ada 6 (enam) objek yang disediakan, yaitu antara lain : [10]

1. *Table*, merupakan wadah yang digunakan oleh database relational untuk menangani data.

2. *Query*, adalah informasi mengenai sebuah data.
3. *Form*, digunakan untuk menampilkan dan mengedit informasi yang ada dalam suatu dabase.
4. *Report*, merupakan sebuah laporan yang merangkum informasi dan menyajikannya dalam bentuk yang lebih mudah dibaca dan dipahami.
5. *Macro*, adalah daftar perintah sederhana yang berguna untuk memelihara database.
6. *Module*, merupakan program kecil atau dalam bahasa pemrograman sering disebut *procedure*, kegunaannya hampir semua dengan *macro*.