Bab 2
Landasan Teori

2.1. Sistem

2.1.1. Karakteristik dan Pengertian Sistem

Istilah sistem bukanlah hal yang asing bagi kebanyakan orang. Pada dasarnya, sistem adalah sekumpulan elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan. Sebagai gambaran, jika dalam sebuah sistem terdapat elemen yang tidak memberikan manfaat dalam mencapai tujuan yang sama, maka elemen tersebut dapat dipastikan bukanlah bagian dari sistem.

Dari keberadaan ini dapat ditentukan bahwa sebuah sistem memiliki karakteristik:

- Pertama, sebuah sistem memiliki komponen atau elemen yang sering disebut subsistem.
- Kedua, memiliki batas sistem yang jelas.
- Ketiga, memiliki lingkungan luar sistem.
- Keempat, adanya suatu penghubung sistem.
- Kelima, memiliki *input*, proses dan *output*.
- Keenam, yang dapat kita jadikan pedoman apakah sebuah sistem berhasil atau tidak yakni adanya sasaran atau tujuan.

2.1.1.1. Komponen atau Elemen

Suatu sistem terdiri dari komponen yang saling berinteraksi, artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponen dari suatu sistem dikenal dengan subsistem. Subsistem memiliki sifat-sifat dari sistem itu sendiri dalam menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem juga memiliki sistem yang lebih besar yang dikenal dengan suprasistem.

2.1.1.2. Batas Sistem

Batas sistem merupakan daerah yang membatasi antara sistem yang satu dengan sistem yang lainnya atau dengan lingkungan luarnya. Dengan adanya batas sistem maka sistem dapat membentuk suatu kesatuan, karena dengan batas sistem ini, fungsi dan tugas dari subsistem satu dengan yang lainnya berbeda tetapi tetap saling berinteraksi. Dengan kata lain batas sistem merupakan ruang lingkup atau *scope* dari sistem atau subsistem itu sendiri.

2.1.1.3. Lingkungan Luar Sistem

Lingkungan luar sistem adalah segala sesuatu diluar batas sistem yang mempengaruhi operasi suatu sistem. Lingkungan luar sistem dapat bersifat menguntungkan atau merugikan. Lingkungan luar sistem yang bersifat menguntungkan harus dipelihara dan dijaga agar tidak hilang pengaruhnya, sedangkan lingkungan yang bersifat merugikan harus dimusnahkan agar tidak mengganggu operasi sistem.

2.1.1.4. Penghubung Sistem

Penghubung sistem merupakan suatu media penghubung antara satu subsistem dengan subsistem lainnya yang membentuk satu kesatuan, sehingga sumber-sumber daya mengalir dari subsistem yang satu ke subsistem lainnya. Dengan kata lain melalui penghubung *output* dari subsistem akan menjadi *input* bagi subsistem lainnya.

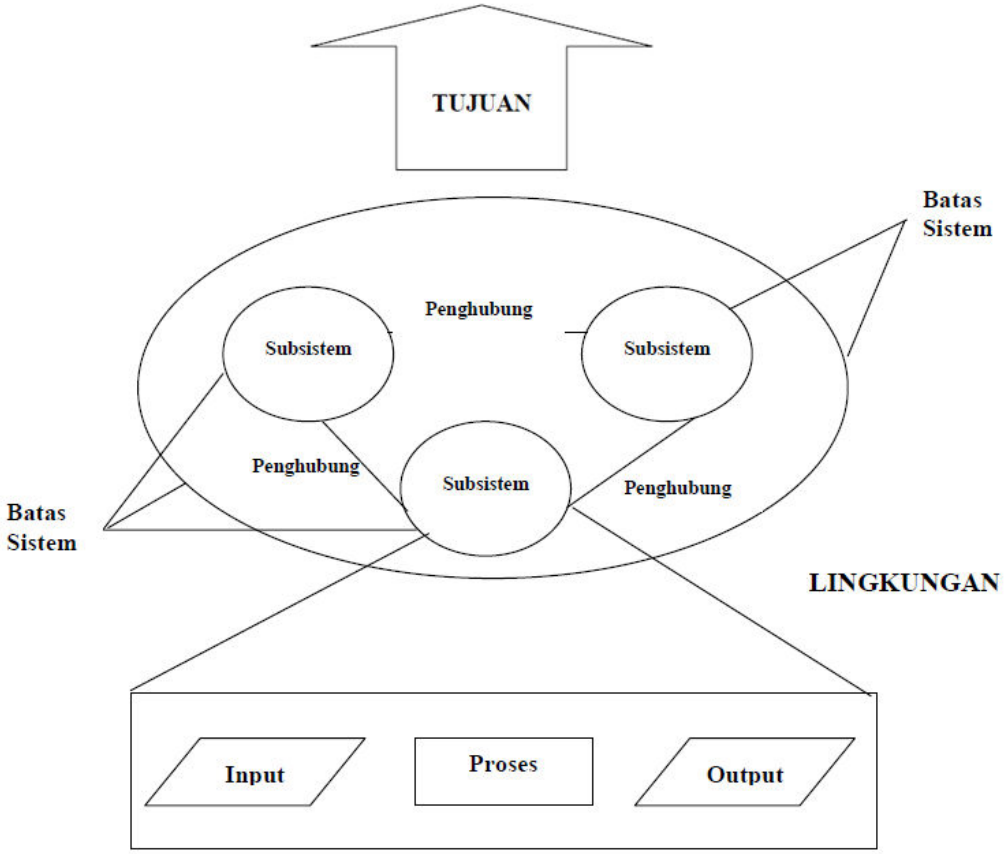
2.1.1.5. *Input*, Proses dan *Output*

Input adalah energi atau sesuatu yang dimasukkan kedalam suatu sistem yang dapat berupa masukkan yaitu energi yang dimasukkan supaya sistem dapat beroperasi atau masukkan sinyal yang merupakan energi yang diproses untuk menghasilkan suatu keluaran. Sistem mempunyai bagian *pengolah* yang akan mengubah *input* menjadi *output* dan sistem juga menghasilkan suatu keluaran atau *output* yang berasal dari energi yang diolah.

2.1.1.6. Tujuan

Setiap sistem mempunyai tujuan atau sasaran yang mempengaruhi *input* yang dibutuhkan dan *output* yang dihasilkan. Dengan kata lain, suatu sistem akan dikatakan berhasil apabila hasil operasi sistem tersebut mengenai sasaran atau tujuan yang telah ditetapkan.

atau



Gambar 2.1. Karakteristik Sebuah Sistem

Sistem terdiri dari struktur dan proses, dimana struktur sistem merupakan komponen-komponen yang membentuk sistem tersebut. Sedangkan proses sistem menjelaskan cara kerja setiap unsur sistem dalam mencapai tujuan sistem. Sistem dibuat untuk menangani sesuatu yang rutin atau berulang kali terjadi.

2.1.2. Pengertian Sistem Informasi

Ada beragam definisi sistem informasi, sebagaimana tercantum dalam tabel 2.1. Dari berbagai definisi tersebut, dapat disimpulkan bahwa sistem informasi mencakup sejumlah komponen (manusia, komputer, teknologi informasi dan prosedur kerja), ada sesuatu yang diproses (data menjadi informasi) dan dimaksudkan untuk mencapai tujuan atau sasaran tertentu.

Tabel 2.1. Defenisi Sistem Informasi

SUMBER	DEFINISI
Alter (1992)	Sistem informasi adalah kombinasi antara prosedur kerja, informasi, orang dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi.
Bodnar dan Hopwood (1993)	Sistem informasi adalah kumpulan perangkat keras dan perangkat lunak yang dirancang untuk mentransformasikan data ke dalam bentuk informasi yang berguna.
Gelinas, Oram dan Wiggins (1990)	Sistem informasi adalah suatu sistem buatan manusia yang secara umum terdiri dari sekumpulan komponen berbasis komputer dan manual yang dibuat untuk menghimpun, menyimpan dan mengelola data serta menyediakan informasi keluaran kepada para pemakai.
Hall (2001)	Sistem informasi adalah sebuah rangkaian prosedur formal dimana data dikelompokkan, diproses menjadi informasi dan didistribusikan kepada pemakai.
Turban, McLean dan Wetherbe (1999)	Sebuah sistem informasi mengumpulkan, memproses, menyimpan, menganalisis dan menyebarkan informasi untuk tujuan yang spesifik.
Wilkinson (1992)	Sistem informasi adalah kerangka kerja yang mengkoordinasikan sumber daya (manusia, komputer) untuk mengubah masukan (*input*) menjadi keluaran (informasi) guna mencapai sasaran-sasaran perusahaan.

Sistem Informasi Manajemen adalah sekumpulan sistem informasi yang saling berinteraksi, yang memberikan informasi baik untuk kepentingan operasi atau kegiatan manajerial (Jogiyanto H.M., 1995) .

Sistem Informasi Manajemen adalah keterpaduan sistem manusia-mesin untuk menyediakan informasi yang mendukung operasi dan menjamin pengambilan keputusan dalam sebuah organisasi (Gordon B. Davis 1974). Sistem ini menggunakan perangkat keras dan perangkat lunak komputer, prosedur pedoman, model untuk analisa, perencanaan, pengawasan dan pengambilan keputusan dan suatu database.

Yang terpenting dari aspek tersebut adalah keseluruhannya. Karena Sistem Informasi Manajemen akan melintasi seluruh sistem penyedia informasi di berbagai lapisan dalam organisasi. Oleh karena itu perlu ditekankan bahwa Sistem Informasi Manajemen adalah kumpulan sistem informasi dan bukan sistem keseluruhan (total sistem*).*

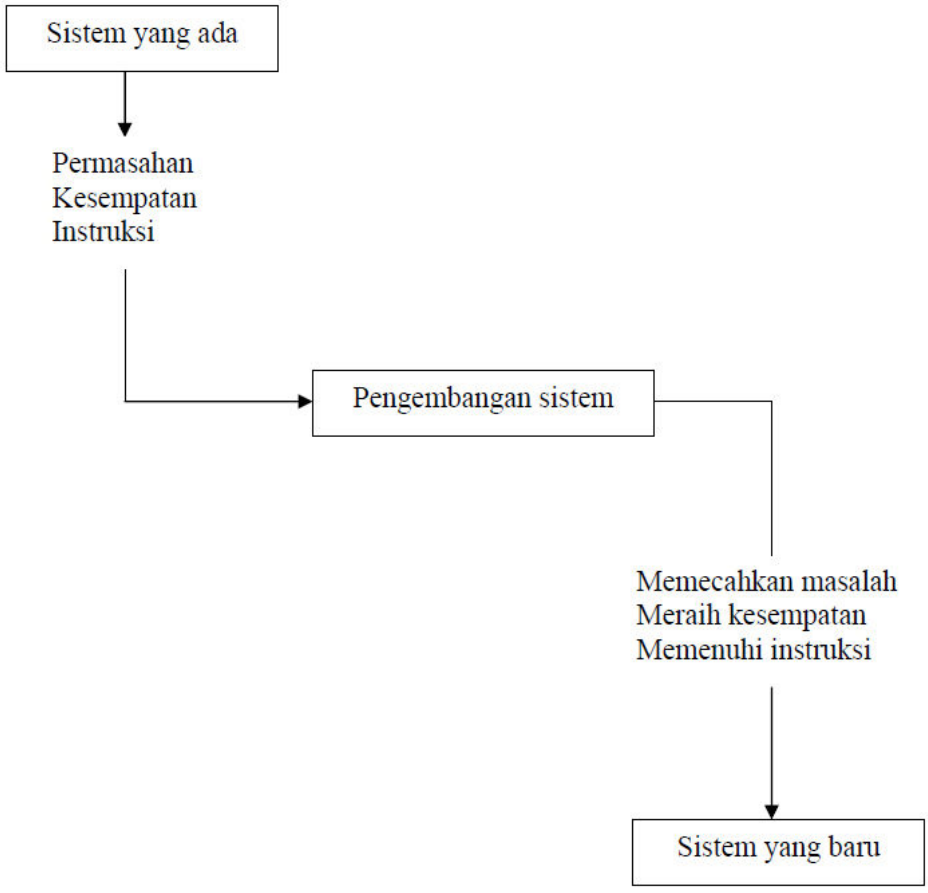
2.2. Pengembangan Sistem

Pengembangan sistem (sistem *development*) dapat berarti menyusun sistem yang baru untuk menggantikan sistem yang lama secara keseluruhan atau memperbaiki sistem yang sudah ada. Sistem yang lama perlu diperbaiki atau diganti disebabkan karena beberapa hal, yaitu sebagai berikut:

1. Adanya permasalahan-permasalahan *(problems)* yang timbul di sistem yang lama. Permasalahan yang timbul dapat berupa:
   - Ketidakberesan dalam sistem yang lama yang menyebabkan sistem yang lama tidak dapat beroperasi sesuai dengan yang diharapkan.
   - Pertumbuhan organisasi, yang menyebabkan harus disusunnya sistem yang baru. Pertumbuhan organisasi diantaranya adalah kebutuhan informasi yang semakin luas, volume pengolahan data yang semakin meningkat, perubahan prinsip akuntansi atau pengolahan data yang baru.

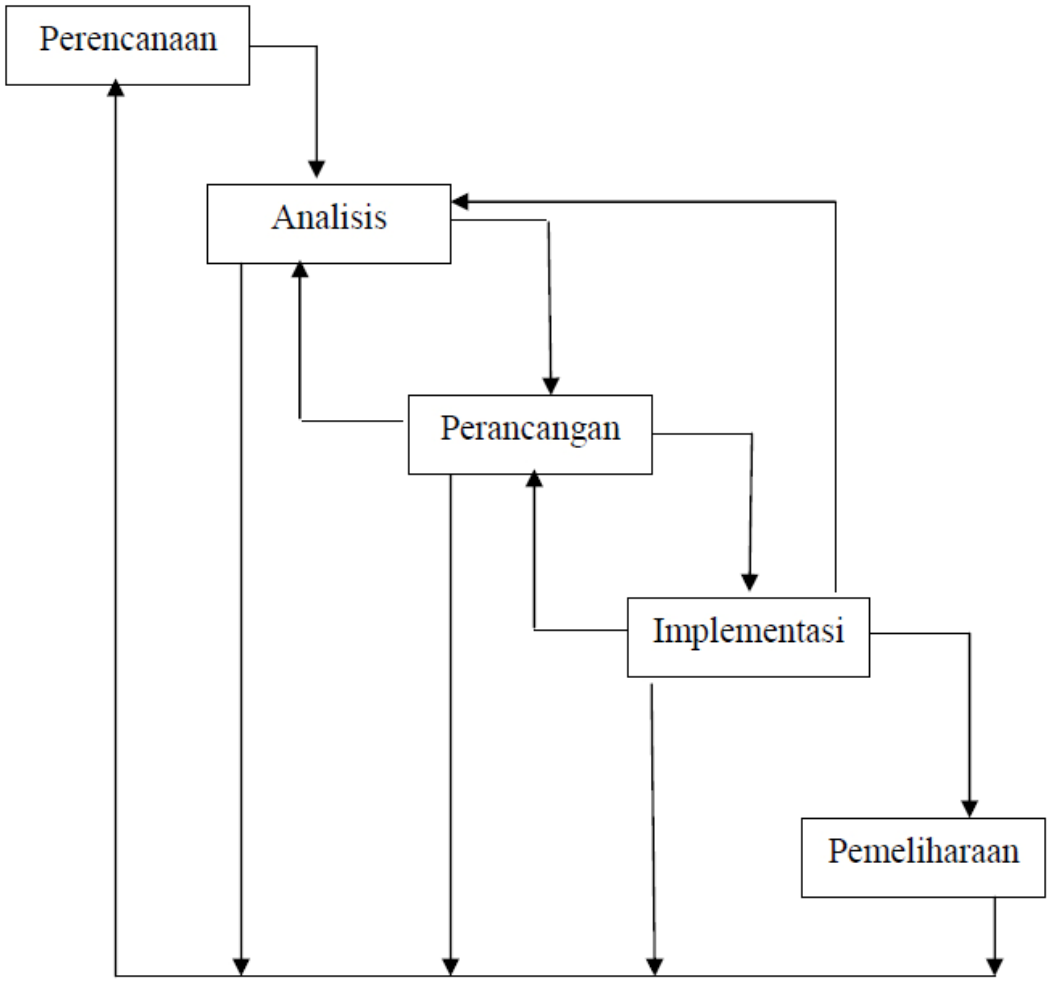
2. Untuk meraih kesempatan-kesempatan *(opportunities)*, teknologi informasi telah berkembang dengan cepatnya. Perangkat keras komputer, perangkat lunak dan teknologi komunikasi telah begitu cepat berkembang. Organisassi mulai merasakan bahwa teknologi informasi ini perlu digunakan untuk meningkatkan pelayanan informasi sehingga dapat mendukung dalam proses pengambilan keputusan yang akan dilakukan oleh pihak manajemen.
3. Adanya instruksi-instruksi *(directives)*, penyusunan sistem yang baru dapat juga terjadi karena adanya instruksi-instruksi dari atas pimpinan ataupun dari luar organisasi.

Karena adanya permasalahan, kesempatan atau instruksi, maka sistem yang baru perlu dikembangkan untuk emecahkan permasalahan-permasalahan yang timbul, meraih kesempatan-kesempatan yang ada atau memenuhi instruksi yang diberikan.



Gambar 2.2. Pengembangan Sistem Informasi

Sistem yang baik adalah sistem yang selalu menyesuaikan dengan perubahan lingkungan yang terjadi disekitarnya atau sistem tersebut harus dinamis menuju pada keadaan yang lebih baik.



Gambar 2.3. Kerangka Kerja Pengembangan Sistem Informasi

Tahap awal, yaitu tahap perencanaan, menyangkut studi kebutuhan *user*, studi kelayakan baik secara teknis maupun teknologi serta penjadwalan pengembangan suatu proyek sistem informasi.

Tahap berikutnya adalah tahap analisis, yaitu tahap dimana kita berusaha mengenali segenap permasalahan yang muncul pada pengguna, mengenali komponen-komponen sistem, obyek-obyek, hubungan antar obyek dan sebagainya.

Kemudian tahap ketiga adalah tahap perancangan, yaitu tahap dimana kita mencoba mencari solusi permasalahan yang didapat dari tahap analisis.

Tahap keempat adalah tahap implementasi dimana kita mengimplementasikan perancangan sistem ke situasi yang nyata. Pada tahap ini. Kita mulai dengan pemilihan perangkat keras, penyusunan perangkat lunak aplikasi (pengkodean atau coding) apakah sistem yang dibuat sudah sesuai dengan kebutuhan *user* atau belum. Jika belum, proses selanjutnya adalah iteratif, yaitu kembali ke tahap-tahap sebelumnya. Disinilah keuntungan metodologi berorientasi obyek mulai terlihat, dimana mulai dari tahap analisis hingga tahap implementasi, kita bisa menggunakan *tools* yang sama sehingga proses iteratif itu dapat berjalan dengan lebih efektif serta efisien ditinjau dari segi uang dan waktu.

Kemudian tahap yang terakhir adalah tahap pemeliharaan atau perawatan, dimana kita bisa mulai melakukan pengoperasian sistem dan jika diperlukan dapat melakukan perbaikan-perbaikan kecil. Kemudian jika waktu penggunaan sistem habis, maka kita akan masuk lagi pada tahap perencanaan.

2.3. Administrasi

Administrasi berasal dari bahasa Latin :Ad = intensif danministrare = melayani, membantu, memenuhi. Administrasi merujuk pada kegiatan atau usaha untuk membantu,melayani, mengarahkan, atau mengatur semua kegiatan di dalam mencapai suatu tujuan.

- Administrasi adalah proses yang pada umumnya terdapat pada semua usaha kelompok, pemerintah atau swasta, sipil atau militer, besar atau kecil (White, 1958).
- Administrasi sebagai kegiatan kelompok yang mengadakan kerjasama guna
- menyelesaikan tugas bersama (Simon, 1958).
- Administrasi didefinisikan sebagai bimbingan, kepemimpinan dan pengawasan
- usaha kelompok individu guna mencapai tujuan bersama (Newman, 1963).
- Pengertian Administrasi dalam bahasa Indonesia ada 2 (dua),Administrasi berasal dari bahasa Belanda, “Administratie” yang merupakan pengertian

Administrasi dalam arti sempit, yaitu sebagai kegiatan tata usaha kantor (catat-mencatat, mengetik, menggandakan, dan sebagainya). Kegiatan ini dalam bahasa Inggris disebut : Clerical works (FX.Soedjadi, 1989).

- Administrasi dalam arti luas, berasal dari bahasa Inggris “Administration” , yaitu proses kerjasama antara dua orang atau lebih berdasarkan rasionalitas tertentu untuk mencapai tujuan bersama yang telah ditentukan (S.P. Siagian, 1973).

Berdasarkan hal tersebut diatas, administrasi ialah proses penyelenggaraan kerja yang dilakukan bersama-sama untuk mencapai tujuan yang telah ditetapkan. Administrasi, baik dalam pengertian luas maupun sempit di dalam penyelenggaraannya diwujudkan melalui fungsi-fungsi manajemen, yang terdiri dari perencanaan, pengorganisasian, pelaksanaan dan pengawasan.

Jadi administrasi adalah penyelenggaraannya, dan manajemen adalah orang-orang yang menyelenggarakan kerja. Maka kombinasi dari keduanya adalah penyelenggaraan kerja yang dilakukan oleh orang-orang secara bersama-sama (kerjasama) untuk mencapai tujuan yang yang telah ditetapkan.

Dalam dunia kerja saat ini istilah administrasi dan manajemen tentu sudah banyak menjadi acuan dalam mencapai suatu tujuan khususnya dalam bidang perkantoran. administrasi berasal dari bahasa yunani yang berarti suatu proses kerja sama antara dua orang atau lebih untuk mencapai suatu tujuan bersama yang telah ditetapkan sebelumya. sedangkan manajemen adalah suatu proses mengantur,mengendalikan dan melaksanakan proses kerjasama untuk mencapai tujuan yang telah ditetapkan. sebagaimana dalam prinsip manajemen adalah "pengorbanan yang sekecil-kecilnya untuk memperoleh hasil yang semaksimal mungkin". Dalam sub bab ini akan dibahas secara singkat tentang pengertian administrasi dan manajemen, unsur-unsur administrasi dan manajemen, peranan adminstrasi dan manajemen dan hubungan administrasi dan manajemen.

2.4. Pengertian Administrasi

Administrasi pendidikan terdiri dari 2 kata yang masing-masing mempunyai pengetian tersendiri. kata administrasi menurut asal katanya berasal dari bahasa latin. ad + ministrae, ad berarti intensif sedangkan ministrae berarti melayani, membantu, dan memenuhi, jadi tugas utama seorang adminisatrator/manajer adalah memberikan layanan prima.

Administrare adalah kata kerja sedangkan kata bendanya adalah administration dan kata sifatnya adalah administratiavus. dalam kegiatan sehari-hari untuk istilah administrasi dibagi menjadi 2 pengertian yaitu:

1. Administrasi dalam pengertian yang sempit adalah suatu pekerjaan tata usaha dalam kantor.
2. Administrasi dalam pengertian yang luas adalah seluruh proses kerja sama orang atau lebih dalam mencapai tujuan bersama.

- **Administrasi** adalah segenap proses penyelenggaraan dalam setiap usaha kerjasama sekelompok manusia untuk mencapai tujuan tertentu.
- **Proses** adalah rangkaian perbuatan manusia yang mengandung maksud tertentu yang memang dikehendaki oleh yang melakukan perbuatan itu. Disini termasuk pula segenap aktivitas yang terjadi sebagai akibat perbuatan itu.
- **Perbuatan** adalah suatu kegiatan manusia yang mengandung maksud tertentu berupa tindakan dan pemikiran.
- **Prosedur** adalah suatu rangkaian metode yang telah menjadi pola tetap dalam melakukan sesuatu pekerjaan yang merupakan suatu kebulatan.
- Suatu rangkaian prosedur yang telah merupakan suatu kebetulan untuk melaksanakan fungsi disebut **system**.
- **Metode** adalah cara yang telah menjadi pasti ketepatannya dan merupakan pola dalam mekukan suatu pekerjaan. Apabila metode merupakan kamahiran dari seseorang yang sudah tergolong ahli, ini disebut teknik.

Jadi administrasi adalah penyelenggaraannya, dan manajemen adalah orang-orang yang menyelenggarakan kerja. Maka kombinasi dari keduanya adalah

penyelenggaraan kerja yang dilakukan oleh orang-orang secara bersama-sama (kerjasama) untuk mencapai tujuan yang yang telah ditetapkan.

1. Administrasi Sebagai Proses Atau Kegiatan Perumusan-perumusan administrasi sebagai kegiatan yang terdapat dalam kepustakaan Indonesia berbunyi sebagai berikut : 1. " jadi apabila bicara mengenai 'administrasi' maka jelas yang dimaksud adalah penyelenggaraan kegiatan-kegiatan untuk mencapai tujuan-tujuan poko dari masyarakat dan anggota-anggotanya"(sumber: Prof. Dr. H. Arifin Abdulrachman Theori, Pengembangan dan Filosofi Kepemimpinan Kerja, 1971, pagina 11)
2. Pengertian Administrasi itu dapat ditinjau dari tiga sudut, yaitu : 1. Administrasi dalam arti Institutionil, yang mana administrasi dimaksudkan sebagai keseluruhan orang/kelompok orang-orang yang sebaga suatu kesatuan menjalankan proses kegiatan-kegiatan untuk mencapai tujuan bersama 2. Administrasi dalam arti fungsionil, yang dimaksud dengan fungsionil ialah segala kegiatan dan tindakan yang dilakukan untuk mencapai tujuan (termasuk juga didalamnya tindakan untuk menenyukan tujuan itu sendiri, atau dengan kata lain bersifat melihat kedepan, artinya melihat kepada pencapaian tujuan pada masa yang akan datang.
3. Administrasi sebagai proses, sebagai proses administrasi berarti keseluruhan proses yang berupa kegiatan-kegiatan, pemikiran-pemikiran, pengaturan-pengaturan sejak dari penentuan tujuan sampai penyelenggaraan sehingga tercapainya suatu tujuan.

Defenisi Administrasi Menurut Para Ahli yaitu:

1. Leonard D white (1955: P.1) merumuskan sebaagai "administration is a process comman to all group effort public or provaate, civil or millitaaary, large scaale or smaall scall" (administrasi adalah suatu proses yang biasanya terdapat pada semua usaha kelompok baik usaha pemerintah, ataupun swasta, sipil atau militer baik secara besar-besaran ataupun kecil-kecilaan).
2. H.A.Simon (1961:P.3) "administration can be defined aas the aactivitiesa if group cooperating to accomplish common goals" (administrasi dapat

didefenisikan sebagai kegiatan kelompok orang-orang yaang melakukan kerjasama untuk mencapai tujuan bersama) .

3. Dwoght Waldo (1971:P.20) "administrasi adalah suaatu bentuk daya upaya manusia yang kooperatif yang mempunyai tingkat rasionalaiteit yaang tinggi".
4. The Lianag Gie (1965:P.5) "administrasi adalah segenap proses penyelenggaraaaan dalam segenap usaha kerjasama sekelompok manusia untuk mencapai tujuan tertentu". defenisi ini mendapat perubahan (1972 :P.a37) yatiu peoses penyelenggaraan diganti dengan rangkaian penaataaan. kemudian lebih disempurnakan (1977:13) yaitu administraasi adalah segenap rangkaian kegiatan terhadap pekerjaan pokok yang dilakukan oleh sekelompok orang dalam kerjasama mencapaai tujuan tertentu.
5. S.p siagiaan (1985:P.3) " adalah keseluruhan proses kerjasama antara dua orang manusia atau lebih yang didasarkan pada rasionalitas tertentu untuk mencapai tujuan yang telah ditentukan sebelumnya.
6. Parajudi Atmosudirjo (11975 : P) administrasi adalah pengendalian dan penggerakk dari suatu organisasi sedemikiaan rupa sehingga organisasi itu menjadi hidup dan bergerak menuju ketercapainya segala sesuatu yang telah ditetapkan oleh administrator yakni kepala organisasi.
7. Sedangkan ilmu administrasi didefinisikan oleh Guilick sebagai berikut: ”The science of administration in thus the system of knowledge whereby men may understand relationship, predict result, and influence outcomes in any situation where men are organized at work together for a common purpose”.

Albert Lepawsky menunjukkan 6 peranan administrasi dalam kehidupan dan perkembangan masyarakat:

1. The Universal Importance of Administration
   - Bahwa administrasi sebagai studi ilmiah yang serius diperlihatkan dari disintegrasi bagi kurangnya abilitas adminstrasi dan efisiensi manajemen terutama pada periode Perang dunia II.
   - Fayol katakan, melalui pengetahuan administrasi kita mengerti peranan orang, komando, koordinasi dan control yang jadi dasar untuk tenaga kerja (workman).

- Dengan demikian tiap orang membutuhkan Adm sehingga adm adalah suatu subyek dari kepentingan universal.

2. The stabilizing role of administration in society
   - Paul pigors menyatakan bahwa fungsi primer administrasi adalah untuk stabilisasi institusi-institusi social.
3. The role administration in social change
   - Brook adam mengatakan bahwa fungsi pokok administrasi adalah jaminan stabilitas social dengan sediakan fasilitas bagi perubahan social.
4. The treat of a managerial revolution
   - James burnham : kualitas esensial dari the emerging society adalah karakter manajemennya, bahwa dalam kenyataan manajer-manajer mempersiapkan untuk taken over masyarakan modern.
5. The prospect of a managerial evolution
   - Charles merriem: bahwa lambat laun akan diberikan atribut posisi manajer dari suatu masyarakat demokratik.
   - Ia menyebut organisasi pemerintahan tidak lain merupakan organisasi manajerial.
6. Adminstration as the key to modern society
   - Charles A. Besrd : Masyarakat modern adalah masayarakat besar. Setiap usaha masyarakat modern akan mendapatkan ketenangan dibawah adminstrasi.

Kedudukan administrasi dalam kehidupan manusia serba kebutuahan, kebutuhaan untuk tempat tinggal, makan, pakaian dan lain-lain yang biasa dikenal dengan istilah kebutuhan- kebutuhan primer atau kebutuhan pokok. untuk dapat memenuhi kebutuhan ini apa itu kebutuhan primer atau kebutuhan sekunder .orang harus mengusahakan dengan perbuatan-perbuatan yang nyata berusaha seorang diri maupuna secara bekerja sama dengan perbuatan yang nyata akan dengan perbuatan nyata maka kebutuhan itu menjelma menjadi tujuan.

Didalam memenuji kebutuhannya dalam banyak hal orang harus bekerja sama atau dengan kata lain orang harus melaksanakan suatu proses penyelenggaraan usaha kerja sama dalam mencapai tujuaannya. proses penyelenggaraan inilah yang disebut dengan administrasi dan pada masyarakat moden yaang makin berkembang ini makin penting pula tujuan-tujuan yang ingin dan hendak dicapainya maka makin baik dan tepat pula administrasi yanag harus diaarahkannya oleh karena itu makin penting pulaa kedudukanaadministrasi sebagai kegiatan-kegiatan untuk mencapai tujuan tersebut.

Kedudukan administrasi penting dalam suatu negara yaang telah maju apalagi bagi negara yang sedang berkembang. yang mana administrasi akan memberi sumbangan pengalaman dibidang apapun telah mengajarka kepada negara-negera itu bahwa masalah kemajuan negara bukanlah dititik beratkan hanya pada modal yang cukup sumber-sumber alam dan kekayaan bumi yaang berlimpah-limpaah tenaga kerja manusia yang belebih-lebihan tetapi sangat dibutuhkan peranan/ kedudukan adminstrasi

Administrasi merupakan modal yang berhargan sekali bagi negara-negara tersebut untuk melaksanakan tugas dalam mencapai tujuan apalagi asas utama administrasi adalah daya guna kerja berarti bahwa manusia ingin mencapai suatu hasil secara maksimum atau terbaik dengan menyelenggarakan sesuatu keja atau usaha secara minimum atau teringan dalam mencapai tujuan yang telah ditentukaan sebelumnya disamping administrasi penting bagi suatu negara maka admnistrasi jugaa penting bagi badan-badaan atau organisasi perusahaaan dan perindustrian juga bagi lembaga-lembaga seperti lembaga peradilan lembaga permasyarakatan bahkan pembrontak dan orang matipun membutuhkan admnistrasi sehingga dapat dikatakan bahwa dimana ada terdapat kegiatan dari orang-orang yang bekerja sama untuk mencapai suatu tujuan bersama maka disana terdapat administrasi.

Bidang administrasi atas 3 golongan besar dengan rincaian sebagai berikut:

- Adminisrasi negara ialah administrasi yang berobyek kenegaraan terdiri dari:
  - Administrasi sipil ialah seluruh kegiatan yang dilakukan oleh departemen, jawatan, kantor kecamatan, dan kantor kelurahan atau seluruh kegiatan negara dikurangi kegiatan perusahaan negara dan kegiatan militer/TNI

- Administrasi kegiatan angkatan bersenjata yang terdiri dari administrasi
- angkatan udara, angkatan laut, angkatan darat dan angkatan kepolisian.

- Administrasi niaga ialah administrasi yang berobyek swasta perniagaan yaitu:
  - Administrasi perusahaan ialah kegiatan-kegiatan dibidang produksi, transportasi, asuransi, perbankan dan lain-lain dibidang perusahaan swasta.
  - Administrsi sosial bukan perusahaan biasanya cenderung kearah usaha sosial seperti administrasi sosial sekolah swasta, rumah sakit swasta, yayasan, klub dll.
- administrasi internasional ialah administrasi yang bergerak dibidang internasional seperti yang dilakukan oleh PBB beserta cabang-cabangnya misalnya UNICEP,ILO,UNESCO, dll.

2.5. Teknologi *Object Oriented*

Teknologi *Object-oriented* merupakan paradigma baru dalam rekayasa *software* yang didasarkan pada obyek dan kelas. Diakui para ahli bahwa *Object-oriented* merupakan metodologi terbaik yang ada saat ini dalam rekayasa *software*. *Object-oriented* memandang *software* bagian per bagian dan menggambarkan satu bagian tersebut dalam satu obyek. Satu obyek dalam sebuah model merupakan suatu fokus selama dalam proses analisis, perancangan dan implementasi dengan menekankan pada *state*, perilaku (*behavior*) dan interaksi obyek dalam model tersebut.

Teknologi obyek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh obyek. Manusia adalah obyek, komputer adalah obyek. Obyek memiliki atribut seperti manusia memiliki nama, pekerjaan, rumah, dan lain-lain. Mobil memiliki warna, merek, sejumlah roda, dan lain-lain. Komputer memiliki kecepatan, sistem operasi, dan lain-lain. Obyek dapat beraksi dan bereaksi. Manusia dapat berjalan, berbicara, makan, minum kemudian mobil dapat berjalan, mengerem dan komputer dapat mengolah data, menampilkan gambar, dan lain-lain.

Keunggulan teknologi obyek dengan demikian adalah bahwa model yang dibuat akan sangat mendekati dunia nyata yang masalahnya akan dipecahkan oleh sistem yang dibangun. Model obyek, atribut dan kelakuan bisa langsung diambil dari obyek yang ada di dunia nyata.

Sistem yang dibangun dengan teknologi obyek memiliki fleksibilitas yang tinggi terhadap perubahan karena menggunakan konsep komponen yang bisa digunakan kembali. Didalam dunia perangkat lunak, penggunaan berulang merupakan hal yang biasa. Contohnya ide dan formula yang hampir sama digunakan berulang oleh *programmer* yang berbeda untuk mengembangkan aplikasi keuangan yang khusus diadaptasikan sesuai kebutuhan dan permintaan masing-masing klien. Oleh karena itu penggunaan berulang suatu obyek merupakan hal yang seharusnya bisa dilakukan. Suatu obyek bisa diambil untuk dimodifikasi berupa penambahan atau pengulangan untuk memecahkan suatu masalah baru.

Ada empat prinsip dasar dari pemrograman berorientasi obyek, yaitu Abstraksi, enkapsulasi, modularitas dan hirarki. Abstraksi memfokuskan perhatian pada karakteristik obyek yang paling penting dan paling dominan yang bisa digunakan untuk membedakan obyek tersebut dari obyek lainnya. Dengan Abstraksi ini developer bisa menerapkan konsep KIS (*Keep It Simple*) pada obyek didunia nyata yang memiliki tingkat kerumitan yang tinggi.

Enkapsulasi menyembunyikan banyak hal yang terdapat dalam obyek yang tidak perlu diketahui oleh obyek lain. Dalam praktek pemrograman, enkapsulasi diwujudkan dengan membuat suatu kelas *interface* yang akan dipanggil oleh obyek lain, sementara didalam obyek yang dipanggil terdapat kelas lain yang mengimplementasikan apa yang terdapat dalam kelas *interface*. Obyek lain hanya tahu dia perlu memanggil kelas *interface* tanpa perlu tahu proses apa saja yang dilakukan didalam kelas implementasinya dan untuk menuntaskan proses tersebut perlu berhubungan dengan obyek mana saja. Dengan demikian bila terjadi proses perubahan pada proses implementasi maka obyek pemanggil tidak akan terpengaruhi secara langsung.

Modularitas membagi sistem yang rumit menjadi bagian-bagian yang lebih kecil yang bisa mempermudah developer memahami dan mengelola obyek tersebut. Contohnya adalah sistem akademis yang bisa dibagi menjdi kemahasiswaan, daftar mata kuliah, pembayaran uang kuliah, dan lain-lain.

Hirarki berhubungan dengan Abstraksi dan modularitas, yaitu pembagian berdasarkan urutan dan pengelompokkan tertentu. Misalnya untuk menentukan obyek mana yang berada pada kelompok yang sama, obyek mana yang merupakan komponen dari obyek yang memiliki hirarki lebih tinggi. Semakin rendah hirarki obyek berarti semakin jauh Abstraksi dilakukan terhadap suatu obyek. Ke empat prinsip dasar ini merupakan hal yang mendasari teknologi obyek yang perlu ditanamkan dalam cara berpikir *developer* berorientasi obyek

.

2.6. *Object Oriented Analysis and Design*

Object-oriented mencakup bidang aplikasi yang sangat luas. Para pengguna sistem komputer dan sistem lain yang didasarkan atas teknologi komputer merasakan efek *Object-oriented* dalam bentuk meningkatnya aplikasi *software* yang mudah digunakan dan servis yang lebih fleksibel, yang muncul dalam berbagai bidang industri, seperti dalam perbankan, telekomunikasi, dan sebagainya. Sedangkan bagi *software engineer*, *Object-oriented* berpengaruh dalam bahasa pemrograman, metodologi rekayasa, manajemen proyek, hardware dan sebagainya.

Analisis dan perancangan berorientasi obyek amat sangat perlu dilakukan dalam pengembangan sistem berorientasi obyek. Hanya dengan kemampuan menggunakan bahasa pemrograman berorientasi obyek yang andal, kita dapat membangun suatu sistem berorientasi obyek, namun sistem aplikasi yang dibangun akan menjadi lebih baik lagi bila langkah awalnya didahului dengan proses analisis dan perancangan berorientasi obyek, terutama untuk membangun sistem yang mudah dipelihara.

Analisis berorientasi obyek adalah metode analisis yang memeriksa *requirements* (syarat atau keperluan yang harus dipenuhi suatu sistem) dari sudut pandang kelas-kelas dan obyek-obyek yang ditemui dalam ruang lingkup permasalahan. Sedangkan perancangan berorientasi obyek adalah metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi obyek-obyek sistem atau subsistem.

2.6.1. Konsep Dasar dalam *Object Oriented Analysis and Design*

2.6.1.1. Obyek

Obyek adalah benda secara fisik atau konseptual, yang dapat kita temui disekeliling kita. *Hardware*, *software*, dokumen atau manusia dan bahkan konsep semuanya adalah contoh obyek. Untuk kepentingan memodelkan perusahaannya, seorang kepala eksekutif akan melihat gedung, karyawan, divisi, dokumen dan keuntungan sebagai obyek. Seorang teknisi mesin akan melihat ban, pintu, mesin, laju tertinggi dan banyaknya bahan bakar sebagai sebuah obyek. Dan seorang *software engineer* akan memandang tumpukan, antrian, jendela dan *check box* sebagai sebuah obyek.

Sebuah obyek memiliki keadaan *(state)* dan perilaku *(behavior). State* dari sebuah obyek adalah kondisi obyek tersebut atau himpunan dari keadaan yang menggambarkan obyek tersebut. Sebagai contoh, bola lampu adalah obyek, dan salah satu keadaan nyala atau tidak nyala adalah *state* dari obyek bola lampu tersebut.

State dinyatakan dengan nilai dari sebuah atribut obyeknya. Atribut adalah nilai internal suatu obyek yang mencerminkan antara lain karakteristik obyek, kondisi obyek, kondisi sesaat, koneksi dengan obyek lain dan identitas. Perubahan *state* dicerminkan oleh perilaku (*behavior*) obyek tersebut.

Behavior suatu obyek mendefinisikan bagaimana sebuah obyek bertindak (beraksi) dan memberikan reaksi. *Behavior* ditentukan oleh himpunan semua atau beberapa operation yang dapat dilakukan dalam obyek itu sendiri. *Behavior* dari sebuah obyek dicerminkan oleh *interface, service dan method* dari obyek tersebut.

Interface adalah pintu untuk mengakses service obyek. *Service* adalah fungsi yang bisa diemban obyek. *Method* adalah mekanisme internal obyek yang mencerminkan perilaku (*behavior*) obyek tersebut. Sebagai contoh, jika printer merupakan sebuah obyek maka perilaku (*behavior*) atau servisnya mencetak apapun yang dia terima.

2.6.1.2 Kelas (*Class*)

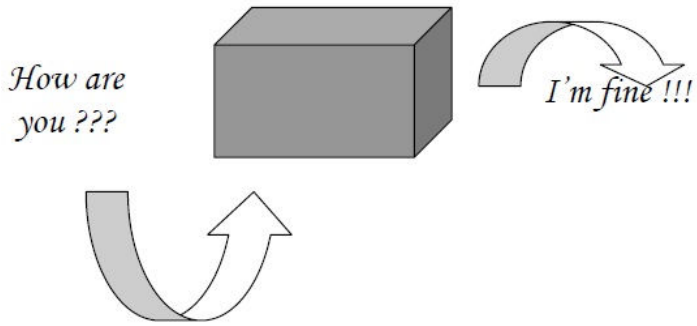
Kelas *(Class)* adalah definisi umum (pola, *template* atau cetak biru) untuk himpunan obyek sejenis. Kelas menetapkan spesifikasi perilaku (*behavior*s) dan atribut obyek-obyek tersebut. *Class* adalah Abstraksi dari entitas dalam dunia nyata. Obyek adalah contoh dari sebuah kelas. Sebagai contoh, atribut untuk kelas binatang adalah berkaki empat dan memiliki ekor. Perilakunya adalah makan dan tidur. Contoh yang mungkin dari kelas binatang ini adalah kucing, gajah dan kuda.

2.6.1.3 Kotak Hitam dan *Interface*



Gambar 2.4. Sebuah Obyek digambarkan sebagai Kotak Hitam

Encapsulation adalah proses menyembunyikan detil implementasi sebuah obyek. Satu-satunya jalan untuk mengakses data obyek tersebut adalah melalui *interface*. *Interface* melindungi internal *state* sebuah obyek dari campur tangan pihak luar. Oleh karena itu obyek sering digambarkan sebagai kotak hitam (*black box*) yang menerima dan mengirim pesan-pesan (*message*). Dalam *Object-oriented* programming kotak hitam tersebut berisi kode (himpunan instruksi dengan bahasa yang dipahami komputer) dan data (informasi dimana instruksi tersebut beroperasi dengannya). Dalam *Object-oriented programming*, kode dan data disatukan dalam sebuah benda yang tersembunyi isinya yaitu obyek. Pengguna obyek tidak perlu mengetahui isi dalam kotak tersebut. Untuk dapat berkomunikasi dengan obyek, diperlukan pesan (*message*).



Gambar 2.5. *User* Memakai Pesan dalam Menggunakan Obyek

Secara formal kita mendefinisikan *message* sebagai permintaan untuk obyek penerima (*receiver object*) untuk membawa metode yang ditunjukkan atau perilaku dan mengembalikan *result* dari aksi tersebut kepada obyek pengirim (*sender object*). Sebagai contoh, satu obyek manusia mengirim obyek bola lampu sebuah pesan untuk menyala (melalui saklar). Obyek bola lampu memiliki satu perilaku yang akan mengubah keadaan atau *state* nya dari padam menjadi menyala. Obyek bola lampu menyalakan dirinya dan menunjukkan kepada obyek orang tersebut bahwa *state* barunya adalah menyala.

2.6.1.4. *Association dan Aggregation*

Association (asosiasi) adalah hubungan antar obyek yang saling membutuhkan. Sedangkan *aggregation* (agregasi) adalah bentuk khusus dari asosiasi yang menggambarkan seluruh bagian suatu obyek merupakan bagian dari obyek lainnya. Sebagai contoh, obyek tanggal dapat disusun dari obyek hari, obyek bulan dan obyek tahun.

2.6.2. Keunggulan *Object Oriented* dalam Banyak Kasus

Sekarang ini terdapat beberapa paradigma yang digunakan dalam rekayasa *software*, diantaranya *procedure-oriented*, *Object-oriented*, data *structure-oriented*, data *flow-oriented* dan *constraint oriented*. Tiap-tiap paradigma tersebut cocok untuk beberapa kasus dan bagian dari seluruh kemungkinan ruang lingkup aplikasi, tetapi menurut Booch berdasarkan pengalamannya, *Object-oriented* dapat diaplikasikan dalam seluruh ruang lingkup.

Untuk memahami mengapa OOAD unggul dalam banyak kasus, harus memahami masalah yang dihadapi perusahaan rekayasa *software*. Pertama, *software* sulit untuk dimodifikasi bila memerlukan pengembangan. Kedua, proses pembuatan *software* memerlukan waktu yang cukup lama sehingga kadang kala melebihi anggaran dalam pembuatannya. Ketiga, para *programmer* selalu membuat *software* dari dasar karena tidak adanya kode yang bisa digunakan ulang (*reuse*).

Kebanyakan perusahaan tersebut sebelumnya telah menggunakan *pemrograman structural*. Metode *structural* menggunakan pendekatan dengan pendekatan *top-down*, yang mana pendekatan ini memecahkan masalah dengan membagi masalah kedalam komponen-komponen hingga didapatkan komponen yang tidak dapat dibagi-bagi lagi. Pendekatan dengan cara *top-down* ini telah membuat peningkatan dalam kualitas *software*, tetapi dalam sistem yang berskala besar, pendekatan ini sering menemui banyak masalah. Pemrograman *structural* seringkali tidak dapat mendesain apa yang akan terjadi dalam sistem yang telah selesai tanpa mengimplementasikan sistem terlebih dahulu. Jika ditemui kesalahan dalam sistem, maka desain harus disusun ulang dari awal sampai akhir. Hal ini akan terjadi pemborosan biaya dan waktu.

Perbedaan antara metode *structural* dan OOAD terletak pada bagaimana data dan fungsi disimpan. Dalam metode *structural*, data dan fungsi disimpan terpisah. Biasanya semua data ditempatkan sebelum fungsi ditulis. Seringkali tidak terpikirkan sebelumnya untuk mengetahui data mana yang digunakan dalam suatu fungsi tertentu. Tetapi dalam OOAD, data dan fungsi yang berhubungan dalam suatu obyek disimpan bersama-sama dalam satu kesatuan.

Orang akan lebih mudah memahami sistem sebagai obyek daripada prosedur, karena biasanya orang berpikir dalam bentuk obyek. Sebagai contoh, orang melihat mobil sebagai sebuah sistem yang memiliki mesin, roda , stir, tank bahan bakar dan sebagainya. Kebanyakan orang tidak melihat mobil sebagai sebuah urutan prosedur yang membuat mobil tersebut dapat berjalan. Karena secara

alamiah lebih mudah memikirkan suatu sistem dalam bentuk obyek-obyek, tidak heran kalau OOAD menjadi lebih terkenal.

Satu alasan mengapa *Object Oriented* menguntungkan bagi *programmer* adalah karena *programmer* dapat mendesain program dalam bentuk obyek-obyek dan hubungan antar obyek tersebut untuk kemudian dimodelkan dalam sistem nyata. Keuntungan yang lain adalah proses pembuatan *software* dapat dilakukan dengan lebih cepat karena *software* dibangun dari obyek-obyek standar, dapat menggunakan ulang model yang ada, dan dapat membuat model yang cepat melalui metodologi. Kualitas yang tinggi dari *software* dapat dicapai karena adanya tested *component*. Lebih mudah dalam *maintenance* karena perbaikan kode hanya diperlukan pada satu tempat (bukan diurut dari awal). Mudah dalam membangun sistem yang besar karena subsistem dapat dibuat dan diuji secara terpisah. Mengubah sistem yang sudah ada tidak memerlukan membangun ulang keseluruhan sistem.

2.6.3. Pemrograman Berorientasi Object

Pemrograman berorientasi obyek merupakan kelanjutan dari proses analisis dan desain berorientasi obyek. Dalam pemrograman berorientasi obyek ini, komponen yang didesain dalam proses desain kemudian diimplementasikan dengan menggunakan bahasa pemrograman berorientasi obyek. Syarat sebuah bahasa pemrograman berorientasi obyek adalah bila bahasa pemrograman tersebut memiliki fitur untuk mengimplementasikan ke 4 konsep berorientasi obyek, yaitu Abstraksi, *encapsulation*, *polymorphism* dan *inheritance*. Dikenal beberapa bahasa pemrograman berorientasi obyek, seperti C++, Java, Visual Basic, dan sebagainya.

2.7. *Unified Modelling Language* (UML)

2.7.1. Sejarah Singkat *Unified Modelling Language* (UML)



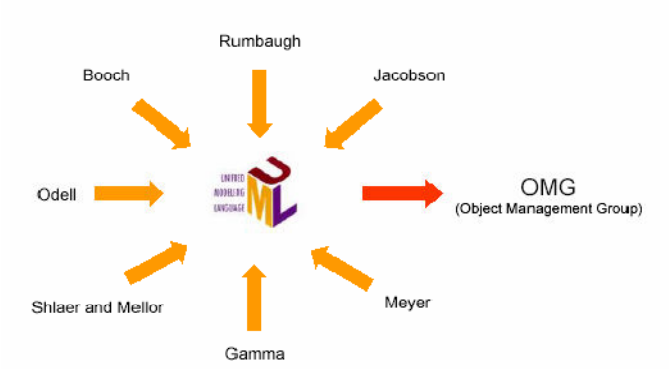
Gambar 2.6. Logo *Unified Modelling Language*

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk menentukan, visualisasi, merancang dan mendokumentasikan *Artifact* dari sistem *software*, untuk memodelkan bisnis dan sistem non *software* lainnya. UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *Class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

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

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah, metodologi Booch, metodologi Coad , metodologi OOSE, metodologi OMT, metodologi Shlaer-Mellor, metodologi Wirfs-Brock, dan sebagainya. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerja sama dengan group atau perusahaan lain yang menggunakan metodologi yang berlainan.



Gambar 2.7. Metodologi dalam UML

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikatakan metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http: atau atau www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

Object Management Group, Inc. (OMG) adalah sebuah organisasi international yang dibentuk pada 1989, didukung lebih dari 800 anggota, terdiri dari

perusahaan sistem informasi, *software* developer, dan pada *user* sistem komputer. organisasi ini salah satunya bertugas membuat spesifikasi "manajemen objek" untuk menetapkan kerangka bersama dalam rekayasa *software*.

Sasaran OMG adalah membantu perkembangan *Object-oriented technology* dan mengarahkannya dengan mendirikan *Object Management Architecture* (OMA). OMA menentukan infrastruktur konseptual yang didasarkan pada seluruh spesifikasi yang dikeluarkan OMG.

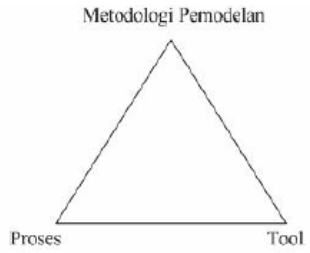
OMG kemudian mengeluarkan UML, dimana dengan adanya UML ini diharapkan dapat mengurangi kekacauan dalam bahasa pemodelan yang selama ini terjadi dalam lingkungan industri. UML diharapkan juga dapat menjawab masalah penotasian dan mekanisme tukar menukar model yang terjadi selama ini.

2.7.2. Tinjauan Mengenai UML

Saat ini piranti lunak semakin luas dan besar lingkupnya, sehingga tidak bisa lagi dibuat asal-asalan. Piranti lunak saat ini seharusnya dirancang dengan memperhatikan hal-hal seperti *scalability*, *security* dan eksekusi yang *robust* walaupun dalam kondisi yang sulit. Selain itu arsitekturnya harus didefinisikan dengan jelas, agar bug mudah ditemukan dan diperbaiki, bahkan oleh orang lain selain *programmer* aslinya. Keuntungan lain dari perencanaan arsitektur yang matang adalah dimungkinkannya penggunaan kembali modul atau komponen untuk aplikasi piranti lunak lain yang membutuhkan fungsionalitas yang sama.

Pemodelan (*modeling*) adalah proses merancang piranti lunak sebelum melakukan pengkodean (*coding*). Model piranti lunak dapat dianalogikan seperti pembuatan blueprint pada pembangunan gedung. Membuat model dari sebuah sistem yang kompleks sangatlah penting karena kita tidak dapat memahami sistem semacam itu secara menyeluruh. Semakin komplek sebuah sistem, semakin penting pula penggunaan teknik pemodelan yang baik.

Dengan menggunakan model, diharapkan pengembangan piranti lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat, termasuk faktor-faktor seperti scalability, robustness, security, dan sebagainya. Kesuksesan suatu pemodelan piranti lunak ditentukan oleh tiga unsur, yang kemudian terkenal dengan sebutan segitiga sukses (*the triangle for success*). Ketiga unsur tersebut adalah metode pemodelan (*notation*), proses (*process*) dan *tool* yang digunakan. Memahami notasi pemodelan tanpa mengetahui cara pemakaian yang sebenarnya (proses) akan membuat proyek gagal. Dan pemahaman terhadap metode pemodelan dan proses disempurnakan dengan penggunaan tool yang tepat.



Gambar 2.8 Kesuksesan suatu Model Piranti Lunak

2.7.2.1. *Artifact* UML

UML menyediakan beberapa notasi dan *Artifact* standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. *Artifact* didalam UML didefinisikan sebagai informasi dalam bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat. Contohnya adalah *source code* yang dihasilkan oleh proses pemrograman.

Yang harus diperhatikan untuk menjaga konsistensi antar *Artifact* selama proses analisis dan desain adalah bahwa setiap perubahan yang terjadi pada satu *Artifact* harus juga dilakukan pada atifact sebelumnya. Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut:

- *Use Case* diagram
- *Class* diagram
- *behavior* diagram
  - *statechart* diagram
  - *activity* diagram

- *Interaction* diagram
  - o *sequence* diagram
  - o c*ollaboration* diagram

❖ *implementation* diagram
- *component* diagram
- *deployment* diagram

Diagram-diagram tersebut diberi nama berdasarkan sudut pandang yang berbeda-beda terhadap sistem dalam proses analisis atau rekayasa. Dibuatnya berbagai jenis diagram diatas karena:

- Setiap sistem yang kompleks selalu paling baik jika didekati melalui himpunan berbagai sudut pandang yang kecil yang satu sama lain hampir saling bebas *(independent).* Sudut pandang tunggal senantiasa tidak mencukupi untuk melihat sistem yang besar dan kompleks.
- Diagram yang berbeda-beda tersebut dapat menyatakan tingkatan yang berbeda-beda dalam proses rekayasa.
- Diagram-diagram tersebut dibuat agar model yang dibuat semakin mendekati realitas.

2.7.2.2. Semantik dalam UML

OMG telah menetapkan semantik (makna istilah) semua notasi UML dalam model struktural dan model *behavior*. Model struktural (model statis), menekankan stuktur obyek dalam sebuah sistem, menyangkut kelas-kelas, *interface*, atribut dan hubungan antar komponen. Model *behavioral* (model dinamis), menekankan perilaku obyek dalam sebuah sistem, termasuk metode, interaksi, kolaborasi dan *state history*.

2.7.2.3. Tujuan UML

Tujuan utama UML diantaranya untuk:

➢ Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.

- Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
- Menyatukan praktek-praktek terbaik yang terdapat dalam bahasa pemodelan.

2.7.2.4. Cakupan UML

Pertama, UML menggabungkan konsep Booch, OMT dan OOSE, sehingga UML merupakan suatu bahasa pemodelan tunggal yang umum dan digunakan secara luas oleh para *user* ketiga metode tersebut dan bahkan para *user* metode lainnya.

Kedua, UML menekankan pada apa yang dapat dikerjakan dengan metode-metode tersebut.

Ketiga, UML berfokus pada suatu bahasa pemodelan standar, bahkan pada proses standar. Meskipun UML harus diaplikasikan dalam konteks sebuah proses, dari pengalaman, bahwa organisasi dan masalah yang berbeda juga memerlukan proses yang berbeda pula. UML tidak mencakup:

- Bahasa Pemrograman *UML* adalah bahasa pemodelan visual, bukan dimaksudkan untuk menjadi suatu bahasa pemrograman visual, tetapi UML memberikan arah untuk bergerak kearah kode.
- *Tool* (*software* aplikasi) pemodelan

  Membuat standar sebuah bahasa diperlukan oleh tool-tool dan proses. UML mendefinisikan semantik dan notasi, bukan sebuah tool. Contoh *tool* yang menggunakan UML sebagai bahasanya adalah *Rational Rose dan Enterprise Architect.*
- Proses rekayasa

  UML digunakan sebagai bahasa dalam proyek dengan proses yang berbeda-beda. UML bebas dari proses dan mendefinisikan sebuah proses standar bukan tujuan UML atau RFP dari OMG. Dalam pembahasan ini kita akan menggunakan sebuah proses yang dikeluarkan *Rational Software, yaitu Rational Unified Process (RUP)*

2.8. Notasi dalam UML

2.8.1. *Actor*



Gambar 2.9. Notasi *Actor*

Actor menggambarkan segala pengguna *software* aplikasi (*user*). *Actor* memberikan suatu gambaran jelas tentang apa yang harus dikerjakan *software* aplikasi. Sebagai contoh sebuah *Actor* dapat memberikan *input* kedalam dan menerima informasi dari *software* aplikasi, perlu dicatat bahwa sebuah *Actor* berinteraksi dengan use case, tetapi tidak memiliki kontrol atas use case. Sebuah *Actor* mungkin seorang manusia, satu *device*, *hardware* atau sistem informasi lainnya.

2.8.2. *Use Case*



Gambar 2.10. Notasi *Use Case*

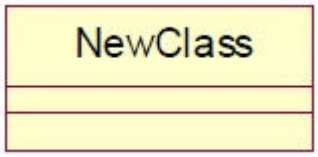
Use Case menjelaskan urutan kegiatan yang dilakukan *Actor* dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, namun *Use Case* hanya menjelaskan apa yang dilakukan oleh *Actor* dan sistem bukan bagaimana *Actor* dan sistem melakukan kegiatan tersebut.

- *Use-case Konkret* adalah *Use Case* yang dibuat langsung karena keperluan *Actor*. *Actor* dapat melihat dan berinisiatif terhadapnya

- *Use-case Abstrak* adalah *Use Case* yang tidak pernah berdiri sendiri. *Use Case* abstrak senantiasa termasuk didalam (*include*), diperluas dari (*extend*) atau memperumum (*generalize*) *Use Case* lainnya.

Untuk menggambarkannya dalam *Use Case* model biasanya digunakan *Association relationship* yang memiliki *stereotype include*, *extend* atau *Generalization relationship*. Hubungan *include* menggambarkan bahwa suatu *Use Case* seluruhnya meliputi fungsionalitas dari *Use Case* lainnya. Hubungan *extend* antar *Use Case* berarti bahwa satu *Use Case* merupakan tambahan fungsionalitas dari *Use Case* yang lain jika kondisi atau syarat tertentu terpenuhi.

2.8.3. *Class*



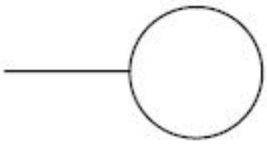
Gambar 2.11. Notasi *Class*

Class merupakan pembentuk utama dari sistem berorientasi obyek, karena *Class* menunjukkan kumpulan obyek yang memiliki atribut dan operasi yang sama. *Class* digunakan untuk mengimplementasikan *interface*.

Class digunakan untuk mengabstraksikan elemen-elemen dari sistem yang sedang dibangun. *Class* bisa merepresentasikan baik perangkat lunak maupun perangkat keras, baik konsep maupun benda nyata. Notasi *Class* berbentuk persegi panjang berisi 3 bagian, persegi panjang paling atas untuk *nama Class*, persegi panjang paling bawah untuk *operasi,* dan persegi panjang ditengah untuk atribut.

Atribut digunakan untuk menyimpan informasi. Nama atribut menggunakan kata benda yang bisa dengan jelas merepresentasikan informasi yang tersimpan didalamnya. Operasi menunjukkan sesuatu yang bisa dilakukan oleh obyek dan menggunakan kata kerja.

2.8.4. *Interface*



Gambar 2.12. Notasi *Interface*

Interface merupakan kumpulan operasi tanpa implementasi dari suatu *Class*. Implementasi operasi dalam *interface* dijabarkan oleh operasi didalam *Class*. Oleh karena itu keberadaan *interface* selalu disertai oleh *Class* yang mengimplementasikan operasinya. *Interface* ini merupakan salah satu cara mewujudkan prinsip enkapsulasi dalam obyek.

2.8.5. *Interaction*



Gambar 2.13. Notasi *Interaction*

Interaction digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek. Biasanya *Interaction* ini dilengkapi juga dengan teks bernama operation signature yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan.

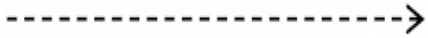
2.8.6. *Note*



Gambar 2.14. Notasi *Note*

Note digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa disertakan ke semua elemen notasi yang lain.

2.8.7. *Dependency*



Gambar 2.15. Notasi *Dependency*

Dependency merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada dibagian tanpa tanda panah.

Terdapat 2 *stereotype* dari *Dependency*, yaitu *include* dan *extend*. *Include* menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah).
Extend menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan kedalam elemen yang ada di garis dengan panah.

2.8.8. *Association*

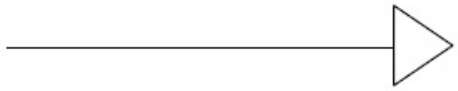
Gambar 2.16. Notasi *Association*

Association menggambarkan navigasi antar *Class* (*navigation*), berapa banyak obyek lain yang bisa berhubungan dengan satu obyek (*multiplicity* antar *Class*) dan apakah suatu *Class* menjadi bagian dari *Class* lainnya (*aggregation*).

Navigation dilambangkan dengan penambahan tanda panah di akhir garis. *Bidirectional navigation* menunjukkan bahwa dengan mengetahui salah satu *Class* bisa didapatkan informasi dari *Class* lainnya. Sementara *UniDirectional navigation* hanya dengan mengetahui *Class* diujung garis *Association* tanpa panah kita bisa mendapatkan informasi dari *Class* di ujung dengan panah, tetapi tidak sebaliknya.

Aggregation mengacu pada hubungan “has-a”, yaitu bahwa suatu *Class* memiliki *Class* lain, misalnya Rumah memiliki *Class* Kamar.

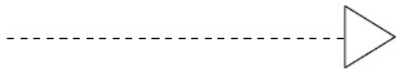
2.8.9. *Generalization*



Gambar 2.17. Notasi *Generalization*

Generalization menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan *Generalization*, *Class* yang lebih spesifik (sub*Class*) akan menurunkan atribut dan operasi dari *Class* yang lebih umum (super*Class*) atau *“subClass is superClass”*. Dengan menggunakan notasi *Generalization* ini, konsep inheritance dari prinsip hirarki dapat dimodelkan.

2.8.10. *Realization*



Gambar 2.18. Notasi *Realization*

Realization menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah. Misalnya *Class* merealisasikan *package*, *component* merealisasikan *Class* atau *interface*.

2.9. Pertanyaan

2.9.1. Siapa Yang Menggunakan Aplikasi Ini?

Aplikasi ini diperuntukkan untuk siapa, aplikasi harus bisa mengerti secara jelas dan organisasi yang mana untuk mensponsori sistem yang baru. 2 macam tipe sponsor yang paling penting adalah sponsor keuangan dan pengguna.

Sponsor keuangan sangat penting karena mereka membayar sistem barunya. Mereka berharap proyek akan selesai sesuai jadwal dan sesuai dengan modal. Aplikasi seharusnya mendapatkan sponsor keuangan untuk setuju kepada tindakan menuju sukses. Dimana aplikasi harus tau ketika sistem selesai dan sesuai dengan keinginan mereka.

Pengguna adalah sponsor juga, tetapi dalam artian lain. Pengguna pada akhirnya penentu kesuksesan sistem yang baru dilihat dari bertambah atau berkurang peoduktivitas atau efektivitas mereka pengguna dapat menolong kamu jika mereka cepat mengerti dan membuat komentar mengkritik. Mereka dapat meningkatkan sistem kamu dengan mengatakan kepada kamu apa yang hilang dan apa yang dapat ditingkatkan.

Pada umumnya, pengguna tidak akan mempertimbangkan *software* yang baru setidaknya mereka memiliki keinginan mengkompailnya-baik perorangan atau bisnis. Kita seharusnya mencoba untuk menolong mereka menemukan kepentingan pribadi dalam proyek kamu jadi kamu dapat menghasilkan apa yang mereka beli.

2.9.2. Apa Masalah Yang Dapat Diselesaikan?

Masalah apa yang akan diselesaikan? aplikasi harus secara jelas membatasi besar usaha untuk sistem yang baru dan menyusun cakupannya. aplikasi seharusnya menentukan fitur yang mana akan menjadi sistem yang baru dan fitur yang mana yang tidak. aplikasi seharusnya mendapatkan berbagai macam pengguna pada organisasi berbeda dengan pandangan mereka sendiri. aplikasi seharusnya tidak hanya memutuskan fitur yang mana yang sesuai, tetapi kamu juga harus memperoleh persetujuan dari orang yang berpengaruh.

2.9.3. Dimana Sistem Akan Digunakan?

Dimana sistem akan digunakan? Pada tahap awal, sistem sangat menolong untuk mendapat ide umum dimana sistem akan digunakan. aplikasi seharusnya

menentukan jika sistem yang baru adalah *software* misi kritis untuk organisasi, *software* experiment, atau kemampuan baru yang dapat kita sebarkan tanpa mengganggu alur kerja. aplikasi seharusnya memiliki ide yang sukar tentang bagaimana sistem yang baru akan melengkapi sistem yang sudah ada. Sangat penting untuk mengetahui jika *software* akan digunakan secara local atau akan didistribusikan melalui jaringan.

2.9.4. Kapan Aplikasi Dibutuhkan?

Kapan aplikasi dibutuhkan? Dua aspek waktu sangatlah penting. Yang pertama adalah waktu yang dimungkinkan, waktu pada sistem yang mana dapat dikembangkan dalam keterbatasan biaya dan sumber yang ada. Yang lain adalah waktu yang dibutuhkan, ketika sistem dibutuhkan untuk mempertemukan tujuan bisnis.

Aplikasi harus secara yakin bahwa waktu yang diharapkan didorong oleh kelayakan teknis adalah konsisten dengan waktu yang dibutuhkan organisasi. Jika ada pemutusan, aplikasi harus berinisiatif berdialog antara ahli teknologi dan ahli bisnis untuk mencapai solusi.

2.9.5. Kenapa Aplikasi Dibutuhkan?

Kenapa aplikasi dibutuhkan? Aplikasi diharuskan bersiap sebuah kasus bisnis untuk sistem yang baru jika seseorang belum melakukannya. Kasus bisnis mengandung pembenaran keuangan untuk sistem yang baru, termasuk biaya, keuntungan yang nyata, keuntungan yang tak nyata, resiko, dan alternative utama. Aplikasi seharusnya mempertimbangkan biaya pengembangan, pembelian, dan penyebaran.

Aplikasi harus yakin bahwa aplikasi secara jelas mengerti motivasi bisnis untuk sistem yang baru. Kasus bisnis akan memberimu pandangan ke dalam apa yang diharapkan sponsor, secara jelas mengidentifikasi ruang lingkup dan juga dapat menyediakan informasi untuk, mencoba model.

2.9.6. Bagaimana Aplikasi Akan Bekerja?

Bagaimana sistem ini akan bekerja? Aplikasi seharusnya mendapat ilham tentang kemungkinan dari masalah. Untuk sistem yang besar, aplikasi harus mempertimbangkan kegunaan dari berbagai arsitektur, keuntungan dari spekulasi ini adalah tidak untuk memilih sebuah solusi, tapi untuk meningkatkan percaya diri dapat diselesaikan secara beralasan.

Aplikasi mungkin membutuhkan beberapa *prototype* dan percobaan. Aplikasi dapat menyelamatkan kode data atau ide dari sistem yang lalu. Biasanya putusan yang terbaik adalah membeli *software*, jika Kita dapat menemukan *software* yang sangat memenuhi persyaratan.

2.10. Cara Menghubungkan Visual Basic dengan Database Access

Ini adalah tentang cara untuk terhubung ke database dan memanipulasi database menggunakan VB. Ini akan bekerja (dengan beberapa perubahan kecil) dalam VBA juga. Ada beberapa cara untuk terhubung ke database (misalnya, akses), melalui kontrol data link, DAO atau ADO. Secara umum saya tidak menggunakan data kontrol terikat karena aku ingin tetap mengontrol apa yang terjadi pada data, sehingga sisanya sekitar DAO / ADO. Pertama, Anda perlu menciptakan beberapa variabel-variabel berikut:

- ADODB.Connection Workspace - Ini diperlukan jika anda menggunakan Proses Transaksi.
- database ini terhubung ke database recordset.
- recordset adodb ini adalah query tabel variabel tingkat.
- ADODB.Field Fields - Ini memungkinkan kita untuk mendapatkan informasi tentang bahan baku.

Dengan database Access, Anda dapat terhubung ke database dalam 2 cara, JET atau ODBC. Secara pribadi saya menggunakan ODBC untuk manajemen file lebih mudah, untuk mengubah nama file atau path hanya menggunakan panel kontrol ODBC Administrator. Catatan: Contoh-contoh di sini untuk menunjukkan

apa yang harus dilakukan dengan sangat sederhana, beberapa perintah memiliki opsi lebih dari terdaftar, silahkan periksa file bantuan untuk lebih jelasnya.

DAO example - JET Connection

```
Dim ws as Workspace
Dim db as Database

Set ws=DBEngine.Workspaces(0)
set db=ws.OpenDatabase({databasepath and name})
```

DAO example - ODBC Connection

```
Dim ws as Workspace
dim db as database
dim strConnection as string

set ws=DBEngine.Workspaces(0)
let strConnection= "ODBC;DSN=" & DatabaseName & ";UID=" & UserName
& ";PWD=" & UserPassword
set db=ws.OpenDatabase("", False, False, strConnection)
```

ADO Example

```
Dim ad as ADODB.Connection

set ad=New ADODB.Connection
Let ad.ConnectionString= "ODBC;DSN=" & DatabaseName & ";UID=" &
UserName & ";PWD=" & UserPassword
ad.Open
```

2.11. Paradigma Database Manajemen

Tabel 2.1. Paradigma Database Manajemen

DBMS paradigm	Rekomendasi aplikasi
Hierarchical DBMS	• Terlalu tua untuk aplikasi yang baru, mungkin ditemui di aplikasi yang sudah langka.
Network DBMS	

Relational DBMS	• Aplikasi Mendukung keputusan • Merupakan Aplikasi bisnis • Aplikasi ini dapat menggunakan bahasa generasi ke empat • Aplikasi yang terintegrasi dengan aplikasi RDBMS yang sudah langka • Implementasi konservatif
OO-DBMS	• Rekayasa disain aplikasi • Multimedia aplikasi • Pengetahuan basis • Aplikasi memerluka distribusi dan bersamaan • Aplikasi yang memerlukan fitu-fitur yang canggih • Perangkat elektronik dengan perangkat lunak tertanam
Real-time DBMS	• Aplikasi untuk yang data harus benar dan tepat waktu
Deductive DBMS	• Aplikasi yang menggabungkan manajemen data dan kesimpulan
Active DBMS	• Aplikasi yang menggabungkan manajemen data dengan perilaku berkembang

2.12. Rekomendasi Pengendalian Eksternal

Tabel 2.2. Rekomendasi Pengendalian Eksternal

Pengendalian External	Rekomendasi
Procedure-driven	Disukai untuk sebahagian besar aplikasi interface pengguna kurang substansial
Event-driven	Pilihan untuk aplikasi yang membutuhkan user interface fleksibel
Concurrent	Jarang digunakan dalam aplikasi yang sering digunakan seluruh aplikasi
Declarative	Kadang-kadang layak dipertimbangkan untuk sistem basis aturan