SISTEM INFORMASI MANAJEMEN PENDIDIKAN

IKAPI IKATAN PENERBIT INDONESIA
Appti
UNY PRESS

Lantip Diat Prasojo

SISTEM INFORMASI MANAJEMEN PENDIDIKAN

Lantip Diat Prasojo

Sanksi Pelanggaran Pasal 44:
Undang-undang Nomor 7 Tahun 1987 Tentang
Perubahan atas Undang-undang Nomor 6 Tahun 1982
Tentang Hak Cipta

1. Barangsiapa dengan sengaja dan tanpa hak mengumumkan atau memperbanyak suatu ciptaan atau memberi izn untuk itu, dipidana dengan pidana penjara paling lama 7 (tujuh) tahun dan/ atau denda paling banyak Rp. 100.000.000,00 (seratus juta rupiah)

2. Barangsiapa dengan sengaja menyerahkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil Pelanggaran Hak Cipta sebagaimana dimaksud dalam ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/ atau denda paling banyak Rp. 50.000.000,00 (lima puluh juta rupiah)

SISTEM INFORMASI MANAJEMEN

Lantip Diat Prasojo

UNY PRESS

2013

SISTEM INFORMASI MANAJEMEN PENDIDIKAN

Oleh: Lantip Diat Prasojo

Edisi Pertama,

Cetakan Pertama: Maret 2013, **ISBN: 978-602-7981-01-0**

Diterbitkan dan dicetak oleh:
UNY Press
Jl. Gejayan, Gg Alamanda, Komplek Fakultas Teknik UNY
Kampus UNY Karangmalang Yogyakarta 55281 Telp: 0274 – 589346
Mail: unypress.yogyakarta@gmail.com

Editor: Setyawan Pujiono
Disain sampul: Pudji Triwibowo
Tata Letak: Yudiati R

Perpustakaan Nasional: Katalog Dalam Terbitan

Lantip Diat Prasojo
SISTEM INFORMASI MANAJEMEN PENDIDIKAN/
Lantip Diat Prasojo
-Ed.1, Cet.1.- Yogyakarta: UNY Press 2013
viii + 119 hlm; 16 x 23 cm
ISBN: **ISBN: 978-602-7981-01-0**
1. sistem informasi manajemen pendidikan

1.Judul

Kata Pengantar

*Alhamdulillahi robbil 'aalamiin...*penulis panjatkan ke hadirat Allah SWT., atas segala nikmat yang diberikan sehingga buku ini dapat terselesaikan tanpa rintangan yang berarti.

Belakangan ini sudah banyak buku yang membahas tentang Sistem Informasi Manajemen Pendidikan (SIM Pendidikan), tetapi masih ditemukan kekurangan di berbagai sisi. Hampir seluruh buku yang beredar hanya membahas salah satu dari bagian SIM Pendidikan, sehingga untuk menguasainya, pembaca harus membeli banyak buku, belum lagi dihadapkan dengan masalah kualitas isi buku, tentunya hal ini merupakan investasi yang sangat mahal.

Buku ini hadir untuk memberi pemahaman dasar tentang Sistem Informasi Manajemen, sebelum lebih jauh mendalaminya. Pembahasan yang singkat, padat, jelas, dan disertai berbagai ilustrasi gambar, serat contoh-contoh aplikasi SIM Pendidikan.

Buku ini ditujukan untuk berbagai kalangan pembaca, mulai dari siswa SMA/SMK, Mahasiswa, Guru, Dosen, maupun kalangan umum yang tertarik dengan teknologi informasi. Buku ini dapat dijadikan referensi untuk mata kuliah Sistem Informasi Manajemen Pendidikan, Manajemen Informasi, Teknologi Informasi dan Komunikasi (TIK), dan berbagai mata kuliah sejenis lainnya.

Secara singkat, buku ini membahas berbagai hal yang terkait dengan SIM Pendidikan, seperti sistem, data, informasi, Sistem Manajemen Basis Data, Aplikasi SIM Pendidikan dalam dunia nyata, dan lain-lain.

Pada kesempatan ini penulis ingin menghaturkan rasa terima atas kontribusi berbagai pihak, diantaranya adalah:

- Jajaran Pimpinan fakultas Ilmu Pendidikan dan UNY.
- Prodi Manajemen Pendidikan Fakultas Ilmu Pendidikan UNY.
- Pihak Penerbit UNY Press.
- Orang tua dan Keluargaku tercinta

Anda dapat berinteraksi dengan memberikan pertanyaan atau saran mengenai materi buku demi perbaikan isi buku pada edisi berikutnya melalui alamat e-mail: lantip1975@gmail.com.

Yogyakarta, Januari
Penulis,
Lantip Diat Prasojo

Daftar Isi

Bab 1

Sistem Informasi Manajemen

1.1 Konsep Dasar

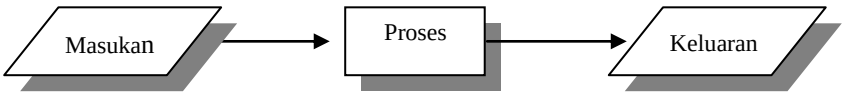
1.1.1 Pengertian Sistem

Makna kata "sistem" didefinisikan dengan berbagai pendekatan dan beragam istilah. Menurut Lucas (1992), Sistem adalah suatu pengorganisasian yang saling berinteraksi, saling tergantung dan terintegrasi dalam kesatuan variabel atau komponen. Jogiyanto (1999) mendefinisikan sistem ke dalam dua kelompok pendekatan, yaitu menekankan pada prosedur dan komponen atau elemennya.

Pendekatan sistem yang lebih menekankan pada prosedur mendefinisikan sistem sebagai suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkelompok dan bekerjasama untuk melakukan kegiatan pencapaian sasaran tertentu. Makna dari prosedur sendiri, yaitu urutan yang tepat dari tahapan-tahapan instruksi yang menerangkan apa (*what*) yang harus dikerjakan, siapa (*who*) yang mengerjakan, kapan (*when*) dikerjakan dan bagaimana (*how*) mengerjakannya. Pendekatan yang menekankan pada komponen mendefinisikan "sistem" sebagai kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu.

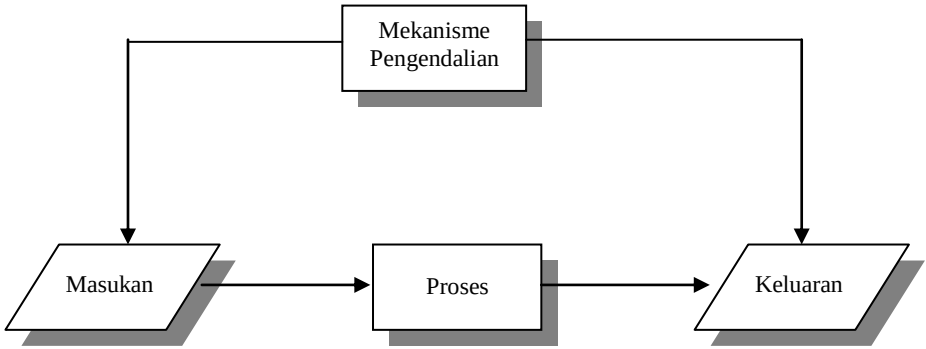
Beberapa penulis yang mendefinisikan "sistem" yang menekankan pada elemen atau komponennya adalah Barry E. Cushing (1974:12), Gordon B. Davis (1974:81).

McLeod, Jr., (1995: 13-14) menyatakan bahwa secara prinsip sistem dapat dikelompokkan menjadi dua, yaitu sistem terbuka dan sistem tertutup. Sistem terbuka adalah sistem yang dihubungkan dengan lingkungannya melalui arus sumber daya. Secara sederhana sistem terbuka dapat diilustrasikan seperti Gambar 1.1 berikut ini.



Gambar 1.1 Sistem Terbuka

Sistem tertutup adalah sistem yang tidak berinteraksi secara langsung dengan lingkungannya melalui arus sumber daya. Skema sistem tertutup dapat dilihat pada Gambar 1.2 di bawah ini.



Gambar 1.2 Sistem Tertutup

Berkaitan dengan Sistem Informasi Manajemen, di mana implementasinya memanfaatkan teknologi komputer, penulis mencoba membawa "sistem" yang dimaksud adalah sistem berbasis komputer. Dengan meminjam definisi dari *Webster's Dictionary* sebagaimana yang dikutip oleh Roger S. Pressman dalam bukunya "Rekayasa Perangkat Lunak", *Sistem Berbasis Komputer* didefinisikan sebagai serangkaian atau tatanan elemen-elemen yang diatur untuk mencapai tujuan yang ditentukan sebelumnya melalui pemrosesan informasi.

Tujuan yang dimaksud dimungkinkan untuk mendukung fungsi bisnis dari sistem itu sendiri. Selanjutnya, elemen-elemen sistem berbasis komputer yang digunakan untuk mencapai tujuan yang dimaksud terdiri atas:

1) Perangkat Lunak (*software*). Program (aplikasi) komputer, struktur data, dan dokumen yang berhubungan yang berfungsi untuk mempengaruhi metode logis, prosedur, dan kontrol yang dibutuhkan.
2) Perangkat Keras (*hardware*). Perangkat elektronik yang memberikan kemampuan penghitungan, dan perangkat elektromekanik.
3) Manusia (SDM). Pemakai dan operator perangkat keras dan lunak.
4) Sistem Basis Data (DBMS). Kumpulan data yang besar dan terorganisasi yang diakses melalui perangkat lunak.

5) Dokumentasi. Manual, formulir, dan informasi deskriptif lainnya yang menggambarkan penggunaan dan atau pengoperasian sistem.
6) Prosedur. Langkah-langkah yang menentukan penggunaan khusus dari masing-masing elemen sistem atau konteks prosedural dimana sistem berada.

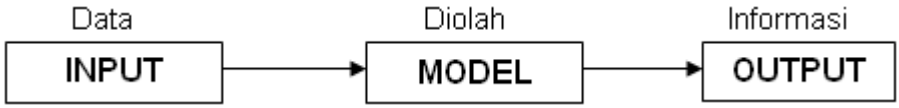
Elemen-elemen tersebut bergabung dengan cara tertentu untuk selanjutnya mentransformasikan informasi.

1.1.2 Pengertian Data dan Informasi

Data sering disebut sebagai bahan informasi. Data adalah fakta yang dikumpulkan dari pengukuran atau pengamatan (Tsichritis dan Lochovsky, 1970). Synanski dan Pulschen (1995) mendefinisikan data sebagai fakta mentah (dapat berupa angka, huruf, karakter khusus) yang menyampaikan sedikit arti. Agar data-data terkumpul menjadi berarti dan memberi manfaat, maka data-data tersebut harus diproses lebih lanjut.

Davis (1993: 28-29) mendefinisikan data sebagai bahan baku informasi, didefinisikan sebagai kelompok teratur simbol-simbol yang mewakili kuantitas, tindakan, benda dan sebagainya. Data terbentuk dari karakter yang dapat berupa alphabet, angka maupun simbul khusus seperti *, $, dan /. Data disusun untuk diolah dalam bentuk sruktur data, struktur file, basis data.

Menurut Jogiyanto (1999) data adalah sumber dari informasi. Data merupakan bentuk jamak dari bentuk tunggal datum atau data-item. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata. Kejadian-kejadian (*event*) adalah sesuatu yang terjadi pada saat yang tertentu. Kesatuan nyata (*fact* dan *entity*) adalah berupa suatu objek nyata seperti tempat, benda, dan orang yang betul-betul ada dan terjadi. Agar menjadi informasi yang berguna, data perlu diolah melalui sebuah siklus. Siklus ini disebut siklus pengolahan data (*data processing life cycle*).



Gambar 1.3 Siklus Pengolahan Data

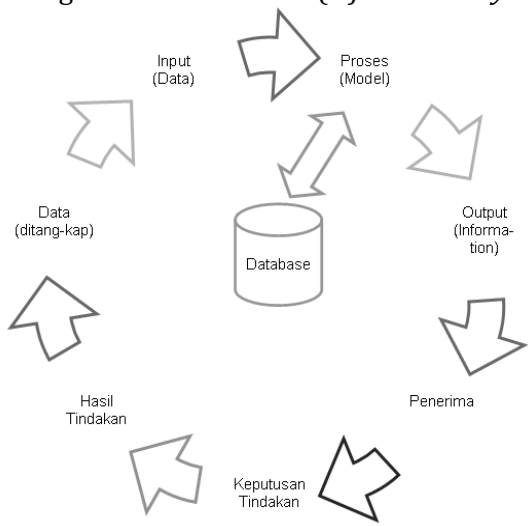
Informasi adalah arti dari hubungan dan penafsiran data yang mengizinkan seseorang untuk membuat keputusan (Tsichritis dan Lochovsky, 1970). Informasi dikatakan berharga jika informasi itu mempengaruhi proses pengambilan keputusan yang lebih baik.

Sasaran utama dari sistem informasi adalah menyediakan informasi yang akurat dan penting. Informasi juga dapat berarti beberapa kesatuan yang tak terukur yang dapat mengurangi ketidakpastian tentang suatu peristiwa atau langkah (Lucas, 1992).

Menurut Synanski dan Pulschen (1995), Informasi adalah pemrosesan data yang tampak dalam konteks untuk menyampaikan arti kepada orang lain. Lebih lanjut, Jogiyanto mendefinisikan informasi sebagai data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya.

1.1.2.1 Siklus Informasi

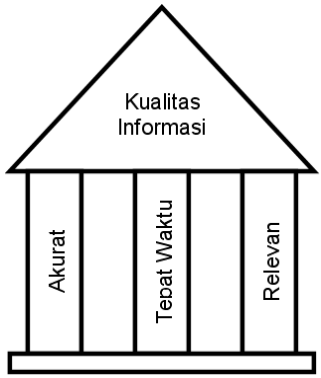
Sebagaimana telah disebutkan sebelumnya, data merupakan bentuk yang masih mentah yang belum dapat bercerita banyak, sehingga perlu diolah lebih lanjut. Data diolah melalui suatu model untuk menghasilkan informasi. Data diolah melalui model tertentu menjadi informasi yang dapat dimanfaatkan oleh penerima dalam membuat keputusan dan melakukan tindakan, yang berarti melakukan suatu tindakan lain yang akan membuat sejumlah data kembali. Data yang masih belum diolah akan disimpan dalam bentuk basis data. Data yang disimpan ini nantinya dapat diambil kembali untuk diolah kembali menjadi informasi. Data tersebut akan ditangkap sebagai input, diproses kembali lewat suatu model tertentu dan seterusnya membentuk suatu siklus. Siklus ini oleh John Burch disebut dengan siklus informasi (*information cycle*).



Gambar 1.4 Siklus Informasi

1.1.2.2 Kualitas Informasi

Kualitas dari suatu informasi tergantung dari 3 hal, yaitu: akurat (*accurate*), tepat pada waktunya (*timeliness*), dan relevan (*relevance*). John Burch dan Grudnitski menggambarkan kualitas informasi dengan bentuk bangunan yang ditunjang oleh tiga buah pilar.



Gambar 1.5 Pilar Kualitas Informasi

Keterangan :

- **Akurat**
  Informasi harus bebas dari kesalahan-kesalahan dan tidak bias atau menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya.
- **Tepat pada Waktunya**
  Informasi yang datang pada penerima tidak boleh terlambat, informasi yang sudah usang tidak akan mempunyai nilai lagi. Dewasa ini mahalnya nilai informasi karena cepatnya informasi yang mudah didapat, sehingga diperlukan teknologi-teknologi mutakhir untuk mendapatkannya, mengolah dan mengirimkannya.
- **Relevan**
  Informasi tersebut mempunyai manfaat untuk pemakainya. Hal ini disebabkan relevansi informasi untuk setiap orang satu dengan yang lainnya berbeda.

1.1.2.3 Nilai Informasi (*Cost-effectiveness*)

Nilai dari suatu informasi ditentukan dari dua hal, yaitu manfaat dan biaya mendapatkannya. Secara umum, suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya.

Menurut Synanski dan Pulschen (1995), selain **Accuracy, Relevance, Timeliness, Cost-effectiveness**, terdapat tiga atribut informasi lagi, yaitu:

- ***Completeness***
  Informasi menguraikan segala sesuatu yang harus diketahui untuk memahami situasi. Tujuannya adalah untuk mengumpulkan informasi selengkap mungkin.
- ***Auditability***
  Mengacu pada kemampuan untuk memeriksa kelengkapan dan keakuratan informasi. Tanpa kemampuan *audit* tidaklah mungkin untuk menentukan keakuratan, yang membawa ke dalam pertanyaan apakah kegunaan informasi.
- ***Reliability***
  Informasi tidaklah sempurna atau akurat 100%. Dengan reliabilitas maka dapat diambil rata-rata dari keenam atribut (*accuracy*, *relevance, timeliness, cost-effectiveness, auditability, reliability*) yang lain.

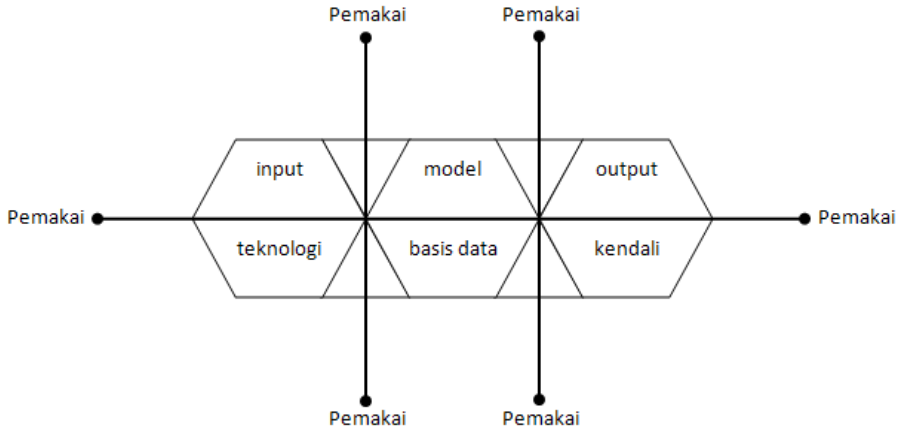
1.1.3 Sistem Informasi

Sebagaimana yang dikutip Jogiyanto dalam bukunya *Analisis dan Desain Sistem Informasi*, Robert A. Leitch dan K. Roscoe Davis mendefinisikan Sistem Informasi sebagai berikut.

> *Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.*

1.1.3.1 Komponen Sistem Informasi

John Burch dan Gary Grudnitski mengemukakan bahwa sistem informasi terdiri dari komponen-komponen yang disebutnya dengan istilah blok bangunan (*building block*), yaitu blok masukan (*input block*), blok model (*model block*), blok keluaran (*output block*), blok teknologi (*technology block*), blok basis data (*database block*), dan blok kendali (*controls block*). Sebagai suatu sistem keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasarannya.



Gambar 1.6 Blok Sistem informasi yang berinteraksi

Keterangan:

- **Blok Masukan**
  Input mewakili data yang masuk ke dalam sistem informasi. Input di sini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.
- **Blok Model**
  Blok ini terdiri atas kombinasi prosedur, logika, dan model matematik yang akan memanipulasi input dan data yang tersimpan di basis data dengan cara yang sudah ditentukan untuk menghasilkan keluaran yang diinginkan.
- **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.
- **Blok Teknologi**
  Teknologi merupakan *tool box* dalam sistem informasi. Teknologi digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri atas tiga bagian utama, yaitu teknisi (*humanware* dan *brainware*), perangkat lunak (*software*) dan perangkat keras (*hardware*). Teknisi dapat berupa orang-orang yang mengetahui teknologi dan membuatnya dapat beroperasi. Misalnya teknisi adalah operator komputer, programmer, operator pengolah kata,

spesialis telekomunikasi, analis sistem, penyimpan data dan lain sebagainya.

- **Blok Basis Data**
  Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan di dalam basis data untuk keperluan penyediaan informasi lebih lanjut. Data di dalam basis data perlu diorganisasikan sedemikian rupa, supaya informasi yang dihasilkan berkualitas.
  Organisasi basis data yang baik juga berguna untuk efisiensi kapasitas penyimpannya. Basis data diakses atau dimanipulasi dengan menggunakan perangkat lunak paket yang disebut dengan *DBMS* (*Database Management System*).
- **Blok Kendali**
  Banyak hal yang dapat merusak sistem informasi, seperti misalnya bencana alam, api, temperatur, air, debu, kecurangan-kecurangan, kegagalan-kegagalan sistem itu sendiri, kesalahan-kesalahan, ketidakefisienan, 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 langsung cepat diatasi.

Dalam perkembanganya, sebuah sistem informasi menjadi semakin kompleks dengan melibatkan banyak pemakai dan memerlukan sarana jaringan yang memungkinkan pemakai yang tersebar di pelbagai tempat yang berjauhan dapat berbagi informasi. Oleh karena itu, hal-hal yang bisa dikerjakan oleh sistem informasi terkait dengan kemampuan yang dimiliki dan dapat dilakukannya, sebagaimana menurut Turban, McLean, dan Wetherbe seperti berikut ini.

- Menyediakan komunikasi dalam organisasi atau antar organisasi yang murah, akurat dan cepat.
- Menyimpan informasi dalam jumlah yang sangat besar dalam ruang yang kecil tetapi mudah diakses.
- Memungkinkan pengaksesan informasi yang sangat banyak di seluruh dunia dengan cepat dan murah.
- Meningkatkan efektivitas dan efisiensi orang-orang yang bekerja dalam kelompok pada suatu tempat atau beberapa lokasi.

- Menyajikan informasi dengan jelas yang menggugah pikiran manusia
- Mengotomatiskan proses-proses bisnis yang semi-otomatis dan tugas-tugas yang dikerjakan secara manual.
- Mempercepat pengetikan dan penyuntingan.
- Pembiayaan yang lebih murah daripada mengerjakan secara manual.

1.1.4 Sistem Informasi Manajemen

Sistem Informasi Manejemen (selanjutnya disebut SIM) merupakan penerapatan sistem informasi di dalam organisasi untuk mendukung informasi-informasi yang dibutuhkan oleh semua tingkatan manajemen. SIM didefinisikan oleh George M. Scott sebagai berikut.

> *Suatu SIM adalah kumpulan dari interaksi-interaksi sistem-sistem informasi yang menyediakan informasi baik untuk kebutuhan manajerial maupun kebutuhan operasi*.

Menurut Barry E. Cushing SIM adalah sekumpulan dari manusia dan sumber-sumber daya modal di dalam suatu organisasi yang bertanggungjawab mengumpulkan dan mengolah data untuk menghasilkan informasi yang berguna untuk semua tingkatan manajemen di dalam kegiatan perencanaan dan pengendalian.

Menurut Gordon B. Davis SIM adalah sistem manusia atau mesin yang menyediakan informasi untuk mendukung operasi manajemen dan fungsi pengambilan keputusan dari suatu organisasi.

Lebih lanjut Gordon B. Davis juga menegaskan bahwa SIM selalu berhubungan dengan pengolahan informasi yang berbasis pada komputer (*computer-based information processing*). SIM merupakan suatu sistem yang melakukan fungsi-fungsi untuk menyediakan semua informasi yang mempengaruhi semua operasi organisasi.

1.2 Pengembangan Sistem

Pengembangan sistem dapat berarti menyusun suatu sistem yang baru untuk menggantikan sistem yang lama secara keseluruhan atau memperbaiki sistem yang telah ada. Sistem yang lama perlu diperbaiki atau diganti disebabkan karena beberapa hal, diantaranya adalah sebagai berikut.

1) Adanya permasalahan-permasalahan yang timbul di sistem yang lama, seperti: ketidakberesan dan pertumbuhan organisasi.
2) Untuk meraih berbagai kesempatan. Dengan keungguran sistem yang dimiliki, maka besar kemungkinan mempunyai kesempatan memenangkan persaingan usaha, dan lain-lain.
3) Adanya instruksi-instruksi, baik dari atas pimpinan atau dari luar organisasi, seperti peraturan pemerintah dan lain sebagainya.

Dengan telah dikembangkannya sistem yang baru, maka diharapkan akan terjadi peningkatan-peningkatan di sistem yang baru. Berikut adalah beberapa peningkatan yang dimaksud.

1) Kinerja.
   Merupakan peningkatan terhadap hasil kerja sistem yang baru sehingga menjadi lebih efektif.
2) Informasi.
   Merupakan peningkatan terhadap kualitas informasi yang disajikan oleh sistem.
3) Ekonomis.
   Merupakan peningkatan terhadap manfaat-manfaat atau leuntungan-keuntungan atau penurunan-penurunan biaya yang terjadi.
4) Pengendalian.
   Merupakan peningkatan terhadap pengendalian untuk mendeteksi dan memperbaiki kesalahan-kesalahan serta kecurangan-kecurangan yang dan akan terjadi.
5) Efisiensi.
   Merupakan peningkatan terhadap efisiensi operasi. Efisiensi yang dimaksud adalah yang berhubungan dengan bagaimana sumberdaya tersebut digunakan dengan pemborosan yang paling minimum.
6) Pelayanan.
   Merupakan peningkatan terhadap pelayanan yang diberikan oleh sistem.

1.2.1 Prinsip Pengembangan Sistem

Berikut ini adalah beberapa prinsip yang tidak boleh dilupakan dalam proses pengembangan sistem.

1) Sistem yang dikembangkan adalah untuk manajemen.
2) Sistem yang dikembangkan adalah investasi modal yang besar. Hal-hal yang perlu dipertimbangkan adalah sebagai berikut.
   - Semua alternatif yang ada harus diinvestigasi.
   - Investasi yang terbaik harus bernilai.
3) Sistem yang dikembangkan memerlukan orang yang terdidik.
4) Tahapan kerja dan tugas-tugas yang harus dilakukan dalam proses pengembangan sistem.
5) Proses pengembangan sistem tidak harus urut.
6) Jangan takut membatalkan proyek.
7) Dokumentasi harus ada untuk pedoman pengembangan sistem.

1.2.2 Siklus Hidup Pengembangan Sistem

Pengembangan sistem informasi yang berbasis komputer dapat merupakan tugas kompleks yang membutuhkan banyak sumber daya dan dapat memakan waktu berbulan-bulan bahkan bertahun-tahun untuk menyelesaikannya. Proses pengembangan sistem melewati beberapa tahapan mulai dari sistem itu direncanakan sampai sistem itu diterapkan, dioperasikan, dan dipelihara. Bila operasi sistem yang sudah dikembangkan masih timbul permasalahan-permasalahan yang kritis serta tidak dapat diatasi dalam tahap pemeliharaan sistem, maka perlu dikembangkan kembali suatu sistem untuk mengatasinya dan proses ini kembali ke tahap yang pertama, yaitu tahap perencanaan sistem. Siklus ini disebut dengan siklus hidup suatu sistem (*system life cycle*). Daur atau siklus hidup dari pengembangan sistem merupakan suatu bentuk yang digunakan untuk menggambarkan tahapan utama dan langkah-langkah di dalam tahapan tersebut dalam proses pengembangannya.

Ide dari *system life cycle* adalah sederhana dan masuk akal. Di *system life cycle*, tiap-tiap bagian dari pengembangan sistem dibagi menjadi beberapa tahapan kerja. tiap-tiap tahapan ini mempunyai karakteristik tersendiri. Tahapan utama siklus hidup pengembangan sistem dapat terdiri atas tahap perencanaan sistem (*systems planning*), analisis sistem (*systems analysis*), desain sistem (*systems design*), seleksi sistem (*systems selection*), implementasi sistem (*systems implementation*), dan perawatan sistem (*systems maintenance*). Berikut ini disajikan beberapa definisi tahapan-tahapan siklus hidup sistem dari berbagai ahli.

Donald H. Sander (1985), dalam bukunya "*Computer Today*" mendefinisikan tahapan siklus hidup sistem sebagai berikut.

1) Definisi masalah (*problem definition*).
2) Analisis sistem (*systems analysis*).
3) Desain sistem (*systems design*).
4) Implementasi sistem (*systems implementation*).

Robert H. Blismer (1985), dalam bukunya "*Computer Annual, An Introduction to Information Systems*" mendefinisikan tahapan siklus hidup sistem sebagai berikut.

1) Memahami sistem yang ada (*understanding the existing system*).
    a. Mengumpulkan informasi (*collecting information*).
    b. Menganalisis sistem yang ada (*analyzing the existing system*).
2) Mendefinisikan kebutuhan sistem yang baru (*defining new system requirement*).
    a. Pertimbangan-pertimbangan perencanaan (*planning considerations*).
    b. Kebutuhan keluaran, masukan, simpanan, dan pengolahan (*output*, *input*, *storage*, *and processing requirement*).
    c. Mengidentifikasi kriteria penilaian (*identify evaluation criteria*).
3) Proses desain sistem (*the system design process*).
    a. Desain keluaran (*output design*).
    b. Desain masukan (*input design*).
    c. Desain file (*file design*).
    d. Desain pengolahan sistem (*system processing design*).
    e. Pengendalian sistem (*system control*).
    f. Dokumentasi rinci (*detail documentation*).
4) Pengembangan dan implementasi sistem (*system development and implementation*).
    a. Menilai perangkat lunak paket (*evaluating packaged software*).
    b. Pengembangan perangkat lunak (*software development*).
    c. Dokumentasi sistem dan pelatihan (*system documentation and training*).
    d. Pengetesan sistem (*system testing*).
    e. Implementasi sistem (*system implementation*).

John Burch dan Gary Grunitski (1986), dalam bukunya "*Information Systems, Theory and Practice*" mendefinisikan siklus hidup sistem sebagai berikut.

1) Kebijakan dan perencanaan sistem (*system policy and planning*).
2) Pengembangan sistem (*system development*).
   a. Analisis sistem (*system analysis*).
   b. Desain sistem secara umum (*general system design*).
   c. Penilaian sistem (*system evaluation*).
   d. Desain sistem terinci (*detailed system design*).
   e. Implementasi sistem (*system implementation*).
3) Manajemen sistem dan operasi (*system management and operation*).

Tahapan-tahapan seperti ini sebenarnya merupakan tahapan di dalam pengembangan sistem teknik (*engineering systems*). Istilah *software engineering* merupakan proses pengembangan perangkat lunak yang merupakan subsistem dari pengembangan sistem informasi. Lebih lanjut tentang pemodelan proses pengembangan perangkat dibahas pada Bab 3.

1.2.3 Pendekatan Pengembangan Sistem

Terdapat beberapa pendekatan untuk mengembangkan sistem, yaitu sebagai berikut ini.

1) Pendekatan klasik lawan pendekatan terstruktur (dipandang dari segi metodologi yang digunakan).
2) Pendekatan sepotong lawan pendekatan sistem (dipandang dari sasaran yang akan dicapai).
3) Pendekatan bawah-naik lawan pendekatan atas-turun (dipandang dari cara menentukan kebutuhan dari sistem).
4) Pendekatan sistem menyeluruh lawan pendekatan moduler (dipandang dari cara mengembangkannya).
5) Pendekatan lompatan jauh lawan pendekatan berkembang (dipandang dari teknologi yang akan digunakan).

1.2.4 Metodologi Pengembangan Sistem

Metodologi adalah kesatuan metode-metode, prosedur-prosedur, konsep-konsep pekerjaan, aturan-aturan, dan postulat-postulat, yang digunakan oleh suatu ilmu pengetahuan, seni atau disiplin ilmu yang lainnya. Sedangkan, metode adalah suatu cara, teknik yang sistematis untuk mengerjakan sesuatu. Metodologi

pengembangan sistem berarti adalah metode-metode, prosedur-prosedur, konsep-konsep pekerjaan, aturan-aturan, dan postulat-postulat yang akan digunakan untuk mengembangkan suatu sistem informasi. Dalam pengembangan sistem informasi, perlu digunakan suatu metodologi yang dapat digunakan sebagai pedoman bagaimana dan apa yang harus dikerjakan selama pengembangan ini. Dengan mengikuti metode atau prosedur-prosedur yang diberikan oleh suatu metodologi, maka pengembangan sistem diharapkan akan dapat diselesaikan dengan berhasil. Urut-urutan prosedur untuk pemecahan masalah ini dikenal dengan istilah algoritma (*algorithm*).

Ide munculnya metodologi ini karena terjadinya kegagalan-kegagalan dalam pengembangan sistem. Kegagalan yang dimaksud adalah tidak efisien, kurang berhasil, dan kegagalan lainnya. Sebagian besar dari metodologi yang dibuat dimaksudkan hanya untuk tahap desain sistem saja. Akan tetapi, banyak juga yang dapat digunakan untuk tahap analisis sistem. Metodologi yang umum digunakan adalah metodologi analisis dan desain sistem terstruktur (*structured systems analysis and design*). Metodologi ini dapat digunakan pada tahap analisis dan desain.

1.3 Kebijakan dan Perencanaan Sistem

Sebelum suatu sistem informasi dikembangkan, umumnya terlebih dahulu dimulai dengan adanya suatu kebijakan dan perencanaan untuk mengembangkan sistem itu. Tanpa adanya perencanaan sistem yang baik, pengembangan sistem tidak akan dapat berjalan sesuai dengan yang diharapkan. Tanpa adanya kebijakan pengembangan sistem oleh manajemen puncak (*top management*), maka pengembangan sistem tidak akan mendapat dukungan dari manajemen puncak ini. Padahal dukungan dari manajemen puncak merupakan unsur yang sangat penting demi realisasinya kebijakan dan perencanaan sistem. Kebijakan sistem merupakan landasan dan dukungan dari manajemen puncak untuk membuat perencanaan sistem. Perencanaan sistem merupakan pedoman untuk melakukan pengembangan sistem.

1.3.1 Kebijakan Sistem

Kebijakan untuk mengembangkan sistem informasi dilakukan oleh manajemen puncak karena manajemen menginginkan untuk meraih kesempatan-kesempatan yang ada yang tidak dapat diraih oleh sistem yang lama atau sistem yang lama mempunyai banyak kelemahan-kelemahan yang perlu diperbaiki, misalnya untuk

meningkatkan efektivitas manajemen, meningkatkan produktivitas atau meningkatkan pelayanan yang lebih baik kepada langganan.

1.3.2 Perencanaan Sistem

Setelah manajemen puncak menetapkan kebijakan untuk mengembangkan sistem informasi, sebelum sistem ini sendiri dikembangkan, maka perlu direncanakan terlebih dahulu dengan cermat. Perencanaan sistem ini menyangkut estimasi dari kebutuhan-kebutuhan fisik, tenaga kerja, dan dana yang dibutuhkan untuk mendukung pengembangan sistem ini, serta untuk mendukung operasinya setelah diterapkan. Perencanaan sistem dapat terdiri atas perencanaan jangka pendek (*short-range*) dan perencanaan jangka panjang. Perencanaan jangka pendek meliputi periode 1 sampai 2 tahun. Perencanaan jangka panjang melingkupi periode sampai dengan 5 tahun. Karena perkembangan teknologi komputer yang sangat cepat, maka perencanaan pengembangan sistem informasi untuk periode lebih dari 5 tahun sudah tidak tepat lagi.

Dalam merencanakan sistem, terdapat beberapa bagian atau departemen dengan tugas dan fungsi masing-masing. Bagian-bagian yang dimaksud adalah sebagai berikut.

1) *Planning staff* mempunyai tugas untuk melakukan perencanaan sistem berdasarkan kebijakan sistem yang telah ditetapkan oleh manajemen puncak. Bila staf ini tidak ada, fungsinya dapat digantikan oleh departemen pengembangan sistem.
2) Departemen pengembangan sistem mempunyai tugas untuk mengembangkan sistem sesuai dengan rencana yang telah dibuat oleh *planning staff*. Bila departemen ini tidak ada, maka fungsinya dapat digantikan oleh konsultan pengembangan sistem di luar perusahaan.
3) Departemen pengolahan data mempunyai tugas untuk mengoperasikan sistem yang telah dikembangkan oleh departemen pengembangan sistem. Bila departemen ini tidak ada, maka harus dibentuk atau dapat digabung dengan departemen akuntansi bila ruang lingkupnya hanya berkisar pada pengolahan data akuntansi saja.

1.3.3 Proses Perencanaan Sistem

Proses dari perencanaan sistem dapat dikelompokkan dalam 3 proses utama, seperti berikut ini.

1) Merencanakan proyek-proyek sistem yang dilakukan oleh staf perencana sistem.

2) Menentukan proyek-proyek sistem yang akan dikembangkan dan dilakukan oleh komite pengarah.
3) Mendefinisikan proyek-proyek sistem yang akan dikembangkan dan dilakukan oleh analis sistem.

1.4 Analisis Sistem

Analisis sistem (*systems analysis*) dapat didefinisikan sebagai berikut.

> *Penguraian dari suatu sistem informasi yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengiden-tifikasi dan mengevaluasi permasalahan-permasalahan, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.*

Tahap analisis sistem dilakukan setelah tahap perencanaan sistem dan sebelum tahap desain sistem. Tahap analisis merupakan tahap yang kritis dan sangat penting, karena kesalahan di dalam tahap ini akan menyebabkan juga kesalahan di tahap selanjutnya.

1.4.1 Langkah-langkah Analisis Sistem

Langkah-langkah di dalam tahap analisis sistem hampir sama dengan langkah-langkah yang dilakukan dalam mendefinisikan proyek-proyek sistem yang akan dikembangkan di tahap perencanaan sistem. Perbedaannya terletak pada ruang lingkup tugasnya. Di analisis sistem, ruang lingkup tugasnya adalah lebih rinci. Di analisis sistem, penelitian yang dilakukan oleh analis sistem merupakan penelitian terinci, sedangkan perencanaan sistem sifatnya hanya penelitian pendahuluan.

Di dalam tahap analisis sistem terdapat langkah-langkah dasar yang harus dilakukan oleh analis sistem sebagai berikut ini.

1) *Identify*, yaitu mengidentifikasi masalah.
2) *Undestand*, memahami kerja dari sistem yang ada.
3) *Analyze*, menganalisis sistem.
4) *Report*, yaitu membuat laporan hasil analisis.

Untuk masing-masing langkah-langkah ini, beberapa tugas perlu dilakukan oleh analis sistem. Supaya memudahkan untuk melakukan koordinasi dan pengawasan, koordinator tim analis dapat membuat suatu kertas kerja yang memuat tugas-tugas yang harus dikerjakan untuk masing-masing langkah analisis sistem ini.

1.4.2 Identifikasi Masalah

Mengidentifikasi masalah merupakan langkah pertama yang dilakukan dalam tahap analisis sistem. Masalah dapat didefinisikan sebagai suatu pertanyaan yang diinginkan untuk dipecahkan. Masalah inilah yang menyebabkan sasaran dari sistem tidak dapat dicapai. Oleh karena itu, pada tahap analisis sistem, langkah pertama yang harus dilakukan oleh analis sistem adalah mengidentifikasi terlebih dahulu masalah-masalah yang terjadi. Tugas yang harus dilakukannya adalah sebagai berikut.

1) Mengidentifikasi penyebab masalah.
   Analis sistem harus mempunyai pengetahuan yang cukup tentang aplikasi yang sedang dianalisisnya. Untuk aplikasi bisnis, analis sistem perlu mempunyai pengetahuan tentang sistem bisnis yang diterapkan di organisasi, sehingga dapat mengidentifikasi penyebab-penyebab terjadinya masalah ini. Tahap mengidentifikasi penyebab masalah dapat dimulai dengan mengkaji ulang terlebih dahulu subjek-subjek permasalahan yang telah diutarakan oleh manajemen atau yang telah ditemukan analis sistem di tahap perencanaan sistem.
2) Mengidentifikasi titik keputusan.
   Setelah penyebab terjadinya masalah dapat diidentifikasi, selanjutnya harus diidentifikasi titik keputusan penyebab masalah tersebut. Sebagai dasar identifikasi titik-titik keputusan ini, dapat digunakan dokumen sistem bagan alir formulir (*paperwork flowchart* atau *form flowchart*) bila dokumentasi ini dimiliki oleh perusahaan.
3) Mengidentifikasi personil-personil kunci.
   Setelah titik-titik keputusan penyebab masalah dapat diidentifikasi beserta lokasi terjadinya, maka selanjutnya yang perlu diidentifikasi adalah personil-personil kunci, baik yang langsung maupun yang tidak langsung dapat menyebabkan terjadinya masalah tersebut. Identifikasi personil-personil kunci ini dapat dilakukan dengan mengacu pada bagan alir dokumen yang ada di perusahaan serta dokumen deskripsi jabatan (*job description*).

1.4.3 Memahami Cara Kerja dari Sistem yang Ada

Langkah kedua dari tahap analisis sistem adalah memahami kerja dari sistem yang ada. Langkah ini dapat dilakukan dengan mempelajari secara terinci bagaimana sistem yang ada beroperasi. Untuk mempelajari operasi dari sistem ini diperlukan data yang

dapat diperoleh dengan cara melakukan penelitian. Bila ditahap perencanaan sistem juga pernah dilakukan penelitian untuk memperoleh data, penelitian ini sifatnya adalah penelitian pendahuluan. Selanjutnya, pada tahap analisis sistem, penelitian yang dilakukan adalah penelitian terinci.

Beberapa tugas yang dilakukan dalam memahami kerja sistem yang ada adalah sebagai berikut.

1) Menentukan jenis penelitian.
   Sebelum penelitian dilakukan, sebaiknya ditentukan terlebih dahulu jenis dari penelitian untuk masing-masing titik keputusan yang akan diteliti.
2) Menentukan jadwal penelitian.
   Penelitian akan dilakukan di tiap-tiap lokasi titik keputusan yang akan diteliti. Penelitian juga biasanya akan dilakukan oleh beberapa peneliti dan memakan waktu yang cukup lama. Supaya penelitian dapat dilakukan secara efisien dan efektif, maka jadwal dari penelitian harus direncanakan terlebih dahulu yang meliputi:
   - Mengatur jadwal wawancara.
   - Mengatur jadwal observasi.
   - Mengatur jadwal pengambilan sampel.
3) Membuat penugasan penelitian.
   Setelah rencana jadwal penelitian selesai dibuat, maka tugas dari tiap-tiap anggota tim analis sistem untuk melakukan penelitian telah dapat ditentukan. Koordinator tim analis sistem dapat membuat surat penugasan kepada masing-masing anggota tim analis sistem ini dengan menyertakan lampiran kegiatan penelitian yang harus dilakukan.
4) Membuat agenda wawancara.
   Sebelum suatu wawancara dilaksanakan, akan lebih bijaksana bila waktu dan materi wawancara ini direncanakan terlebih dahulu. Rencana ini dapat ditulis di agenda wawancara dan dibawa selama wawancara berlangsung. Pewawancara dapat melakukan wawancara dengan dasar agenda wawancara ini. Tujuan utama pembuatan agenda wawancara supaya wawancara dapat diselesaikan tepat waktu dan tidak ada materi yang terlewatkan.
5) Mengumpulkan hasil penelitian.
   Fakta atau data yang diperoleh dari hasil penelitian harus dikumpulkan sebagai suatu dokumentasi sistem lama.

Dokumentasi dari hasil penelitian ini diperlukan untuk beberapa hal, yaitu sebagai berikut.

- Membantu kelengkapan (*aid to completeness*). Dengan digunakan formulir-formulir standar untuk mencatat fakta, maka data yang belum terkumpul akan terlihat.
- Membantu analisis (*aid to analysis*). Data yang dicatat dalam bentuk tabel atau bagan memungkinkan sistem akan lebih mudah dipahami dan dianalisis.
- Membantu komunikasi (*aid to communication*). Formulir-formulir yang standar akan membantu anggota-anggota tim analis untuk berkomunikasi dengan efektif satu dengan yang lainnya. Selain itu, dapat membantu komunikasi antara analis, pemrogram komputer, operator, dan pemakai sistem.
- Membantu pelatihan (*aid to training*). Pelatihan akan lebih efektif bila dilampiri dengan bahan-bahan yang diperlukan secara tertulis.
- Membantu keamanan (*aid to security*). Dokumentasi yang berisi dengan fakta terkumpul dapat diibaratkan sebagai *bestek* rancangan gedung yang telah digambar oleh arsitek dan telah dihitung oleh insinyur teknik sipil. Bila gedung yang dibangun tidak sesuai dengan keinginan pemakai, atau ada perubahan-perubahan yang perlu dilakukan, atau misalnya gedung yang sudah dibuat meng-alami kerusakan-kerusakan, maka dengan adanya dokumentasi, perbaikan-perbaikan atau modifikasi-modifikasi akan lebih mudah dilakukan.

1.4.4 Menganalisis Hasil Penelitian

Langkah analisis dilakukan berdasarkan data yang telah diperoleh dari hasil penelitian yang telah dilakukan. Menganalisis hasil penelitian sering sulit dilakukan oleh analis sistem yang masih baru. Pengalaman menunjukkan bahwa banyak analis sistem yang masih baru mencoba untuk memecahkan masalah tanpa menganalisisnya. Beberapa hasil penelitian yang perlu dianalisis adalah sebagai berikut.

1) Menganalisis kelemahan sistem. Tujuannya untuk menemukan kelemahan-kelemahan dan permasalahan-permasalahan dari sistem yang ada, diantaranya:

- Menganalisis distribusi pekerjaan. Distribusi pekerjaan menunjukkan beban dari masing-masing personal atau unit organisasi dalam menangani kegiatan yang sama.
- Menganalisis pengukuran pekerjaan. Pengukuran pekerjaan menunjukkan pemahaman terdahap kebijaksanaan dan prosedur, kepuasan produktivitas karyawan, ketercapaian sasaran, dan ukuran pekerjaan lainnya.
- Menganalisis keandalan. Keandalan menunjukkan banyaknya kesalahan-kesalahan yang dilakukan dalam suatu kegiatan. Semakin handal berarti semakin sedikit kesalahan yang dilakukan.
- Menganalisis dokumen. Analisis dokumen menunjukkan seberapa perlu dokumen-dokumen yang ada, efektivitas rancangan dokumen, mengetahui tembusan-tembusan dokumen, dan lain sebagainya.
- Menganalisis laporan. Analisis laporan menunjukkan tingkat kemudahan mempersiapkan file atau dokumen yang ada, mengetahui ada atau tidaknya duplikasi dokumen, dan lainnya.
- Menganalisis teknologi. Analisis teknologi menunjukkan teknologi yang digunakan serta tingkat efektivitas dan efisiensinya dalam penelitian tersebut.

2) Menganalisis kebutuhan informasi pemakai/manajemen.
   Walaupun menganalisis kelemahan-kelemahan dan permasalahan-permasalahan yang terjadi merupakan tugas yang perlu, tetapi tugas ini saja belumlah cukup. Tugas lain dari analis sistem yang masih diperlukan sehubungan dengan sasaran utama sistem informasi, yaitu menyediakan informasi yang dibutuhkan bagi para pemakainya.

1.4.5 Membuat Laporan Hasil Analisis

Setelah proses analsis sistem ini selesai dilakukan, tugas berikutnya dari analis sistem dan timnya adalah membuat laporan hasil analisis. Laporan ini diserahkan kepada *steering committe* yang nantinya akan diteruskan ke manajemen. Pihak manajemen bersama-sama dengan panitia pengarah dan pemakai sistem akan mempelajari temuan-temuan dan analis yang telah dilakukan oleh analis sistem yang disajikan dalam laporan ini. Tujuan utama dari penyerahan laporan ini kepada manajemen adalah sebagai berikut.

- Pelaporan bahwa analisis telah selesai dilakukan.
- Meluruskan kesalahan-kesalahan mengenai apa yang telah ditemukan dan dianalisis oleh analis sistem tetapi tidak sesuai menurut manajemen.
- Meminta pendapat-pendapat dan saran-saran dari pihak manajemen.
- Meminta persetujuan kepada pihak manajemen untuk melakukan tindakan selanjutnya, dapat berupa meneruskan ke tahap desain sistem atau menghentikan proyek bila dipandang tidak layak lagi.

1.5 Desain Sistem

Desain sistem mengidentifikasi komponen-komponen sistem informasi yang akan didesain secara rinci. Tahap desain ini dilakukan setelah tahap analisis sistem selesai dilakukan dan hasil analisis disetujui oleh manajemen. Tujuan dari desain adalah untuk memberikan gambaran secara umum dan terinci kepada pengguna tentang sistem yang baru. Untuk mencapai tujuan ini, analis sistem harus dapat mencapai sasaran-sasaran sebagai berikut.

1) Sistem desain harus berguna, mudah dipahami dan nantinya mudah digunakan. Ini berarti bahwa data harus mudah ditangkap, metode-metode harus mudah diterapkan, dan informasi harus mudah dihasilkan, serta mudah dipahami dan digunakan.
2) Desain sistem harus dapat mendukung tujuan utama perusahaan sesuai dengan yang telah didefinisikan pada tahap perencanaan sistem yang dilanjutkan pada tahap analisis sistem.
3) Desain sistem harus efisien dan efektif untuk dapat mendukung pengolahan transaksi, pelaporan manajemen dan mendukung keputusan yang akan dilakukan oleh manajemen, termasuk tugas-tugas lainnya yang tidak dilakukan oleh komputer.
4) Desain sistem harus dapat mempersiapkan rancang bangun yang terinci untuk masing-masing komponen dari sistem informasi yang meliputi data dan informasi, simpanan data, metode-metode, prosedur-prosedur, orang-orang, perangkat keras, perangkat lunak, dan pengendalian intern.

1.5.1 Teknik Desain Sistem

Pada desain sistem informasi, semua teknik-teknik yang digunakan di tahap analisis sistem dapat juga digunakan pada tahap ini, seperti *flowchart* dan formulir-formulir. Selain itu, terdapat juga teknik-teknik lain yang dapat diterapkan pada tahap desain sistem ini, yaitu teknik sketsa di kertas kosong dan pembuatan prototipe (*prototyping*).

Teknik sketsa kertas kosong (*blank-paper sketching*) dilakukan dengan menggunakan lembar kertas kosong untuk sketsa desain. *Prototyping* merupakan pendekatan yang diilhami dari desain teknik, seperti pembuatan prototipe bangunan oleh arsitek. Penekanan dari teknik ini adalah pada pembuatan suatu model kerja dari sistem final secara cepat. Model ini yang disebut dengan prototipe (*prototype*). Sistem prototipe ini kemudian dapat diperiksa oleh pengguna untuk menentukan apakah sudah sesuai dengan yang diinginkan atau belum. Jika belum sesuai dengan yang diinginkan, maka prototipe dapat direvisi sampai sesuai dengan yang diinginkan. Pekerjaan ini dapat dilakukan dengan mudah dan cepat, karena pembuatan prototipe baru atau merevisinya tidak dibutuhkan waktu yang lama. Secara bertahap prototipe ini akan dikembangkan menjadi sistem yang final. Pendekatan prototipe pada tahap desain sistem ini disebut dengan *design by prototyping*. *Prototyping* ini menggantikan cara desain tradisional yang menggunakan kertas. Prototipe di-*review* oleh pengguna, perubahan-perubahan dicatat dan prototipe baru kemudian dikembangkan. Proses ini disebut dengan *prototype loop*.

Prototyping belum banyak digunakan pada masa lalu, karena banyak pemrograman komputer yang dilakukan dengan menggunakan bahasa-bahasa yang tidak mendukung pembuatan prototipe, kalaupun bisa akan sulit dan memakan waktu yang lama, apalagi bila prototipe selalu diubah-ubah dan dikembangkan terus sampai dapat diterima dan disetujui oleh pengguna. Bahasa-bahasa yang dimaksud adalah COBOL, FORTRAN, PL/1, dan BASIC. Sejak muncul bahasa-bahasa komputer yang lebih mudah digunakan, yaitu bahasa generasi keempat (*fourth generation language*) atau yang lebih baru, seperti dBASE IV, Borland Delphi, Microsoft Visual Basic, dan lain sebagainya. Pemakaian bahasa-bahasa ini akan mudah untuk membuat program penghasil laporan-laporan dan form masukan. Setelah prototipe selesai dibuat dan disetujui, analis sistem dapat menentukan keputusannya, yaitu prototipe diteruskan lagi dengan

bahasa pemrograman yang telah digunakan atau prototipe diubah dengan bahasa pemrograman yang lain.

1.5.2 Desain Komponen Sistem

Pada tahap desain, komponen-komponen sistem informasi dirancang dengan tujuan untuk dikomunikasikan kepada pengguna bukan untuk pemrogram. Komponen sistem informasi yang didesain adalah model, masukan, keluaran, basis data, teknologi, dan kontrol.

1.5.2.1 Desain Teknologi

Teknologi digunakan untuk menerima masukan, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran, dan membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri dari tiga bagian utama, yaitu perangkat keras (*hardware*), perangkat lunak (*software*), dan teknisi (*humanware* atau *brainware*). Teknisi dapat berupa orang-orang yang mengetahui teknologi dan membuatnya dapat beroperasi. Teknisi misalnya adalah operator komputer, pemrogram, spesialis telekomunikasi, penyimpan data, sistem analis, dan lain sebagainya.

1.5.2.2 Desain Model

Analis sistem dapat mendesain model dari sistem informasi yang diusulkan dalam bentuk *physical system* dan *logical model*. Bagan alir sistem (*flowchart*) merupakan alat yang tepat digunakan untuk menggambarkan *physical system*. Simbol-simbol bagan alir sistem ini menunjukkan secara tepat arti fisiknya, seperti simbol terminal, *hard disk*, laporan-laporan.

Logical model dari sistem informasi lebih menjelaskan kepada pengguna bagaimana nantinya fungsi-fungsi di sistem informasi secara logika akan bekerja. *Logical model* dapat digambar dengan menggunakan diagram arus data (*data flow diagram*). Arus data di DFD dapat dijelaskan dengan menggunakan kamus data (*data dictionary*). Teori teknis dan cara penggunaan *flowchart* dan DFD akan dibahas pada bagian lampiran.

1.5.2.3 Desain Basis Data

Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di simpanan luar komputer dan digunakan perangkat lunak tertentu untuk memanipulasinya. Basis data merupakan salah satu komponen yang penting di sistem informasi, karena berfungsi sebagai basis penyedia

informasi bagi para pemakainya. Penerapan basis data dalam sistem informasi disebut dengan sistem basis data (*database system*). Sistem basis data ini adalah suatu sistem informasi yang mengintegrasikan kumpulan dari data yang saling berhubungan satu dengan lainnya dan membuatnya tersedia untuk beberapa aplikasi yang bermacam-macam di dalam suatu organisasi.

Untuk tahap desain basis data, yang perlu dilakukan analisis oleh analis adalah mengidentifikasi terlebih dahulu file-file yang diperlukan oleh sistem informasi. File-file basis data yang dibutuhkan oleh sistem dapat dilihat pada desain model yang digambarkan dalam bentuk diagram arus data. Langkah-langkah desain basis data secara umum adalah sebagai berikut.

1) Menentukan kebutuhan file basis data untuk sistem baru.
   File yang dibutuhkan dapat ditentukan dari DAD sistem baru yang telah dibuat.
2) Menentukan parameter dari file basis data.
   Setelah file-file yang dibutuhkan telah dapat ditentukan, maka parameter dari file selanjutnya juga dapat ditentukan. Parameter ini meliputi:
   - Tipe dari file: file induk, file transaksi, file sementara, dan file lain yang dianggap perlu.
   - Media file: *hard disk*, disket, *flash disk*, atau pita magnetik.
   - Organisasi dari file: apakah file tradisional (file urut, ISAM, atau file akses langsung) atau organisasi basis data (struktur berjenjang, jaringan, atau relasional).
   - *Filed* kunci dari file.

1.5.2.4 Desain Masukan

Merupakan desain masukan untuk sistem yang akan dibangun. Bila sistem yang dibangun berupa perangkat lunak sistem informasi, desain masukannya berupa form-form aplikasi. Selanjutnya, menentukan alat masukan apa yang diperlukan oleh sistem yang akan dibangun. Alat masukan yang dimaksud masih dibedakan lagi menjadi dua golongan, yaitu alat masukan langsung (*online input device*) dan alat masukan tidak langsung (*offline input device*). Alat masukan langsung merupakan alat yang langsung dihubungkan dengan CPU, misalnya adalah *keyboard*, *mouse*, *touch screen*, dan lain sebagainya. Sedangkan, alat masukan tidak langsung adalah alat masukan yang tidak langsung dihubungkan dengan CPU, misalnya KTC (*key to card*), KTT (*key to tape*), dan KTD (*key to disk*).

Hal-hal yang perlu didesain secara rinci untuk masukan adalah bentuk dari dokumen dasar yang digunakan untuk menangkap data, kode-kode input yang digunakan, dan bentuk dari tampilan masukan di alat masukan. Langkah-langkah desain masukan adalah sebagai berikut.

1) Menentukan kebutuhan masukan dari sistem baru.
   Masukan yang akan didesain dapat ditentukan dari DFD sistem yang baru yang telah dibuat. Masukan di DFD ditunjukkan oleh arus data dari kesatuan luar ke suatu proses dan bentuk tampilan masukan di alat masukan ditunjukkan oleh suatu proses memasukkan data.
2) Menentukan parameter dari masukan.
   Setelah masukan-masukan yang akan didesain telah dapat ditentukan, maka parameter dari masukan selanjutnya juga dapat ditentukan. Parameter ini meliputi.
   - Bentuk dari masukan, dokumen dasar atau bentuk isian di alat masukan (dialog layar terminal).
   - Sumber masukan.
   - Jumlah tembusan untuk input berupa dokumen dasar dan distribusinya.
   - Alat masukan yang digunakan.
   - Valome masukan.
   - Periode masukan.

1.5.2.5 Desain Keluaran

Keluaran adalah produk dari sistem informasi yang dapat dilihat. Istilah keluaran ini kadang-kadang membingungkan, karena keluaran dapat terdiri atas macam-macam jenis. Keluaran dapat berupa hasil di media keras (seperti: kertas, *microfilm*) atau hasil di media lunak (berupa tampilan layar video, *software*). Di samping itu, keluaran dapat juga berupa hasil dari suatu proses yang akan digunakan oleh proses lain dan tersimpan di suatu media, seperti *tape*, *disk*, atau kartu.

Desain keluaran secara umum dapat dilakukan dengan langkah-langkah sebagai berikut.

1) Menentukan kebutuhan keluaran dari sistem baru.
   Keluaran yang akan didesain dapat ditentukan dari DFD sistem baru yang telah dibuat. Keluaran di DFD ditunjukkan oleh arus data dari suatu proses ke kesatuan luar atau dari proses ke proses yang lainnya.
2) Menentukan parameter dari keluaran.

Setelah keluaran-keluaran yang akan didesain telah dapat ditentukan, maka parameter dari keluaran selanjutnya juga dapat ditentukan. Parameter ini meliputi tipe dari keluaran, formatnya, media yang digunakan, alat keluaran yang digunakan, jumlah tembusannya, distribusinya, dan periode keluaran.

1.5.2.6 Desain Kontrol

Suatu sistem merupakan subjek dari mismanajemen, kesalahan-kesalahan, kecurangan-kecurangan, dan penyelewengan-penyelewengan umum lainnya. Pengendalian yang diterapkan pada sistem informasi sangat berguna untuk tujuan mencegah atau menjaga terjadinya hal-hal yang tidak diinginkan. Pengendalian intern juga dapat digunakan untuk melacak kesalahan-kesalahan yang sudah terjadi sehingga dapat dikoreksi. Dalam pengembangan suatu sistem informasi, analisis dan perancang sistem harus memikirkan pengendalian yang ada atau yang akan diterapkan. Sistem informasi sebagai sistem yang terbuka tidak bisa dijamin sebagai suatu sistem yang bebas dari kesalahan-kesalahan atau kecurangan-kecurangan. Apabila sistem tersebut dilengkapi dengan suatu pengendalian yang berguna untuk mencegah atau menjaga hal-hal yang negatif tersebut, maka sistem akan dapat terus melangsungkan hidupnya. Suatu sistem harus dapat melindungi dirinya sendiri. Pengendalian yang baik merupakan cara bagi suatu sistem informasi untuk melindungi dirinya sendiri dari hal-hal yang merugikan.

Pengendalian dalam sistem informasi dapat dikategorikan lebih lanjut ke dalam pengendalian secara umum (*general control*) dan pengendalian aplikasi (*application control*). Pengendalian secara umum merupakan pengendalian di luar aplikasi pengolahan data, seperti: pengendalian organisasi, pengendalian dokumentasi, pengendalian perangkat keras, pengendalian keamanan fisik, pengendalian keamanan data, dan pengendalian komunikasi. Pengendalian aplikasi merupakan pengendalian yang diterapkan selama proses pengolahan data berlangsung. Pengendalian aplikasi dapat dikategorikan ke dalam pengendalian masukan (*input control*), pengendalian pengolahan (*procrssing control*), dan pengendalian keluaran (*output control*).

1.5.3 Laporan Desain

Setelah komponen-komponen sistem informasi didesain, maka laporan mengenai ini perlu dibuat dan diberikan kepada pemakai sistem dan manajemen. Pemakai sistem dan manajemen dapat memberikan pendapat-pendapat dan usulan-usulan perbaikan dari desain ini. Melalui laporan desain ini, analis sistem mengonfirmasikan kepada pemakai sistem dan manajemen apakah benar sistem informasi seperti ini yang mereka butuhkan. Jika pemakai sistem dan manajemen ternyata tidak menghendaki sistem seperti itu, maka waktu analis sistem akan terbuang dengan percuma untuk melakukan desain lagi.

Bab 2

Sistem Manajemen Basis Data

2.1 Konsep Dasar

Sistem basis data didefinisikan sebagai suatu sistem yang terdiri atas kumpulan *file*/tabel yang saling berhubungan (dalam sebuah basis data pada sebuah sistem komputer) dan kumpulan program (sistem manajemen basis data) yang memungkinkan beberapa pemakai dan atau program lain untuk mengakses dan memanipulasi *file*/tabel tersebut (Fathansyah, 2002). Basis data harus mempunyai tiga fitur penting (Martin, 1980), yaitu:

- ***Accessibility***
  Mengacu pada kemampuan akses untuk menyimpan atau memperoleh kembali data dengan identitas tertentu.
- ***Generality***
  Mengacu pada kemampuan dalam mengakses semua informasi untuk memperoleh kembali atau memodifikasi data.
- ***Flexibility***
  Mengacu pada kemampuan dalam kemudahan penggunaan dan pengembangan basis data.

Pada sebuah sistem basis data yang lengkap akan terdapat komponen-komponen utama, yaitu perangkat keras (*hardware*), sistem operasi (*operating system*), basis data, sistem pengolahan basis data (*database management system* atau dbms), dan pemakai.

Tujuan awal dan utama dalam pengelolaan data dalam basis data adalah agar dapat memperoleh/menemukan kembali data yang dicari dengan mudah dan cepat. Hal yang sangat ditonjolkan dalam basis data adalah pengorganisasian data yang akan disimpan sesuai fungsi atau jenisnya. Pengaturan ini dapat berbentuk sejumlah tabel terpisah atau dalam bentuk pendefinisian *field-field* data setiap tabel.

Operasi-operasi dasar yang dapat dilakukan berkenaan dengan basis data, yang biasanya dapat ditangani oleh *DBMS* meliputi: pembuatan (*create*) basis data, penghapusan (*drop*) basis data, pembuatan tabel baru (*create table*) ke dalam basis data, penghapusan tabel (*drop table*) dari suatu basis data, pengubahan

struktur tabel (*alter table*). Operasi rutinitas yang mewakili aktivitas dalam basis data, yaitu pengelolaan dan pengolahan data yang meliputi penambahan data baru (*insert*), pengambilan data (*select*), pengubahan data (*update*), dan penghapusan data (*delete*) dari sebuah tabel.

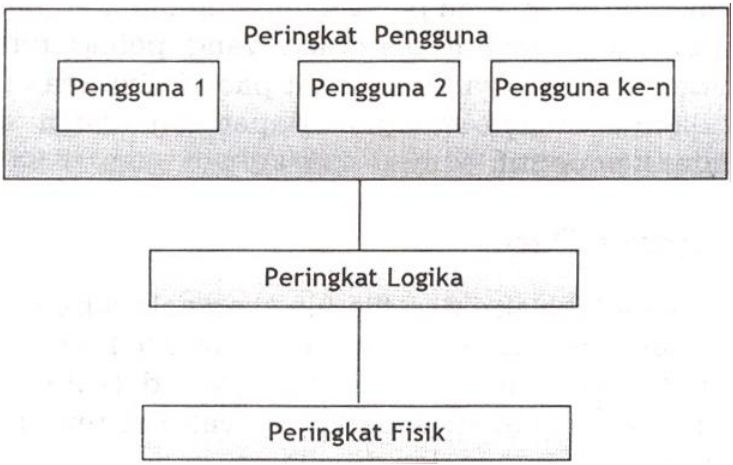
2.2 Abstraksi Data

Sistem basis data adalah koleksi dari file-file yang saling berhubungan di mana program-program yang dibuat pengguna dapat mengakses dan memodifikasi file-file tersebut. Salah satu tujuan dari sistem basis data adalah menyediakan pengguna suatu pandangan abstrak dari data, yaitu sistem menyembunyikan rincian bagaimana data disimpan dan dipelihara.

Agar sistem berguna, pengguna seharusnya dapat memanggil data secara efisien. Karena kebanyakan pengguna sistem basis data adalah orang-orang yang tidak/kurang terlatih di bidang teknologi komputer, pengembang seharusnya menyembunyikan kompleksitas sistem dari pengguna. Ini dapat dilakukan dengan menyediakan beberapa peringkat abstraksi yang bertujuan untuk menyederhanakan interaksi pengguna dengan sistem. Beberapa peringkat abstraksi itu adalah sebagai berikut.

- **Peringkat Fisik**. Ini adalah peringkat terendah dari abstraksi yang mendeskripsikan bagaimana data sesungguhnya disimpan dalam media penyimpanan fisik seperti *harddisk*, pita magnetik, dan sebagainya. Peringkat fisik mendeskripsikan struktur data peringkat rendah yang kompleks secara rinci, misalnya struktur data yang digunakan, pola representasi bit yang digunakan, dan sebagainya.
- **Peringkat Logika**. Peringkat yang lebih tinggi dari peringkat fisik adalah abstraksi yang mendeskripsikan data apa yang disimpan di basis data, dan hubungan apa yang ada antara data-data tersebut. Peringkat logika mendeskripsikan basis data dengan struktur yang relatif sederhana. Meskipun implementasi dari struktur yang sederhana itu mungkin mengandung struktur fisik yang kompleks, pengguna pada peringkat logika tidak perlu tahu kerumitan tersebut. Administrator basis data, seseorang yang memutuskan informasi apa yang harus disimpan di basis data, menggunakan abstraksi peringkat logika.
- **Peringkat Pengguna**. Ini adalah peringkat tertinggi dari abstraksi. Meskipun peringkat logika sudah cukup sederhana,

namun pada basis data yang berukuran sangat besar, kompleksitas masih dijumpai karena banyaknya jenis data dan informasi yang tersimpan pada basis data. Kebanyakan pengguna tidak membutuhkan semua informasi itu, mereka kebanyakan hanya perlu untuk mengakses bagian tertentu dari basis data. Peringkat pengguna menyederhanakan interaksi pengguna dengan sistem. Peringkat pengguna yang sering dijumpai adalah antarmuka pengguna grafis (GUI - *Graphical User Interface*) pada perangkat lunak aplikasi basis data yang menyederhanakan interaksi pengguna dengan sistem basis data.



Gambar 2.1 Tiga peringkat abstraksi data

Kita sekarang akan membahas analogi abstraksi data di atas dengan konsep tipe data yang dapat dijumpai pada bahasa pemrograman tingkat tinggi, katakanlah Pascal. Misalnya kita memiliki struktur *record* seperti di bawah ini:

```
type Mahasiswa = record
        NIM: string[8];
    Nama: string;
    Alamat: string;
end;
```

Kode di atas mendefinisikan suatu *record* dengan nama **Mahasiswa** yang mengandung 3 buah *field*. Pada peringkat fisik, *record* mungkin dapat dideskripsikan sebagai suatu blok tertentu yang berukuran

beberapa *byte* dalam tempat penyimpanan. Pascal menyembunyikan rincian fisik ini dari pemrogram. Dengan cara yang sama, sistem basis data menyembunyikan berbagai cara penyimpanan *record* secara fisik dari pemrogram basis data. Hanya administrator basis data yang kadang-kadang peduli dengan rincian organisasi fisik data serta penyimpanannya di tempat penyimpanan (misalnya dalam *harddisk* atau pita magnetik).

Pada peringkat logika, setiap *record* dideskripsikan dengan definisi *type*. *Record*-*record* lain dibentuk dengan pola dasar yang telah disediakan di atas. Pemrogram menggunakan bahasa pemrograman dalam pekerjaannya pada peringkat logika ini. Dengan cara yang sama, administrator basis data seringkali bekerja pada peringkat abstraksi ini.

Terakhir, pada peringkat pengguna, pengguna komputer yang relatif awam terhadap teknologi komputer bekerja dengan program aplikasi yang menyembunyikan rincian tipe data di atas. Dengan cara yang sama, pada peringkat pengguna, beberapa pemerhati basis data didefinisikan, dan pengguna basis data hanya melihat basis data dari sudut pandangnya saja, alih-alih melihat keseluruhan isi basis data. Sebagai tambahan, selain penyembunyian rincian dari peringkat logika basis data, peringkat pengguna umumnya juga menyediakan mekanisme perlindungan dan keamanan sehingga pengguna hanya dapat mengakses bagian tertentu dari basis data. Sebagai contoh *teller* pada suatu bank dapat melihat rekening-rekening nasabah, tetapi ia tidak dapat mengetahui gaji karyawan-karyawan lain pada bank yang bersangkutan.

2.3 Model Data

Mendasari struktur basis data adalah model data, yaitu sekumpulan cara/peralatan/*tool* untuk mendeskripsikan data-data, hubungannya satu sama lain, semantiknya, serta batasan konsistensi. Untuk memperlihatkan konsep dari model data, dua model data yang paling sering digunakan adalah ERD (*Entity Relationship Diagram*) dan model relasional. Keduanya menyediakan cara untuk mendeskripsikan perancangan basis data pada peringkat logika. Berikut ini adalah beberapa model basis data yang umum digunakan saat ini.

2.3.1 Model ERD

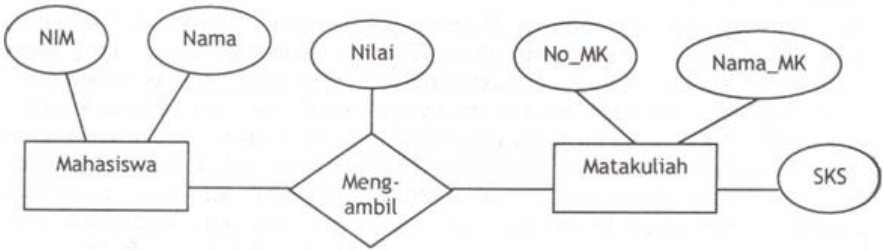
Model data Diagram Hubungan Entitas (ERD - *Entity Relationship Diagram*) dibuat berdasarkan anggapan bahwa dunia

nyata terdiri atas koleksi objek-objek dasar yang dinamakan entitas (*entity*) serta hubungan (*relationship*) antara entitas-entitas itu. Entitas adalah 'sesuatu' atau 'objek' pada dunia nyata yang dapat dibedakan satu terhadap yang lainnya, yang bermanfaat bagi aplikasi yang sedang kita kembangkan. Sebagai contoh, setiap orang adalah entitas dan rekening bank dapat dipertimbangkan sebagai sebuah entitas.

Entitas dalam basis data dideskripsikan berdasarkan atribut-nya. Sebagai contoh, nomor rekening membedakan suatu rekening adalah milik seseorang yang menyimpan uangnya dengan rekening milik orang lain di suatu bank tertentu dan nomor-nomor rekening tersebut merupakan atribut dari entitas rekening yang bersangkutan. Dalam hal ini, nomor rekening secara unik membedakan sebuah rekening dengan rekening yang lainnya. Beberapa rekening mungkin memiliki saldo yang sama, tetapi mereka pasti memiliki nomor rekening yang berbeda.

Relationship adalah hubungan antara beberapa entitas. Sebagai contoh, mahasiswa *memiliki* arangtua. *Memiliki* menjelaskan hubungan tertentu antara mahasiswa dengan orangtuanya. Dalam hal ini, himpunan semua entitas dengan tipe yang sama dan himpunan semua hubungan antarentitas dirujuk sebagai himpunan entitas (*entity set*) dan himpunan relasi (*relationship set*). Secara skematik, basis data dapat dideskripsikan secara gratis dengan ERD yang memiliki komponen-komponen utama sebagai berikut:

- **Empat-persegi-panjang**, yang menggambarkan himpunan entitas.
- **Elips**, yang menggambarkan atribut.
- **Jajaran genjang**, yang menggambarkan relasi/hubungan antarentitas.
- **Garis**, yang menyatukan atribut-atribut pada entitas tertentu serta menyatukan entitas-entitas dalam suatu relasi tertentu.



Gambar 2.2 Penggambaran ERD

Sebagai contoh, perhatikan Gambar 2.2 yang merupakan ERD yang menggambarkan hubungan antara entitas Mahasiswa dengan entitas Matakuliah. Model pada Gambar 2.2 di atas adalah model penggambaran ERD secara umum. Penggambaran itu belum memperlihatkan kardinalitas, yaitu jumlah suatu entitas yang berhubungan dengan entitas yang lainnya.

2.3.2 Model Basis data Relasional

Model basis data relasional menggunakan sekumpulan tabel berdimensi dua (biasa disebut relasi/tabel) yang merupakan tempat data disimpan. Masing-masing tabel dalam model basis data relasional terdiri atas baris dan kolom. Selain itu, pada model basis data relasional juga terdapat istilah *key* (kunci). *Key* adalah satu atau gabungan beberapa atribut yang dapat membedakan semua baris data (*row*) dalam tabel secara unik. Artinya, jika suatu atribut dijadikan *key,* maka tidak boleh ada dua atau lebih baris dengan nilai yang sama untuk atribut/kolom tersebut. Ada tiga macam *key* yang dapat diterapkan pada suatu tabel, yaitu:

- *Superkey*, yaitu merupakan satu atau lebih atribut yang dapat membedakan setiap baris data dalam sebuah tabel secara unik.
- *Candidat-Key*, yaitu merupakan kumpulan atribut minimal yang dapat membedakan setiap baris data dalam sebuah tabel secara unik. Sebuah *candidat-key* tidak boleh berisi atribut atau kumpulan atribut yang telah menjadi *super-key*.
- *Primary-Key*, yaitu merupakan *candidat-key* yang unik yang digunakan sebagai acuan dan kunci utama.

  Perhatikan struktur dan contoh data pada Tabel **Mahasiswa**,

Tabel **Matakuliah**, dan Tabel **Pengambilan Matakuliah** berikut ini.

Tabel 2.1 Tabel **Mahasiswa**

NIM	Nama
123456	Riyanto
123457	Sholihun
123458	Sugiharti

Tabel 2.2 Tabel **Matakuliah**

No_MK	Nama_MK	SKS
110011	Jaringan Syaraf Tiruan	3
120012	Sistem Pendukung Keputusan	2
130013	Teori Bahasa Otomata	4

Tabel 2.3 Tabel **Pengambilan Matakuliah**

NIM	No_MK	Nilai
123456	110011	B
123456	130013	A
123458	110011	B

Ketiga di atas memperlihatkan data-data mahasiswa yang mengambil matakuliah tertentu beserta nilainya masing-masing, misalnya **Riyanto** (NIM=123456) mengambil matakuliah **Jaringan Syaraf Tiruan** (No_MK=110011) dan mendapatkan indeks nilai **B**, dan juga mengambil matakuliah **Sistem Pendukung Keputusan** (No_MK=120012) dan mendapatkan nilai **A**.

Model relasional adalah contoh model berbasis *record*. Ia dinamakan seperti itu sebab basis data memiliki struktur *record* berformat tertentu dimana masing-masing isinya memiliki tipe-tipe yang berbeda (Misalnya tipe data untuk **NIM** adalah string[8] tentu berbeda dengan tipe data untuk **Nama** yang mungkin juga bertipe data string yang panjangnya tidak ditentukan, bergantung pada komputer tempat aplikasi diimplementasikan). Dalam hal ini, setiap kolom pada tabel-tabel mencerminkan atribut-atribut entitas yang bersangkutan yang sering di jumpai di model konseptual ERD.

Dapat dilihat bahwa tabel-tabel dapat disimpan dalam file-file. Sebagai contoh, karakter-karakter khusus, misalnya tanda koma (,) mungkin dapat digunakan untuk memisahkan atribut-atribut

yang berbeda dalam suatu *record*, dan karakter-karakter khusus yang lainnya dapat digunakan untuk memisahkan suatu *record* dengan *record* yang lainnya. Model relasional menyembunyikan implementasi aras rendah basis data dari pengembang aplikasi basis data (pemrogram) serta pengguna. Model relasional adalah abstraksi pada peringkat yang lebih rendah dari ERD. Perancang basis data umumnya pertama kali menggunakan ERD kemudian menerjemah-kannya ke model relasional untuk kemudian diimplementasikan di sistem basis data yang digunakan.

2.3.3 Model Basis Data Herarkhis

Model ini sering dijabarkan dalam bentuk pohon yang menunjukkan hubungan 1-1 (*one to one*) atau *1-M* (*one to many*). Dalam model ini dikenal istilah orang tua (induk atau *parent*) dan anak (*children*). Masing-masing berupa suatu simpul dan terdapat hubungan bahwa setiap anak hanya bisa memiliki satu orang tua (induk) sedangkan orang tua (induk) dapat memiliki sejumlah anak. Model ini tidak mendukung hubungan M-M (*many to many*) karena setiap anak tidak boleh memiliki lebih dari satu orang tua.

2.3.4 Model Basis Data Jaringan

Dalam model jaringan (*network*) tiap entiti dapat mempunyai banyak induk atau banyak anak, sehingga model ini mendukung hubungan M-M (*many to many*). Hal ini lebih fleksibel daripada model hirarkhi. Keuntungan model ini adalah kemampuan untuk menangani antara berbagai *record.*

2.3.5 Model Basis Data Berbasis Objek

Model basis data berbasis objek menggunakan pesan dan objek untuk mengakomodasi jenis data baru dan menyediakan pengembangan penanganan data. Model ini mengizinkan objek untuk menciptakan, merawat, memanipulasi dan memperoleh kembali objek.

2.3.6 Model Basis data Lainnya

Model data berorientasi objek adalah model data lain yang saat ini mulai populer. Model berorientasi objek memperluas ERD dengan penekanan pada pembungkusan (*encapsulation*), metoda (fungsi), serta identitas objek.

Model data objek-relasional mengombinasikan keunggulan-keunggulan model data berorientasi objek dan ketersediaan model

data relasional.

Sesungguhnya ada model-model data yang lainnya yaitu model data jaringan (*network data model*) serta model data hierarki (*hierarchical data model*). Kedua model data ini mendahului model data relasional. Dulu kedua model data ini cukup populer, namun karena keunggulan model data relasional dalam hal kesederhanaannya serta efektivitasnya, perlahan kedua model data ini mulai ditinggalkan.

2.4 Instans dan Skema

Basis data mempunyai informasi yang berubah setiap saat dengan menyisipkan (*insert*) atau menghapus (*delete*) informasi ke dalam basis data. Kumpulan dari informasi yang disimpan dalam basis data dalam suatu saat tertentu disebut **Instant Basis Data**, sedangkan hasil desain dari basis data disebut **Skema Basis Data**.

2.5 Independensi Data

Independensi data adalah kemampuan untuk memodifikasi skema di satu level tanpa mengubah skema di level selanjutnya yang lebih tinggi. Terdapat dua level independensi data, yaitu:

- Independensi fisik yaitu mengubah skema fisik tanpa menyebabkan program aplikasi ditulis ulang.
- Independensi data logis yaitu kemampuan mengubah skema konseptual tanpa mengharuskan program aplikasi ditulis ulang.

Independensi data logis lebih sulit dicapai daripada independensi fisik karena program aplikasi sangat tergantung dengan struktur logika data yang diakses.

2.6 Bahasa Basis Data

Sistem basis data menyediakan bahasa untuk mendefinisikan basis data (*data definition language*) dan memanipulasi basis data (*data manipulation language*) untuk melakukan operasi-operasi tertentu pada basis data. Dalam prakteknya, kedua jenis bahasa basis data itu tidak benar-benar dapat dipisahkan secara tegas. Saat ini keduanya merupakan bagian dari bahasa basis data tunggal yang disebut SQL (*Structured Query Language*) yang merupakan bahasa basis data standar untuk basis data bertipe relasional.

2.6.1 DDL (*Data Definition Language*)

Kita pada umumnya mendefinisikan skema basis data dengan

sekumpulan definisi yang diekspresikan dengan bahasa khusus yang dinamakan DDL (*Data Definition Language*). Sebagai contoh, kita mungkin mendefinisikan tabel **Mahasiswa** dengan cara berikut:

```
create table Mahasiswa
(NIM char(8), Nama char(30), IP real)
```

Eksekusi pernyataan DDL di atas menciptakan tabel Mahasiswa. Sebagai tambahan, pernyataan di atas juga menciptakan apa yang dinamakan kamus data (*data dictionary* atau *data directory*). Kamus data adalah suatu himpunan dari *metadata* (suatu data yang menerangkan data yang lainnya). Skema tabel adalah suatu contoh dari *metadata*. Sistem basis data akan membaca kamus data sebelum membaca atau memodifikasi data yang sebenarnya.

Kita juga menspesifikasi struktur tempat penyimpanan dan metoda akses menggunakan pernyataan khusus DDL yang dinamakan data *storage and definition language*. Pernyataan-pernyataan ini mendefinisikan rincian implementasi skema basis data, yang pada umumnya tersembunyi dari pengguna.

Data yang tersimpan pada basis data biasanya memiliki batasan-batasan (*constrain*) tertentu (Misalnya nilai IP tidak boleh lebih kecil dari 0 dan tidak boleh lebih besar dari 4). Dalam hal ini DDL dapat digunakan untuk menentukan batasan-batasan basis data tersebut, sistem basis data memeriksa batasan-batasan setiap saat basis data disisipkan dan diperbaharui.

2.6.2 DML (*Data Manipulation Language*)

Manipulasi data pada basis data pada umumnya meliputi hal-hal sebagai berikut.

a. Pemanggilan informasi yang tersimpan pada basis data (*query*).
b. Penambahan informasi baru pada basis data.
c. Penghapusan informasi yang tidak diperlukan lagi pada basis data.
d. Modifikasi informasi yang ada pada basis data.

DML (*Data Manipulation Language*) adalah bahasa yang memungkinkan pengguna untuk mengakses atau memanipulasi data dalam sistem basis data yang bertipe relasional. Pada dasarnya ada dua jenis DML, yaitu:

- DML Prosedural yang menghendaki pengguna untuk menspesifikasi data apa yang diperlukan dan bagaimana cara

mendapatkan data itu. Ini dapat dilakukan dengan bahasa-bahasa pemrograman yang mampu mengakses basis data (Misalnya: C+ + atau Java).

- DML Deklaratif (DML Non Prosedural) yang menghendaki pengguna untuk menspesifikasi data apa yang diperlukan tanpa harus menspesifikasi bagaimana caranya mendapat-kannya. Contoh dari DML Non Prosedural ini adalah SOL (*Structured Query Language*).

DML Deklaratif pada umumnya relatif mudah dipelajari dan digunakan dibandingkan DML Prosedural. Tentu saja karena kita tidak harus menspesifikasi bagaimana caranya mendapatkan data yang kita butuhkan, sistem basis data relasional akan menentukan sendiri cara-cara yang efisien untuk mendapatkan data tersebut, menyangkut algoritma yang akan digunakan, strategi-strategi untuk mengoptimalkan kinerja proses, dan sebagainya.

Query adalah pernyataan yang meminta pemanggilan informasi tertentu dari basis data. Sebagian dari DML dinamakan *query language*. Meskipun tidak terlalu tepat, orang sering menyebut DML sebagai bahasa *query*. Suatu pernyataan query untuk menampilkan data-data mahasiswa yang memiliki IP lebih besar atau sama dengan 2.75 adalah sebagai berikut.

```
select *
from Mahasiswa
where IP >= 2.75
```

Perhatikan pada pernyataan SOL di atas, kita hanya menyebutkan informasi-informasi yang dibutuhkan dan tidak menyebutkan bagaimana caranya mendapatkannya. Pada dasarnya pernyataan SOL di atas oleh kompiler akan diterjemahkan ke algoritma-algoritma tertentu yang memungkinkan akses paling efisien pada basis data.

2.7 Pengguna Basis Data dan Administrator

Sasaran utama dari sistem basis data adalah memanggil informasi tertentu dari basis data dan menyimpan informasi baru ke basis data. Orang-orang yang bekerja dengan basis data dapat dikategorikan sebagai pengguna basis data dan administrator.

Ada empat tipe yang berbeda dari pengguna basis data; dibedakan berdasarkan bagaimana caranya mereka berinteraksi dengan sistem basis data. Antarmuka-antarmuka yang berbeda dirancang untuk tipe-tipe pengguna yang berbeda.

1) *Naive User* adalah pengguna yang paling tidak mahir. Mereka berinteraksi dengan basis data dengan menggunakan program-program aplikasi yang telah ditulis sebelumnya. Suatu contoh, seorang *teller* bank yang perlu mentransfer uang sejumlah 1.000.000 rupiah dari rekening A ke rekening B akan memanggil program aplikasi, misalkan *transfer*. Program aplikasi akan bertanya pada *teller* tentang jumlah uang yang perlu ditransfer serta ke rekening mana uang itu perlu ditransfer. Suatu antarmuka yang umum untuk antarmuka pengguna jenis ini adalah form dimana pengguna dapat mengisikan data-data pada *field-field* yang sesuai. Pengguna jenis ini dapat juga secara mudah membaca laporan-laporan yang diciptakan dari basis data.
2) Pemrogram Aplikasi adalah profesional di bidang komputer yang menulis program aplikasi. Pemrogram aplikasi dapat memilih berbagai macam *tool* untuk mengembangkan antarmuka pengguna. Kompiler aplikasi bertipe RAD (*Rapid Application. Development*), misalnya Visual Basic, Visual C++, dan Borland Delphi, memungkinkan pemrogram mengkonstruksi form-form dan laporan-laporan tertentu tanpa perlu menulis program secara rinci seperti pada Java atau C/C++. Dalam hal ini bahasa-bahasa yang mengadopsi konsep-konsep RAD sering dinamakan bahasa generasi ke-4. Selain kemudahannya untuk membuat antarmuka pengguna, bahasa-bahasa ini juga mengandung konsep-konsep pemrograman pada umumnya (runtunan, kondisional, serta perulangan). Kebanyakan sistem basis data komersial saat ini dikembangkan dengan bahasa generasi ke-4 ini.
3) Pengguna canggih berinteraksi dengan sistem tanpa menulis program. Mereka mengakses basis data menggunakan bahasa *query*. Mereka mengirim *query-query* yang ditulis dengan SOL ke *query processor* yang fungsi utamanya adalah untuk memecah pernyataan-pernyataan DML menjadi instruksi-instruksi yang dipahami sistem basis data. Seorang analis yang mengirim *query* untuk mengeksplorasi data dalam basis data termasuk pada golongan ini. OLAP (*Online Analytical Processing*) adalah *tool* yang menyederhanakan pekerjaan analis dengan mengizinkan mereka melihat ikhtisar data dengan berbagai cara. Sebagai contoh, pada basis data penjualan perusahaan misalnya, analis dapat melihat total penjualan per wilayah (misalnya Utara, Timur, Barat, Selatan,

dan Tengah) atau total penjualan produk tertentu per wilayah, dan selanjutnya. Selain OLAP, alat lain yang dapat digunakan untuk mendapatkan hasil yang serupa adalah data mining.

4) Pengguna Terspesialiasi adalah pengguna canggih/mahir yang menulis aplikasi basis data khusus yang tidak termasuk pada pemrosesan data tradisional, misalnya sistem perancangan berbasis komputer (CAD – *Computer Aided Design*), sistem pakar, sistem pendukung keputusan (DSS – *Decision Support System*), Sistem Informasi Geografis (GIS – *Geographical Information System*), dan sebagainya, serta sistem-sistem yang memiliki tipe data kompleks (misalnya data grafis, data suara/audio, animasi, serta gambar bergerak/video).

2.8 Administrator Basis data

Salah satu alasan untuk menggunakan DBMS (*Database Management System*) adalah untuk mendapatkan kendali terpusat pada data maupun program yang mengakses data-data tersebut. Orang yang memiliki tanggung jawab penuh pada sistem basis data sering dinamakan administrator basis data (DBA – Basis data *Administrator*) yang tugas-tugasnya meliputi:

1) Definisi skema. DBA menciptakan skema basis data dengan mengeksekusi pernyataan-pernyataan DDL.
2) Mendefinisikan struktur tempat penyimpanan dan metode aksesnya yang paling efisien.
3) Memodifikasi skema dan organisasi secara fisik. DBA mengatur perubahan-perubahan skema dan organisasi fisik yang mencerminkan perubahan kebutuhan organisasi, atau demi perbaikan kinerja basis data.
4) Memberi hak akses tertentu pada pengguna basis data. Karena ada berbagai jenis pengguna tentu diperlukan juga berbagai hak akses yang mengizinkan pengguna-pengguna hanya mengakses basis data tertentu. Hak akses ini diperlukan demi keamanan organisasi serta demi keamanan data itu sendiri. Seseorang yang tidak berhak mungkin melakukan modifikasi basis data (atau bahkan merusaknya) demi kepentingan pribadi/golongan tertentu. DBA, dalam hal yang telah disebutkan tadi, harus mampu melakukan pencegahan. Salah satu caranya adalah memberi hak-hak akses tertentu bagi setiap individu pengakses basis data.

5) Melakukan pemeliharaan rutin. Tugas-tugas rutin ini adalah tugas yang sangat penting sehingga tidak bisa diabaikan begitu saja. Contoh-contoh tugas rutin DBA adalah:
   - Secara periodik membuat salinan basis data (*backup*), baik ke pita magnetik atau ke server jarak jauh, untuk mencegah kehilangan data akibat perusakan yang disengaja maupun tidak.
   - Memastikan bahwa tempat penyimpanan (misalnya *hardisk*) masih cukup untuk operasi-operasi yang normal. Jika diperlukan, DBA mungkin melakukan *upgrade* tempat penyimpanan.
   - Memantau kinerja basis data. Ini sangat penting saat begitu banyak pengguna melakukan modifikasi-modifikasi basis data yang pada gilirannya menurunkan kinerja basis data.

Dalam prakteknya, organisasi-organisasi besar juga mempekerjakan personal-personal yang punya tanggung jawab lebih luas dari DBA. Personal-personal ini (yang dinamakan administrator data) memiliki tanggung jawab lebih besar dari DBA dan tekanan tanggungjawabnya adalah lebih pada perencanaan basis data alih-alih implementasinya yang lebih merupakan tanggung jawab DBA.

2.9 Keunggulan Menggunakan Sistem Basis Data

Pada bagian ini akan dibahas keunggulan-keunggulan sistem basis data bertipe relasional dibandingkan sistem pemrosesan berkas, meskipun begitu, mencakup keunggulan-keunggulan ini juga tipe-tipe sistem basis data yang lainnya.

2.9.1 Keunggulan Sistem Relasional

Pendekatan basis data menawarkan keunggulan-keunggulan dibandingkan sistem pemrosesan berkas tradisional. Keunggulan-keunggulan itu adalah:

1) Kemandirian Program dan Data. Pemisahan deskripsi data (*meta data*) dari program aplikasi yang menggunakan data tersebut dinamakan kemandirian data (*data independence*). Dengan pendekatan basis data, deskripsi data disimpan di lokasi terpusat yang dinamakan *repository*. Karakteristik sistem basis data ini mengizinkan organisasi data berubah (sampai batas-batas tertentu) tanpa mempengaruhi program aplikasi yang memproses data tersebut (tidak memerlukan pemrograman ulang – *reprogramming*) .

2) Mengurangi Pengulangan Data (Redundansi) yang Tidak Perlu. Sasaran perancangan dengan pendekatan basis data adalah menyatukan berkas-berkas data pada suatu struktur logika yang tunggal. Setiap fakta primer direkam pada hanya satu tempat di basis data. Misalnya, dari contoh di atas, IP seorang mahasiswa hanya tersimpan pada *Tabel Mahasiswa*. Pendekatan basis data tentu saja tidak menghilangkan redundansi data sama sekali, tetapi ia mengizinkan perancang secara hati-hati mengendalikan redundansi. Sebagai contoh (Lihat Gambar 1.4) setiap NIM pada *Tabel Pengambilan Matakuliah* harus berpasangan dengan NIM pada *Tabel Mahasiswa*. Hal ini menegaskan bahwa ada relasi antara *Tabel Mahasiswa* dan *Tabel Pengambilan Matakuliah*.
3) Memperbaiki Konsistensi Data. Dengan mengendalikan redundansi, kita secara dramatis mengurangi kesempatan untuk terjadinya tidak konsistennya data. Misalnya, pada *Tabel Mahasiswa*, setiap NIM berpasangan dengan nama mahasiswa tertentu. Jika kita ingin mengubah nama mahasiswa tertentu, kita cukup melakukannya hanya pada *Tabel Mahasiswa*. Pada *Tabel Pengambilan Matakuliah*, tidak ditemukan kolom nama sehingga – seperti diungkapkan sebelumnya -- kita tidak perlu mengubah nama mahasiswa pada *Tabel Pengambilan Matakuliah* sebab hal itu telah terwakili dengan perubahan nama pada *Tabel Mahasiswa*.
4) Memperbaiki Kesempatan Berbagi Data (*Data Sharing*). Basis data dirancang untuk berbagi sumberdaya data dalam organisasi. Pengguna dengan hak tertentu (yang diatur sebelumnya oleh administrator basis data) dapat mengakses bagian tertentu dalam basis data, di manapun pengguna tersebut berada dalam organisasi.
5) Menambah Produktivitas Pengembangan Program Aplikasi. Salah satu keunggulan pendekatan basis data adalah pengurangan waktu dan biaya untuk mengembangkan aplikasi bisnis yang baru. Ada dua alasan yang memungkinkan aplikasi basis data dikembangkan dengan cara yang lebih cepat dibandingkan aplikasi yang mengakses berkas data pada pendekatan tradisional, yaitu:
    - Dengan mengasumsikan bahwa basis data telah dirancang dan diimplementasikan, pemrogram cukup berkonsentrasi pada fungsi spesifik yang dibutuhkan oleh aplikasi baru tanpa perlu memusingkan perancangan

berkas data atau rincian implementasi aras rendah.

- Sistem manajemen basis data menyediakan sejumlah *tool* seperti pembuatan form dan generator laporan (*report generator*) dan bahasa tingkat tinggi dapat mengotomatisasi beberapa aktivitas perancangan basis data dan implementasinya.

6) Memaksakan Standar. Saat pendekatan basis data diimplementasikan dengan dukungan manajemen secara penuh, administrasi data dan administrasi basis data dapat dilakukan oleh seseorang yang memang ditunjuk untuk hal itu. Dengan pengelolaan basis data pada satu tangan, ia dapat diberi tanggung jawab untuk memaksakan standar basis data untuk keseluruhan organisasi. Standar yang dimaksud antara lain: tata cara penamaan, standar kualitas data, serta prosedur yang seragam untuk mengakses, memperbaharui, serta melindungi data. *Repository* memungkinkan administrator basis data memaksakan standar itu dengan sekumpulan perkakas yang berdaya guna.
7) Memperbaiki Kualitas Data. Kualitas data yang rendah selama ini telah menjadi pusat perhatian dalam organisasi-organisasi skala besar. Pendekatan basis data menyediakan sejumlah kakas dan proses untuk memperbaiki kualitas data. Dua di antaranya adalah:
   - Perancang basis data dapat menspesifikasi batasan-batasan integritas (untuk memelihara konsistensi data) yang dapat dipaksakan oleh sistem basis data.
   - Salah satu sasaran dari data *warehousing* adalah meletakkan hanya data-data yang berkualitas pada data *warehouse*.
8) Memperbaiki Akses Data. Dengan basis data bertipe relasional, pengguna tanpa pengetahuan dan pengalaman pemrograman dapat memanggil dan menampilkan data-data tertentu sesuai hak yang dimilikinya, meskipun cara mendapatkannya mungkin melewati batas-batas departemen dimana pengguna itu berada. Suatu cara untuk melakukannya adalah hanya dengan menuliskan pernyataan-pernyataan SOL (*Structure Query Language*) yang implementasinya dilakukan oleh sistem basis data, tanpa pengguna perlu mengetahui teknik-teknik algoritma-algoritma yang diadaptasi dan diimplementasi sistem basis data untuk mendapatkan data-data itu.

9) Mengurangi Biaya Pemeliharaan Program. Data yang tersimpan harus diubah karena beberapa alasan, seperti: tipe data baru ditambahkan, format data berubah, dan sebagainya. Pada sistem pemrosesan berkas, deskripsi data dan logika untuk mengakses data bersatu dalam suatu program aplikasi. Akibatnya, perubahan format data dan metoda akses data akan mengakibatkan pemrogram harus mengubah program.

2.9.2 Manajemen Transaksi

Seringkali beberapa operasi pada basis data membentuk suatu unit tunggal dari pekerjaan. Suatu contohnya adalah pentransferan dana dari rekening A ke rekening B. Dalam hal ini, sistem akan melakukan pengurangan dana dari rekening A (didebit) serta penambahan dana pada rekening B (dikredit). Di sini dapat dilihat bahwa kedua operasi itu harus benar-benar dilakukan (atau tidak dilakukan sama sekali). Fenomena ini dinamakan **atomisitas**. Sebagai tambahan, pentransferan dana tadi harus tetap dilakukan dengan mempertahankan **konsistensi** basis data, yaitu tetap menjaga kondisi basis data tetap seperti semula tanpa kehilangan (atau terjadi perubahan) pada *field-field* lainnya (kecuali saldonya, tentu saja). Terakhir, setelah eksekusi proses ganda yang sukses tadi, nilai-nilai baru rekening A dan B harus benar-benar mencerminkan operasi-operasi yang telah terjadi dengan mempertimbangkan kemungkinan kegagalan sistem. Hal ini dinamakan **durabilitas**.

Transaksi adalah sekumpulan operasi yang melakukan suatu fungsi logika tunggal pada aplikasi basis data, misalnya pentransferan dana dari rekening A ke rekening B di atas. Transaksi harus tetap menjaga prinsip *atomisitas* dan *konsistensi*, dalam arti transaksi tidak melanggar batasan konsistensi basis data. Ini berarti – seperti dijelaskan di atas – kondisi basis data harus tetap konsisten sebelum dan sesudah eksekusi transaksi.

Merupakan tugas pemrogram untuk secara benar mendefinisikan berbagai transaksi dengan mempertahankan konsistensi basis data. Sebagai contoh pentransferan dana dari rekening A ke rekening B mengandung dua program yang terpisah, yaitu: 1) mendebit rekening A, dan 2) mengkredit rekening B. Eksekusi 2 program di atas sekaligus menjaga konsistensi basis data. Jika hanya satu program yang berjalan, maka fungsi itu bukanlah sebuah transaksi.

Sebagai komponen manajemen transaksi, sistem basis data (DBMS) akan memastikan karakteristik *atomisitas* dan *durabilitas*. Pada sistem tanpa kegagalan, semua transaksi diselesaikan secara

sempurna dan *atomisitas* dicapai dengan mudah. Namun, dengan terjadinya kegagalan sistem, transaksi tidak selalu berakhir dengan sempurna. Dalam hal ini basis data harus kembali ke keadaan sebelum transaksi dimulai. Basis data harus melakukan apa yang dinamakan pemulihan dari kegagalan (*failure recovery*) dimana suatu fungsi tertentu mendeteksi kegagalan sistem kemudian mengembalikan keadaan basis data ke keadaan sebelum terjadinya kegagalan sistem.

Terakhir, ketika beberapa transaksi memperbarui basis data secara mandiri (konkuren), konsistensi data mungkin tidak dapat dipertahankan, meskipun setiap transaksi individual adalah benar. Adalah tanggung jawab. *concurrency-control manager* untuk mengendalikan interaksi antara transaksi yang konkuren untuk memastikan konsistensi basis data.

Sistem basis data yang dirancang untuk suatu komputer pribadi (PC - *Personal Computer*) mungkin tidak memiliki keseluruhan fasilitas di atas. Sebagai contoh, kebanyakan sistem kecil hanya mengizinkan satu pengguna untuk mengakses satu basis data pada suatu waktu tertentu (misalnya yang terjadi pada Microsoft Access). Kekuran lain dari sistem kecil adalah tidak adanya fasilitas *backup* serta pemulihan (*restore*) data. Pembatasan-pembatasan ini berkaitan dengan sumberdaya yang dimiliki komputer, terutama memori. Namun, sistem basis data untuk perusahaan besar yang memiliki kompleksitas tinggi seharusnya memiliki semua fasilitas yang kita bahas sebelumnya demi kelangsungan operasional perusahaan yang bersangkutan.

2.9.3 Struktur Sistem Basis Data

Sistem basis data dibagi ke dalam modul-modul yang masing-masing memiliki tanggung jawab bagi keseluruhan sistem. Komponen fungsional dari sistem basis data secara garis besar dapat dibagi dua, yaitu: manager penyimpanan dan komponen *query*.

- Manajer penyimpanan (*storage manager*) sangat penting sebab basis data secara tipikal membutuhkan sejumlah besar ruang penyimpanan. Basis data perusahaan ukurannya bervariasi dari ratusan *byte* hingga *gigabyte* (untuk basis data yang sangat besar bisa mencapai *terabyte*). *Gygabyte* adalah 1000 *megabyte* (1 milyar *byte*) dan *terabyte* adalah 1 juta *megabyte* (1 triliun *byte*). Karena memori utama komputer tidak dapat menyimpan informasi sebanyak itu, informasi disimpan di *hardisk*. Data berpindah dari *hardisk* ke memori

saat data tersebut dibutuhkan. Karena kecepatan perpindahan data tersebut sangat bergantung pada kecepatan CPU, struktur sistem basis data sangat menentukan untuk meminimalisasi kebutuhan untuk memindahkan data dari *hardisk* ke memori utama.

- *Query processor* juga sangat penting sebab ia membantu sistem basis data untuk menyederhanakan dan memfasilitasi akses pada data. Cara pandang tingkat tinggi (*high level view*) membantu mencapai sasaran, dengannya pengguna sistem tidak secara langsung mengetahui rincian implementasi fisik dari sistem. Bagaimanapun juga, kecepatan pemrosesan dari pembaharuan data (*updating*) dan *query* adalah sangat penting.

  Merupakan tugas dari sistem basis data untuk menerjemah-kan perintah pembaharuan dan *query* yang ditulis dengan bahasa nonprosedural (pada peringkat logika) ke urutan operasi dengan algoritma-algoritma tertentu yang efisien pada peringkat fisik.

2.9.3.1 Storage Manager

Storage manager (manajer penyimpanan) adalah modul-modul program yang menyediakan antarmuka antara data peringkat rendah yang tersimpan di basis data dan program-program aplikasi serta *query* yang dikirimkan ke sistem. Manajer penyimpanan bertanggungjawab untuk interaksi dengan manajer berkas (*file manager*). Data mentah disimpan di *hardisk* menggunakan sistem berkas, yang selalu disediakan oleh sistem operasi konvensional (*operating system*). Manajer penyimpanan menerjemahkan berbagai pernyataan DML ke perintah sistem berkas peringkat rendah. Maka, manajer penyimpanan bertanggungjawab dalam proses penyimpanan (*storing*), pemanggilan kembali (*retrieving*), dan pembaharuan (*updating*) data pada basis data. Komponen manajer penyimpanan meliputi:

- Manajer Otoritas dan Integritas. Tugas manajer ini adalah menguji batasan integritas serta memeriksa keabsahan pengguna yang menggunakan basis data.
- Manajer Transaksi. Tugasnya adalah memastikan basis data tetap pada kondisi yang konsisten saat terjadi kegagalan sistem, serta memastikan transaksi yang terjadi secara konkuren dieksekusi tanpa terjadi konflik-konflik.
- Manajer Berkas. Tugasnya mengelola alokasi ruang pada

tempat penyimpanan dan mengelola struktur data yang digunakan untuk menyimpan informasi di *hardisk*.

- Manajer Penyangga (*Buffer Manager*). Bertanggungjawab untuk melakukan pemanggilan data dari *hardisk* ke memori serta memutuskan data mana yang tetap disimpan di memori utama (baca: data-data residen) untuk kecepatan akses (*caching*). Manajer penyangga adalah bagian yang penting dari sistem basis data, karena ia memungkinkan basis data menangani ukuran data yang lebih besar dari ukuran memori utama.

Manajer penyimpanan menerapkan beberapa struktur data sebagai bagian dari implementasi sistem secara fisik. Beberapa struktur data itu adalah:

- Berkas Data, yang menyimpan basis data itu sendiri.
- Kamus Data (*Data Dictionary*), yang menyimpan metadata tentang struktur basis data, skema dari basis data.
- Indeks, yang menyediakan akses cepat ke data.

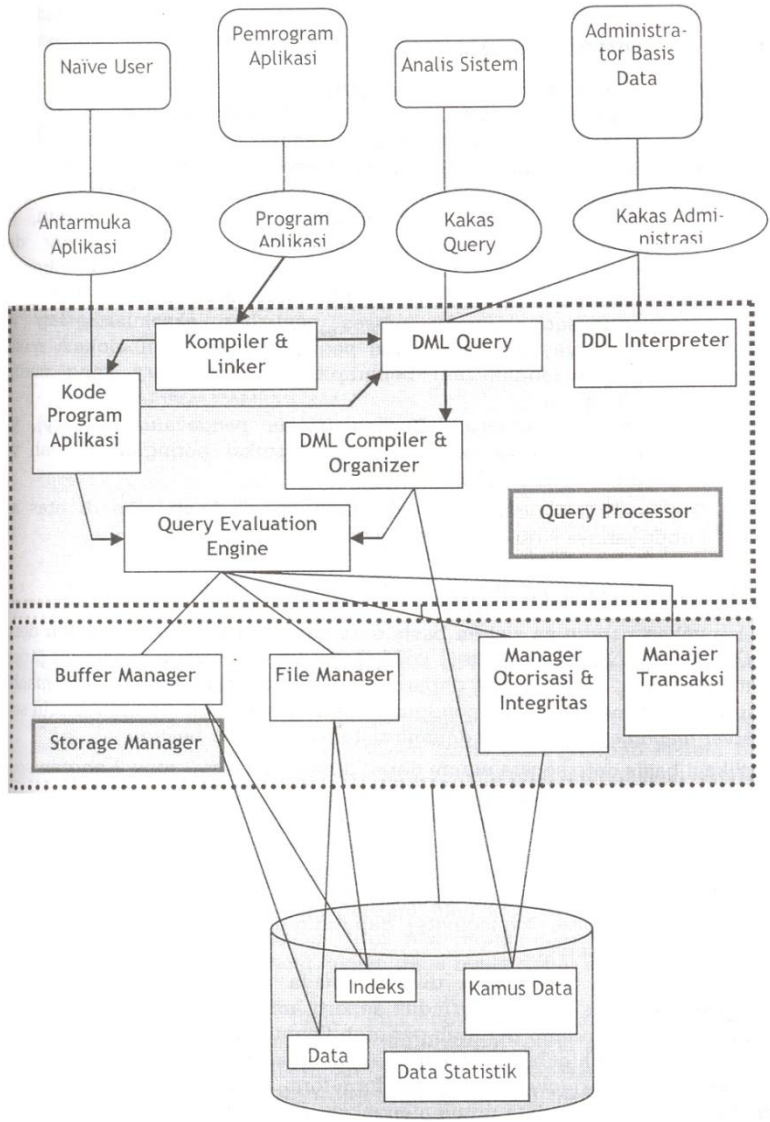
2.9.3.2 Query Processor

Sebagaimana telah disinggung sebelumnya, fungsi *query processor* secara umum adalah menerjemahkan *query* yang dikirim pengguna ke sistem basis data, untuk selanjutnya sistem basis data mengembalikan *output* sesuai *query* yang diproses. Komponen *query processor* mencakup:

- **Penerjemah DDL**, yang menafsirkan pernyataan DDL dan merekam definisi-definisinya di kamus data.
- **DML Compiler**, yang menerjemahkan pernyataan DML dalam *query* ke rencana evaluasi yang mengandung instruksi-instruksi peringkat rendah dengan algoritma-akgoritma tertentu, yang dimengerti oleh mesin pengevaluasi *query*. *Query* dapat diterjemahkan ke beberapa alternatif eksekusi yang memberi hasil yang sama. DML Compiler juga melakukan optimasi *query* (*query optimization*), yaitu proses pemilihan eksekusi query yang berbiaya (dipandang dari segi kecepatan dan alokasi memori paling rendah dari beberapa alternatif *query* yang mungkin dikerjakan.

- **Query Evaluation Engine** (mesin pengevaiuasi *query*), yang melakukan eksekusi instruksi-instruksi peringkat rendah yang dihasilkan oleh DML Compiler.

Gambar 2.3 berikut ini memperlihatkan semua komponen di atas serta hubungan antara komponen satu dengan yang lain.



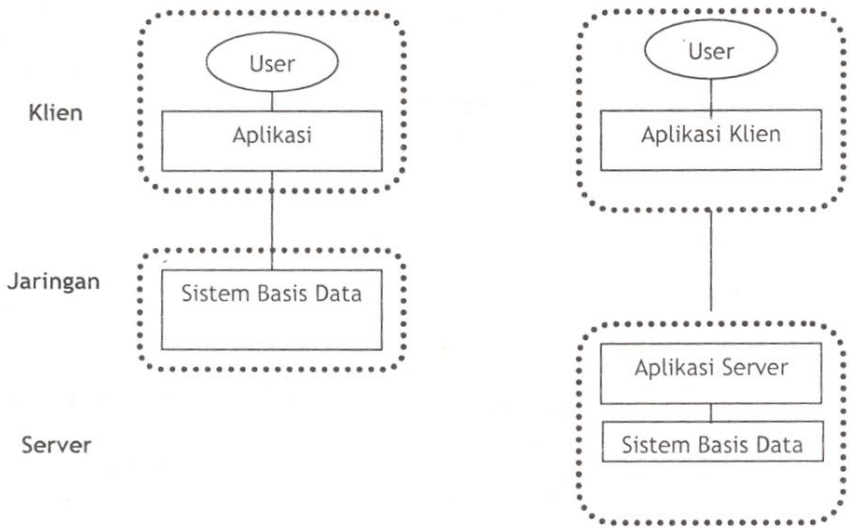
Gambar 2.3 Struktur Sistem Basis Data

2.9.4 Arsitektur Aplikasi

Kebanyakan pengguna sistem basis data saat ini tidak terlalu peduli dengan letak sesungguhnya dari sistem basis data, mereka terhubung lewat jaringan antar-komputer. Saat ini kita dapat memisahkan mesin klien (*client machine*) -- yaitu tempat/mesin di mana pengguna basis data bekerja – dengan mesin server (*server machine*) -- yaitu mesin di mana sistem basis data bekerja.

Aplikasi basis data secara umum dapat dibagi menjadi dua atau tiga bagian seperti yang terlihat pada Gambar 2.4 di bawah. Pada arsitektur 2 lapis (*two-tier architecture*), aplikasi-aplikasi terbagi menjadi komponen-komponen yang berada di mesin klien, yang meminta fungsionalitas sistem basis data yang berada pada mesin server melalui pernyataan-pernyataan bahasa *query*. Antarmuka program aplikasi seperti ODBC (*Open Database Connectivity*) dan JDBC (*Java Database Connectivity*) dapat digunakan sebagai sarana interaksi antara klien dan server.

Kontras dengan pendekatan di atas, pada arsitektur 3 lapis (*three-tier architecture*), mesin klien bertindak seperti antarmuka pengguna (*front-end*) dan tidak mengandung perintah-perintah pemanggilan langsung pada sistem basis data. Alih-alihnya, klien berkomunikasi dengan server aplikasi (*application server*) lewat antarmuka form-form. Server aplikasi berkomunikasi dengan sistem basis data untuk mengakses data. Aplikasi 3 lapis lebih sesuai untuk aplikasi-aplikasi besar dan untuk aplikasi-aplikasi yang berjalan *World Wide Web*.



Gambar 2.4 Arsitektur 2 Lapis (Kiri) dan 3 Lapis (Kanan)

Bab 3

Pemodelan Proses Perangkat Lunak

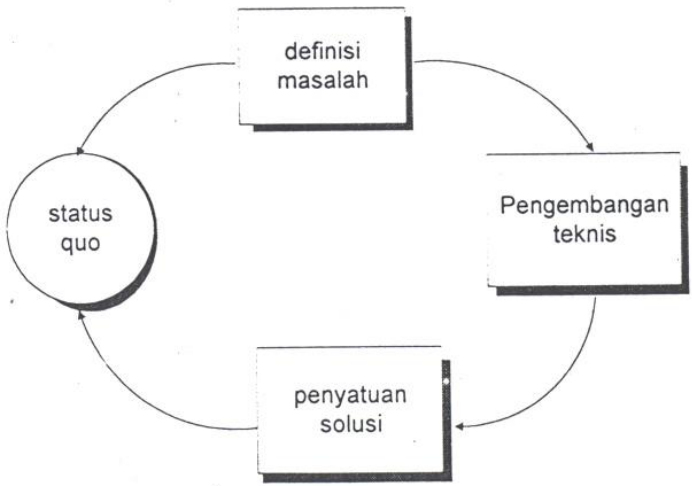
Untuk menyelesaikan masalah aktual di dalam sebuah seting industri, rekayasa perangkat lunak atau tim perekayasa harus menggabungkan strategi pengembangan yang melingkupi lapisan proses, metode, dan alat-alat bantu. Hal tersebut dapat dilihat pada Gambar 3.1 serta fase-fase generik berikut ini.

- Fase Definisi (*Definition Phase*) berfokus pada "apa" (*what*). Pada fase definisi ini pengembang perangkat lunak harus mengidentifikasi informasi apa yang akan diproses, fungsi dan unjuk kerja apa yang dibutuhkan, tingkah laku sistem seperti apa yang diharapkan, *interface* apa yang akan dibangun, batasan desain apa yang ada, dan kriteria validasi apa yang dibutuhkan untuk mendefinisikan sistem yang sukses. Kebutuhan (*requirement*) dari sistem dan perangkat lunak yang didefinisikan. Metode yang diaplikasikan selama fase definisi berbeda, tergantung pada pradigma rekayasa perangkat lunak (atau kombinasi paradigma) yang diaplikasikan. Ada tiga tugas utama yang berada dalam bentuk yang sama, sistem atau rekayasa informasi, perencanaan proyek perangkat lunak, serta analisis kebutuhan.
- Fase Pengembangan (*Development Phase*) berfokus pada "bagaimana" (*how*). Pada fase ini selama masa pengembangan perangkat lunak, teknisi harus mendefinisikan bagaimana data dikonstruksikan, bagaimana fungsi-fungsi diimplementasikan sebagai sebuah arsitektur perangkat lunak, bagaimana detail prosedur akan diimplementasikan, bagaimana *interface* ditandai (dikarakterisasi), bagaimana rancangan akan diterjemahkan ke dalam bahasa pemrograman (atau bahasa non-prosedural), serta bagaimana pengujian akan dilakukan. Metode-metode yang akan diaplikasikan selama masa pengembangan program akan bervariasi, tetapi

ada tiga tugas teknis yang khusus yang harus selalu ada, yaitu: rancangan perangkat lunak, pemunculan kode, dan pengujian perangkat lunak.

- Fase Pemeliharaan (*Maintenance Phase*) berfokus pada perubahan (*change*). Pada fase ini dihubungkan dengan koreksi kesalahan, penyesuaian yang dibutuhkan ketika lingkungan perangkat lunak berkembang, serta perubahan sehubungan dengan perkembangan yang disebabkan oleh perubahan kebutuhan pelanggan. Fase pemeliharaan mengaplikasikan lagi langkah-langkah pada fase definisi dan fase pengembangan, tetapi semuanya tetap tergantung pada konteks perangkat lunak yang ada.

Strategi ini sering diacukan sebagai *model proses* atau *paradigma rekayasa perangkat lunak.* 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. Di dalam sebuah paper yang penuh intrik tentang sifat proses perangkat lunak, L.B.S. Racoon menggunakan fraktal sebagai dasar untuk membahas sifat proses perangkat lunak yang sebenarnya.

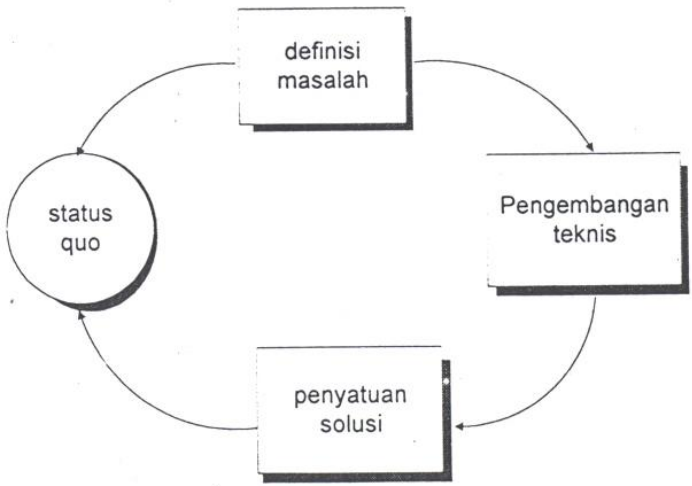


Gambar 3.1 Lapisan Rekayasa Perangkat Lunak

Perkembangan perangkat lunak bisa dianggap sebagai lingkaran pemecahan masalah (Gambar 3.2) di mana terdapat empat keadaan berbeda, yaitu status quo, definisi masalah, perkembangan teknis memecahkan masalah di keseluruhan aplikasi dari banyak teknologi, dan integrasi pemecahan menyampaikan hasil (contohnya

dokumen, program, data, fungsi bisnis baru, produk baru) kepada siapa yang membutuhkan pertama kali. Fase rekayasa perangkat lunak generik serta langkah-langkahnya dengan mudah terpetakan di dalam keadaan-keadaan tersebut.

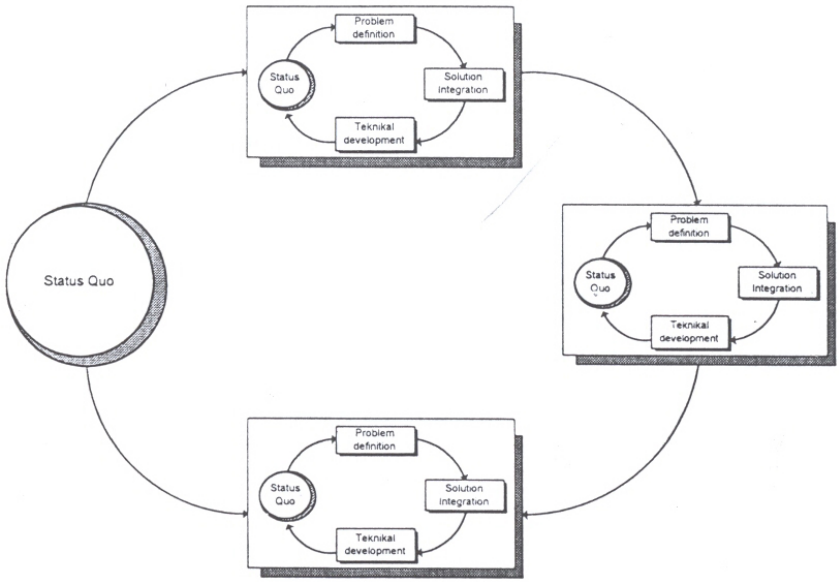
Lingkaran pemecahan masalah seperti yang digambarkan di atas berlaku untuk kerja rekayasa perangkat lunak pada tingkat resolusi yang berbeda. Lingkaran tersebut dapat dipakai pada tingkat makro ketika bagian dalam aplikasi dipakai, pada tingkat tengah jika komponen program direkayasa, dan bahkan pada garis tingkat kode sekalipun. Dengan demikian, pewakilan fraktal dapat dipakai untuk memberikan pandangan proses yang ideal. Dalam Gambar 3.3, masing-masing keadaan di dalam lingkaran pemecahan masalah berisi lingkaran pemecahan masalah lain yang identik, yang juga masih berisi lingkaran pemecahan masalah yang lain (hal ini berlanjut sampai beberapa batas yang rasional, bagi perangkat lunak, garis kode).



Gambar 3.2 Fase lingkaran pemecahan masalah

Secara realistis memang sulit untuk mengadakan penggolongan aktivitas seteratur yang dinyatakan di dalam Gambar 3.3, karena adanya silang pendapat. Di luar keadaan tersebut, pandangan singkat ini membawa kita kepada sebuah gagasan yang sangat penting. Tanpa mempedulikan model proses yang dipilih untuk proyek perangkat lunak, semua keadaan ini (status quo), definisi masalah, pengembangan teknis, dan integrasi pemecahan, secara simultan hidup berdampingan pada beberapa tingkat detail. Misalkan saja sifat rekursif Gambar 3.3, keempat keadaan yang

didiskusikan di atas berlaku secara sama pada analisis dari sebuah aplikasi lengkap dan pada pemunculan sebuah segmen kode yang kecil.



Gambar 3.3 Fase-fase di dalam fase lingkaran pemecahan masalah

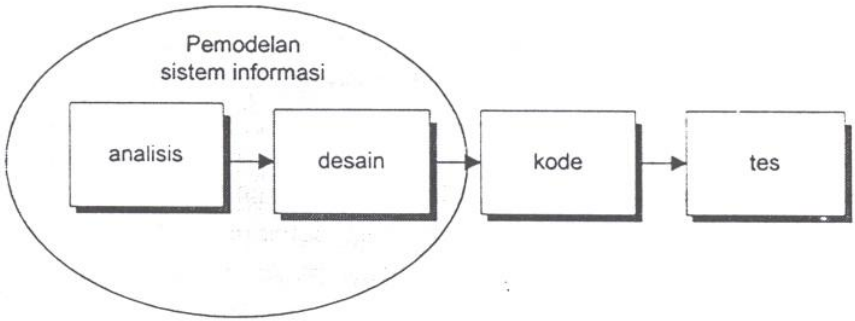
Raccoon mengusulkan sebuah "model Chaos" yang menggambarkan "perkembangan perangkat lunak (sebagai) sebuah kesatuan dari pemakai ke pengembang dan ke teknologi." Pada saat kerja bergerak maju menuju sebuah sistem yang lengkap, keadaan yang digambarkan di atas secara rekursif diaplikasikan kepada kebutuhan pemakai dan spesifikasi teknis perangkat lunak pengembang.

Di dalam bab ini dan selanjutnya didiskusikan bermacam-macam model proses yang berbeda pada perangkat lunak. Masing-masing mewakili sebuah keharusan untuk membawa orde kepada kegiatan yang bersifat kacau yang inheren. Penting untuk diingat bahwa masing-masing model sudah ditandai dengan cara tertentu sehingga (diharapkan) bisa membantu di dalam kontrol dan koordinasi dari proyek perangkat lunak yang nyata. Dengan demikian, pada intinya, semua model menunjukkan karakteristik dari model Chaos.

3.1 Model Sekuensial Linier

Gambar 3.4 menggambarkan *sekuensial linier* untuk rekayasa perangkat lunak, yang sering disebut juga dengan "siklus kehidupan

klasik" atau "model air terjun."



Gambar 3.4 Mode Sekuensial Linier

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. Dimodelkan setelah siklus rekayasa konvensional, model sekuensial linier melingkupi aktivitas-aktivitas sebagai berikut:

- **Rekayasa dan pemodelan sistem/informasi**. Karena perangkat lunak selalu merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke perangkat lunak tersebut. Pandangan sistem ini penting ketika perangkat lunak harus berhubungan dengan elemen-elemen yang lain seperti perangkat lunak, manusia, dan basis data. Rekayasa dan analisis sistem menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis serta desain tingkat puncak. Rekayasa informasi mancakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.
- **Analisis kebutuhan perangkat lunak**. Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khususnya pada perangkat lunak. Untuk memahami sifat program yang dibangun, perekayasa perangkat lunak (analis) harus memahami domain informasi, tingkah laku, unjuk kerja, dan antarmuka (*interface*) yang diperlukan. Kebutuhan baik untuk sistem maupun perangkat lunak didokumentasikan dan dilihat lagi dengan pelanggan.
- **Desain**. Desain perangkat lunak sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah

program yang berbeda, struktur data, arsitektur perangkat lunak, representasi interface, dan detail (algoritma) prosedural. Proses desain menerjemahkan syarat kebutuhan ke dalam sebuah representasi perangkat lunak yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi perangkat lunak.

- **Generasi Kode**. Desain harus diterjemahkan ke dalam bentuk mesin yang bisa dibaca. Langkah pembuatan kode melakukan tugas ini. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.
- **Pengujian**. Sekali kode dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal perangkat lunak, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan-kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.
- **Pemeliharaan**. Perangkat lunak akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah perangkat lunak yang dilekatkan). Perubahan akan terjadi karena kesalahan-kesalahan ditentukan, karena perangkat lunak harus disesuaikan untuk mengakomodasi perubahan-perubahan di dalam lingkungan eksternalnya (contohnya perubahan yang dibutuhkan sebagai akibat dari perangkat *peripheral* atau sistem operasi yang baru), atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan perangkat lunak mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.

Model sekuensial linier adalah paradigma rekayasa perangkat lunak yang paling luas dipakai dan paling tua. Akan tetapi, kritik dari paradigma tersebut telah menyebabkan dukungan aktif untuk mempertanyakan kehandalannya. Masalah-masalah yang kadang-kadang terjadi ketika model sekuensial linier diaplikasikan adalah:

1. Jarang sekali proyek nyata mengikuti aliran sekuensial yang dianjurkan oleh model. Meskipun model linier bisa mengakomodasi iterasi, model itu melakukannya dengan cara tidak langsung. Sebagai hasilnya, perubahan-perubahan dapat menyebabkan keraguan pada saat tim proyek berjalan.

2. Kadang-kadang sulit bagi pelanggan untuk menyatakan semua kebutuhannya secara eksplisit. Model linier sekuensial memerlukan hal ini dan mengalami kesulitan untuk mengakomodasi ketidakpastian natural yang ada pada bagian awal beberapa proyek.
3. Pelanggan harus bersikap sabar. Sebuah versi kerja dari program-program itu tidak akan diperoleh sampai akhir waktu proyek dilalui. Sebuah kesalahan besar, jika tidak terdeteksi sampai program yang bekerja tersebut dikaji ulang.
4. Pengembang sering melakukan penundaan yang tidak perlu. Di dalam analisis yang menarik tentang proyek aktual, Bradac mendapatkan bahwa sifat alami dari siklus kehidupan klasik membawa kepada *blocking state* di mana banyak anggota tim proyek harus menunggu tim yang lain untuk melengkapi tugas yang saling memiliki ketergantungan. Kenyataannya, waktu yang dipakai untuk menunggu bisa mengurangi waktu untuk usaha produktif. *Blocking state* cenderung menjadi lebih lazim pada awal dan akhir sebuah proses sekuensial linier.

Masing-masing dari masalah tersebut brsifat riil. Akan tetapi, paradigma siklus kehidupan klasik memiliki tempat yang terbatas namun penting di dalam kerja rekayasa perangkat lunak. Paradigma itu memberikan *template* di mana metode analisis, desain, pengkodean, pengujian, dan pemeliharaan bisa dilakukan. Siklus kehidupan klasik tetap menjadi model bagi rekayasa perangkat lunak yang paling luas dipakai. Sekalipun memiliki kelemahan, secara signifikan dia lebih baik daripada pendekatan yang sifatnya sembrono kepada pengembangan perangkat lunak.

3.2 Model Prototipe

Sering seorang pelanggan mendefinisikan serangkaian sasaran umum bagi perangkat lunak, tetapi tidak melakukan identifikasi kebutuhan *output*, pemrosesan, ataupun input detail. Pada kasus yang lain, pengembang mungkin tidak memiliki kepastian terhadap efisiensi algoritma, kemampuan penyesuaian dari sebuah sistem operasi, atau bentuk-bentuk yang harus dilakukan oleh interaksi manusia dengan mesin. Dalam hal ini, serta pada banyak situasi yang lain, *prototyping paradigma* mungkin menawarkan pendekatan yang terbaik.

Prototyping paradigma (Gambar 3.5) dimulai dengan pengumpulan kebutuhan. Pengembang dan pelanggan bertemu dan mendefinisikan objektif keseluruhan dari perangkat lunak, mengidentifikasi segala kebutuhan yang diketahui, dan area garis besar di mana definisi lebih jauh merupakan keharusan kemudian dilakukan "perancangan kilat". Perancangan kilat berfokus pada penyajian dari aspek-aspek perangkat lunak tersebut yang akan nampak bagi pelanggan/pemakai (contohnya pendekatan *input* dan format *output*). Perancangan kilat membawa kepada konstruksi sebuah prototipe. Prototipe tersebut dievaluasi oleh pelanggan/-pemakai dan dipakai untuk menyaring kebutuhan pengembangan perangkat lunak. Iterasi terjadi pada saat prototipe disetel untuk memenuhi kebutuhan pelanggan dan pada saat yang sama memungkinkan pengembang untuk memahami apa yang harus dilakukannya.

Gambar 3.5 Prototipe Paradigma

Secara ideal prototipe berfungsi sebagai sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Bila prototipe yang sedang bekerja dibangun pengembang harus mempergunakan fragmen-fragmen program yang ada atau mengaplikasikan alat-alat bantu (contohnya *report generator*, *window manager*, dan lain-lain) yang memungkinkan program bekerja untuk dimunculkan secara cepat.

Akan tetapi, apa yang kita lakukan dengan prototipe tersebut pada saat dia sudah melayani usulan yang digambarkan di atas? Brooks memberikan Jawabannya:

> "Pada sebagian besar proyek, sistem pertama yang dibangun baru saja bisa dipergunakan. Sistem mungkin terlalu pelan, terlalu besar, janggal di dalam pemakaian, atau bahkan ketiganya. Tidak ada altematif lain selain mulai lagi, tidak dengan halus tetapi dengan lebih halus lagi, dan membangun sebuah versi yang dirancang kembali di mana masalah-masalah tersebut bisa diselesaikan... Ketika sebuah konsep sistem baru atau teknologi baru dipergunakan, seseorang harus membangun sebuah sistem sebagai pembuangan, bahkan untuk perencanaan terbaik sekalipun, tidak akan mudah untuk membuatnya benar pada pertama kalinya. Dengan demikian, pertanyaan manajemen tidaklah untuk membangun sebuah sistem contoh dan untuk membuangnya. Anda akan melakukannya. Satu-satunya pertanyaan adalah apakah akan merencanakan lebih dulu untuk membangun sebuah pembuangan, atau menjanjikan untuk menyampaikan pembuangan tersebut kepada pelanggan...".

Prototipe bisa berfungsi sebagai "sistem yang pertama". Brooks setuju bila kita membuangnya. Akan tetapi, mungkin ini merupakan pandangan yang ideal. Memang benar bahwa baik pelanggan maupun pengembang menyukai paradigma prototipe. Para pemakai merasa enak dengan sistem aktual, sedangkan pengembang bisa membangunnya dengan segera. Akan tetapi, *prototiping* bisa juga menjadi masalah karena alasan-alasan sebagai berikut.

1. Pelanggan melihat apa yang tampak sebagai versi perangkat lunak yang bekerja tanpa melihat bahwa prototipe itu dijalin bersama-sama "dengan permen karet dan *baling wire*", tanpa melihat bahwa di dalam permintaan untuk membuatnya bekerja, kita belum menentukan kualitas perangkat lunak secara keseluruhan atau kemampuan pemeliharaan untuk jangka waktu yang panjang. Ketika diberi informasi bahwa produk harus dibangun lagi agar tingkat kualitas yang tinggi bisa dijaga, pelanggan akan meneriakkan kekurangan dan permintaan agar dipakai "beberapa campuran" untuk membuat prototipe menjadi sebuah produk yang bekerja yang lebih sering terjadi, sehingga manajemen

pengembangan perangkat lunak menjadi penuh dengan belas kasihan.

2. Pengembang sering membuat kompromi-kompromi implementasi untuk membuat prototipe bekerja dengan cepat. Sistem operasi atau bahasa pemrograman yang tidak sesuai bisa dipakai secara sederhana karena mungkin diperoleh dan dikenal, algoritma yang tidak efisien secara sederhana bisa diimplementasikan untuk mendemonstrasi-kan kemampuan. Setelah selang waktu tertentu, pengembang mungkin mengenali pilihan-pilihan tersebut dan melupakan semua alasan mengapa mereka tidak cocok. Pilihan yang kurang ideal telah menjadi bagian integral dari sebuah sistem.

Meskipun berbagai masalah bisa terjadi, prototipe bisa menjadi paradigma yang efektif bagi rekayasa perangkat lunak. Kuncinya adalah mendefinisikan aturan-aturan main pada saat awal, yaitu pelanggan dan pengembang keduanya harus setuju bahwa prototipe dibangun untuk mekanisme pendefinisian kebutuhan. Prototipe kemudian disingkirkan (paling tidak sebagian), dan perangkat lunak aktual direkayasa dengan tertuju kepada kualitas dan kemampuan pemeliharaan.

3.3 Model RAD (*Rapid Application Development*)

RAD (*Rapid Aplication Development*) adalah sebuah model proses perkembangan perangkat lunak sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Model RAD ini merupakan sebuah adaptasi "kecepatan tinggi" dari model sekuensial linier di mana perkembangan cepat dicapai dengan menggunakan pendekatan konstruksi berbasis komponen. Jika kebutuhan dipahami dengan baik, proses RAD memungkinkan tim pengembangan menciptakan "sistem fungsional yang utuh" dalam periode waktu yang sangat pendek (kira-kira 60 sampai 90 hari). Karena dipakai pada aplikasi sistem konstruksi, pendekatan RAD melingkupi beberapa fase, yaitu: *bussiness modelling*, *data modelling*, *process modelling*, *aplication generation*, dan *testing and turnover*.

3.3.1 Pemodelan Bisnis (*Bussiness Modelling*)

Aliran informasi di antara fungsi-fungsi bisnis dimodelkan dengan suatu cara untuk menjawab pertanyaan-pertanyaan berikut: Informasi apa yang mengendalikan proses bisnis? Informasi apa yang dimunculkan? Siapa yang memunculkannya? Ke mana informasi itu pergi? Siapa yang memprosesnya?

3.3.2 Pemodelan Data (*Data Modelling*)

Aliran informasi yang didefinisikan sebagai bagian dari fase *bussiness modeling* disaring ke dalam serangkaian objek data yang dibutuhkan untuk menopang bisnis tersebut. Karakteristik (disebut atribut) masing-masing objek diidentifikasi dan hubungan antara objek-objek tersebut didefinisikan.

3.3.3 Pemodelan Proses (*Process Modelling*)

Aliran informasi yang didefinisikan di dalam fase data modeling ditansformasikan untuk mencapai aliran informasi yang perlu bagi implementasi sebuah fungsi bisnis. Gambaran pemrosesan diciptakan untuk menambah, memodifikasi, menghapus, atau mendapatkan kembali sebuah objek data.

3.3.4 Pembentukan Aplikasi (*Application Development*)

RAD mengasumsikan pemakaian teknik generasi keempat. Selain menciptakan perangkat lunak dengan menggunakan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memakai lagi komponen program yang ada (pada saat memungkinkan) atau menciptakan komponen yang bisa dipakai lagi (bila perlu). Pada semua kasus, alat-alat bantu otomatis dipakai untuk memfasilitasi konstruksi perangkat lunak.

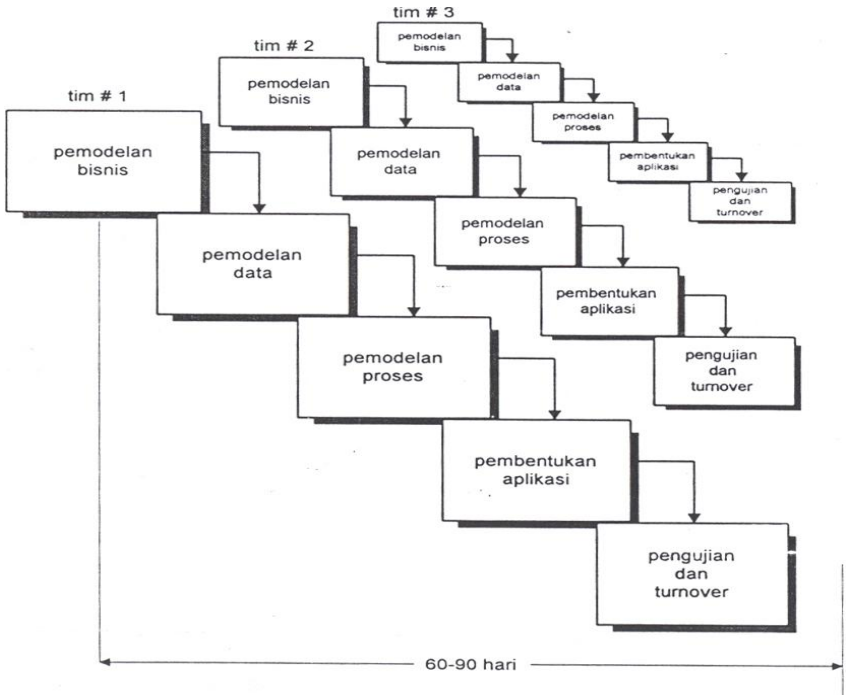
3.3.5 Pengujian (*Testing*) dan *Turnover*

Karena proses RAD menekankan pada pemakaian kembali, banyak komponen program telah diuji. Hal ini mengurangi keseluruhan waktu pengujian. Akan tetapi, komponen baru harus diuji dan semua *interface* harus dilatih secara penuh.

Model proses RAD diilustrasikan pada Gambar 3.6. Secara jelas batasan waktu yang dibebankan pada sebuah proyek RAD memerlukan "ruang lingkup yang bisa diskala". Jika sebuah aplikasi bisnis dapat dimodulkan dengan cara tertentu sehingga memungkinkan setiap fungsi mayor untuk dilengkapi dalam waktu kurang dari 3 bulan (dengan menggunakan pendekatan yang digambarkan di atas), maka aplikasi itu merupakan kandidat bagi RAD. Masing-masing fungsi mayor bisa dibicarakan oleh suatu tim RAD yang terpisah, dan kemudian diintegrasikan untuk membentuk suatu kesatuan.

Seperti semua proses model yang lain, pendekatan RAD memiliki kekurangan:

- Bagi proyek yang besar tetapi berskala, RAD memerlukan sumber daya manusia yang memadai untuk menciptakan jumlah tim RAD yang baik.
- RAD menuntut pengembang dan pelanggan memiliki komitmen di dalam aktivitas *rapid-fire* yang diperlukan untuk melengkapi sebuah sistem, di dalam kerangka waktu yang sangat diperpendek. Jika komitmen tersebut tidak ada dari tiap konstituen, proyek RAD akan gagal.



Gambar 3.6 Model RAD

Bila sistem tidak dapat dimodulkan dengan teratur, pembangunan komponen penting pada RAD akan menjadi sangat problematis. Tidak semua aplikasi sesuai untuk RAD. RAD menjadi tidak sesuai jika risiko teknisnya tinggi. Hal ini terjadi bila sebuah aplikasi baru memforsir teknologi baru atau bila perangkat lunak baru membutuhkan tingkat *interoperabilitas* yang tinggi dengan program komputer yang ada.

RAD menekankan perkembangan komponen program yang bisa dipakai kembali. *Reusabilitas* menjadi batu pertama teknologi objek, dan ditemui di dalam model proses rakitan komponen.

3.4 Model Evolusioner

Ada pengakuan yang sedang tumbuh bahwa perangkat lunak, seperti semua sistem kompleks yang lain, mencukupi lebih dari satu periode waktu. Kebutuhan bisnis dan produk kadang-kadang berubah seiring dengan laju perkembangannya, membentuk sebuah jejak garis lurus menuju hasil akhir yang tidak realistis, batas waktu pasar yang ketat menyebabkan pelengkapan produk perangkat lunak yang komprehensif menjadi tidak mungkin, tetapi sebuah versi yang terbatas harus dikenalkan untuk memenuhi tekanan yang kompetitif, sejumlah produk inti atau ekstensi sistem harus dibatasi. Di dalam situasi ini maupun yang lainnya, perekayasa perangkat lunak membutuhkan sebuah model proses yang sudah dirancang secara eksplisit untuk mengakomodasi produk perkembangan sepanjang waktu.

Model sekuensial linier dirancang bagi perkembangan garis lurus. Pada dasarnya pendekatan air terjun ini mengandalkan bahwa sebuah sistem lengkap akan disampaikan setelah urutan linier tersebut dilengkapi. Model prototipe dirancang untuk mendorong konsumen (atau pengembang) agar memahami kebutuhan. Secara umum model itu tidak dirancang untuk menyampaikan sistem produksi. Sifat evolusioner perangkat lunak tidak dimasukkan di dalam salah satu dari paradigma rekayasa perangkat lunak klasik tersebut.

Model evolusioner adalah model iteratif. Model evolusioner ditandai dengan tingkah laku yang memungkinkan perekayasa perangkat lunak mengembangkan versi perangkat lunak yang lebih lengkap sedikit demi sedikit.

3.5 Model Pertambahan

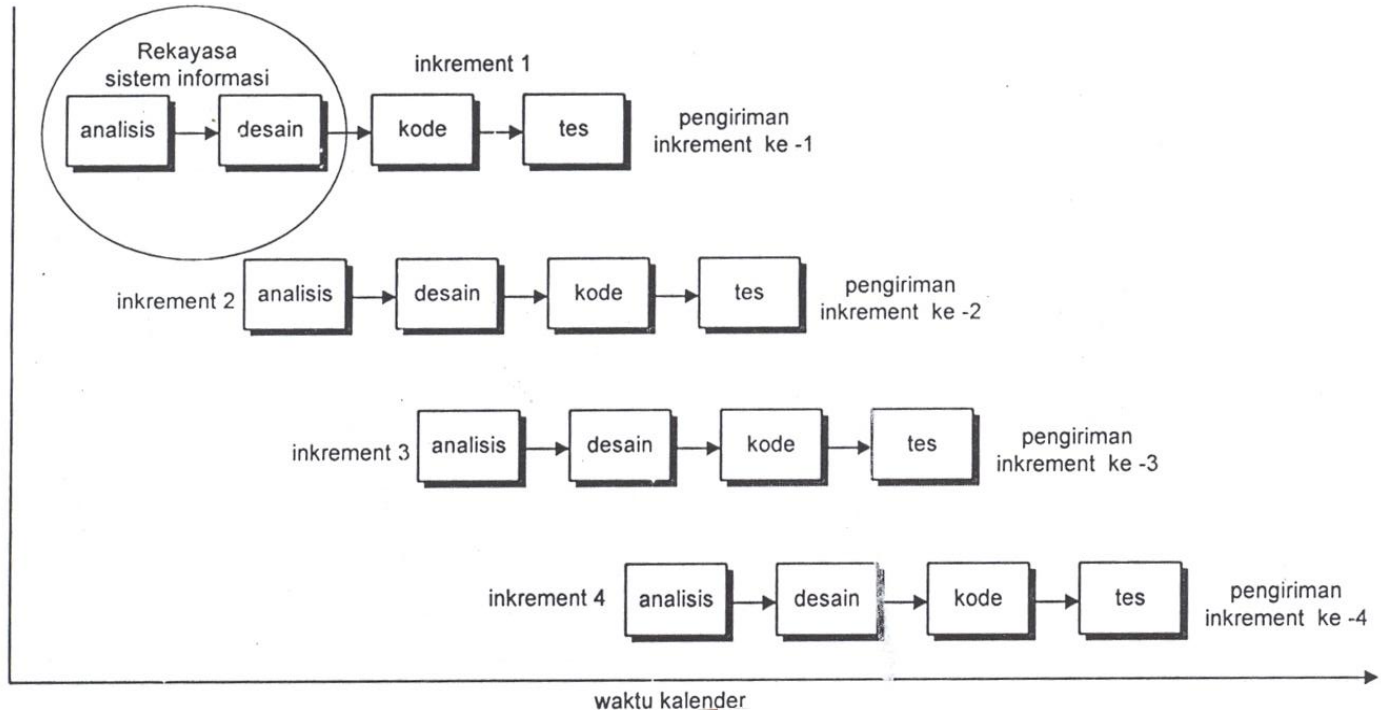
Model inkeremental menggabungkan elemen-elemen model sekuensial linier (diaplikasikan secara berulang) dengan filosofi prototipe iteratif. Seperti diperlihatkan pada Gambar 3.7, model pertambahan memakai urutan-urutan linier di dalam model yang membingungkan, seiring dengan laju waktu kalender. Setiap urutan linier menghasilkan pertambahan, perangkat lunak "yang bisa disampaikan". Contohnya, perangkat lunak pengolah kata yang dikembangkan dengan menggunakan paradigma pertambahan akan menyampaikan manajemen file dasar, *editing*, serta fungsi penghasilan dokumen pada pertambahan pertama, kemudian *editing* dan kemampuan penghasilan dokumen yang lebih canggih pada

pertambahan kedua, pengecekan *spelling* dan tata bahasa pada pertambahan ketiga, serta kemampuan pengaturan halaman tingkat lanjut pada tahap pertambahan keempat. Harus dicatat bahwa aliran proses untuk berbagai pertambahan tersebut dapat menggabungkan paradigma prototipe.

Pada saat model pertambahan dipergunakan, pertambahan pertama sering merupakan produk inti (*core product*), yaitu sebuah model pertambahan yang dipergunakan, tetapi beberapa muka tambahan (beberapa diketahui dan beberapa tidak) tetap tidak disampaikan. Produk inti tersebut dipergunakan oleh pelanggan (atau mengalami pengkajian detail). Sebagai hasil dari pemakaian dan/atau evaluasi, maka dikembangkan rencana bagi pertambahan selanjutnya. Rencana tersebut menekankan modifikasi produk inti untuk secara lebih baik memenuhi kebutuhan para pelanggan dan penyampaian fitur serta fungsionalitas tambahan. Proses ini diulangi mengikuti penyampaian setiap pertambahan sampai bisa menghasilkan produk yang lengkap.

Model proses pertambahan tersebut, seperti model prototipe dan pendekatan-pendekatan evolusioner yang lain, bersifat iteratif. Akan tetapi, tidak seperti model prototipe, model pertambahan berfokus pada penyampaian produk operasional dalam setiap pertambahannya. Pertambahan awal ada di versi *stripped down* dari produk akhir, tetapi memberikan kemampuan untuk melayani pemakai dan juga menyediakan *platform* untuk evaluasi oleh pemakai.

Perkembangan pertambahan, khususnya berguna pada saat *staffing*, tidak bisa dilakukan dengan menggunakan implementasi lengkap oleh batas waktu bisnis yang sudah disepakati untuk proyek tersebut. Jika produk inti diterima dengan baik, maka staf tambahan (bila dibutuhkan) bisa ditambahkan untuk mengimplementasi pertambahan selanjutnya. Sebagai tambahan, sistem mayor yang sedang pada masa perkembangan serta waktu penyampaiannya belum pasti, mungkin membutuhkan keberadaan perangkat keras yang baru. Bisa juga rencana tertentu dibuat untuk menghindari pemakaian perangkat lunak ini, sehingga memungkinkan fungsionalitas partial disampaikan kepada pemakai tanpa harus banyak tertunda.



Gambar 3.7 Model Inkremental

3.6 Model Spiral

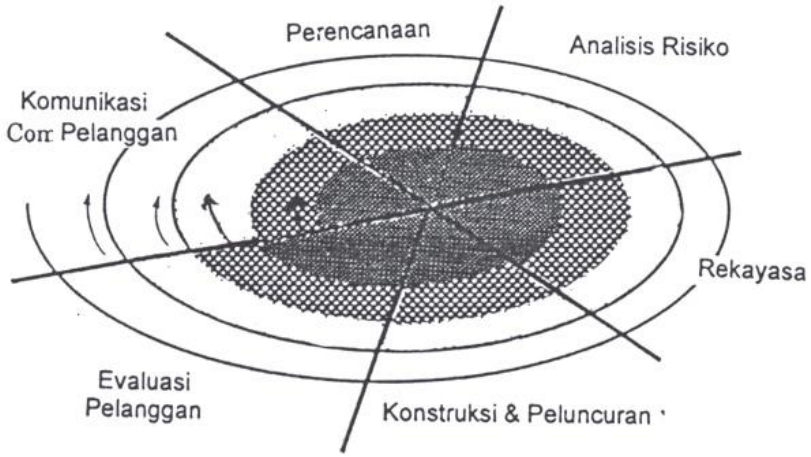
Model spiral (*spiral model*) yang pada awalnya diusulkan oleh Boehm, adalah model proses perangkat lunak yang evolusioner yang merangkai sifat iteratif dari prototipe dengan cara kontrol dan aspek sistematis dari model sekuensial linier. Model itu berpotensi untuk pengembangan versi pertambahan perangkat lunak secara cepat. Di dalam model spiral, perangkat lunak dikembangkan di dalam suatu deretan pertambahan. Selama awal iterasi, rilis inkremental bisa merupakan sebuah model atau prototipe kertas. Selama iterasi berikutnya, sedikit demi sedikit dihasilkan versi sistem rekayasa yang lebih lengkap.

Model spiral dibagi menjadi sejumlah aktifitas *kerangka kerja*, disebut juga wilayah tugas, di antara tiga sampai enam wilayah tugas. Gambar 3.8 menggambarkan model spiral yang berisi enam wilayah tugas:

- Komunikasi pelanggan. Tugas-tugas yang dibutuhkan untuk membangun komunikasi yang efektif di antara pengembang dan pelanggan.
- Perencanaan. Tugas-tugas yang dibutuhkan untuk mendefinisikan sumber-sumber daya, ketepatan waktu, dan proyek informasi lain yang berhubungan.
- Analisis risiko. Tugas-tugas yang dibutuhkan untuk menaksir risiko-risiko, baik manajemen maupun teknis.
- Perekayasaan. Tugas-tugas yang dibutuhkan untuk membangun satu atau lebih representasi dari aplikasi tersebut.
- Konstruksi dan peluncuran. Tugas-tugas yang dibutuhkan untuk mengkonstruksi, menguji, memasang (*install*) dan memberikan pelayanan kepada pemakai (contohnya pelatihan dan dokumentasi).
- Evaluasi pelanggan. Tugas-tugas yang dibutuhkan untuk memperoleh umpan balik dari pelanggan dengan didasarkan pada evaluasi representasi perangkat lunak, yang dibuat selama masa perekayasaan, dan diimplementasikan selama masa pemasangan.

Masing-masing wilayah bersih sederetan tugas kerja yang disesuaikan dengan ciri-ciri dari proyek yang dilakukan. Untuk proyek yang kecil, jumlah tugas kerja dan formalitasnya rendah. Untuk yang lebih besar, proyek-proyek yang lebih kritis, setiap daerah tugas berisi lebih banyak lagi tugas kerja yang dikhususkan untuk mencapai tingkat formalitasnya yang lebih tinggi. Di dalam

semua kasus, diaplikasikan aktivitas pelindung (seperti pengaturan konfigurasi perangkat lunak dan kepastian kualitas perangkat lunak).



Gambar 3.8 Model spiral tipikal

Ketika proses revolusioner ini mulai, tim rekayasa perangkat lunak bergerak searah jarum mengelilingi spiral tersebut dengan dimulai intinya. Lintasan pertama putaran spiral menghasilkan perkembangan dari spesifikasi produk, putaran spiral selanjutnya mungkin dipakai untuk mengembangkan sebuah prototipe, dan secara progresif mengembangkan versi perangkat lunak yang lebih canggih. Masing-masing lintasan yang melalui daerah perencanaan menghasilkan penyesuaian pada rencana proyek. Biaya dan jadwal disesuaikan berdasarkan umpan balik yang disimpulkan dari evaluasi pelanggan. Sebagai tambahan, manajer proyek menyesuaikan jumlah iterasi yang direncanakan yang dibutuhkan untuk melengkapi perangkat lunak.

Sebagaimana model proses klasik yang berakhir pada saat perangkat lunak sudah disampaikan, model spiral bisa disesuaikan agar perangkat lunak bisa dipakai selama hidup perangkat lunak komputer. Sebuah sumbu titik masuk proyek (*project entry point*) bisa didefinisikan seperti pada Gambar 3.9. setiap kubus yang diletakkan sepanjang sumbu mewakili titik awal untuk sebuah tipe proyek yang berbeda. Sebuah proyek perkembangan konsep (*concept development project*) berawal dari inti spiral dan akan terus berlangsung (iterasi ganda berlangsung sepanjang lintasan spiral yang mengikat daerah pusat yang diarsir) sampai perkembanoan konsep itu terlengkapi. Jika konsep tersebut akan dikembangkan ke dalam sebuah produk aktual, proses tersebut akan berlangsung

melalui kubus selanjutnya (titik masuk proyek perkembangan konsep yang baru) dan berarti sebuah proyek pengembangan yang baru telah dimulai. Produk yang baru tersebut akan melewati sejumlah iterasi yang mengelilingi spiral, mengikuti lintasan yang mengikat daerah yang mempunyai arsiran yang lebih terang dibandingkan daerah inti. Aliran proses yang sama juga berlangsung untuk tipe-tipe proyek yang lain.

Gambar 3.9 Model spiral yang disesuaikan untuk siklus hidup bagian dalam

Pada dasar spiral, ketika ditandai dengan cara tertentu, tetap bersifat operatif sampai perangkat lunak itu pensiun. Ada saat-saat di mana proses mengendor, tetapi kapan saja sebuah perubahan dilakukan, proses akan dimulai lagi pada *entri point* yang sesuai (contohnya pengembangan produk).

Model spiral menjadi sebuah pendekatan yang realistis bagi perkembangan sistem dan perangkat lunak skala besar. Karena perangkat lunak terus bekerja selama proses bergerak, pengembang dan pemakai memahami dan bereaksi lebih baik terhadap risiko dari setiap tingkat evolusi. Model spiral menggunakan prototipe sebagai mekanisme pengurangan risiko. Akan tetapi, yang lebih penting lagi,

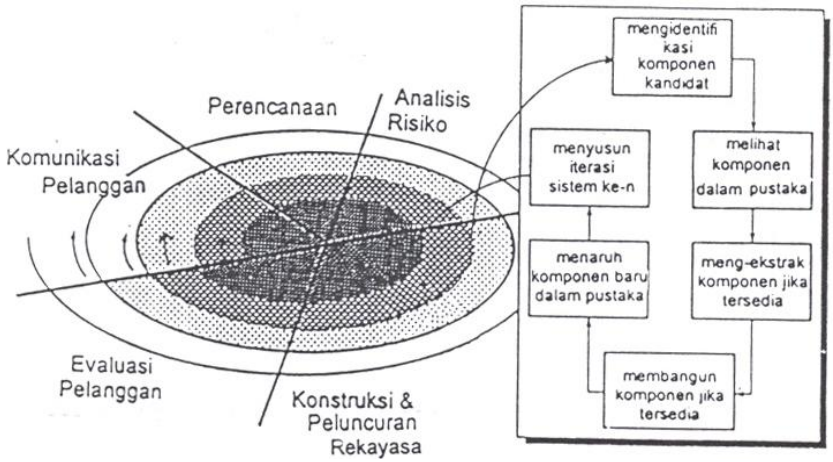
model spiral memungkinkan pengembang menggunakan pendekatan prototipe pada setiap keadaan di dalam evolusi produk. Model spiral menjaga pendekatan langkah demi langkah secara sistematik seperti yang diusulkan oleh siklus kehidupan klasik, tetapi memasukkannya ke dalam kerangka kerja iteratif yang secara realistis merefleksikan dunia nyata. Model spiral membutuhkan pertimbangan langsung terhadap risiko teknis pada semua keadaan proyek, yang jika dipakai secara benar akan mengurangi risiko sebelum menjadi sangat problematik.

Berdasarkan paradigma yang lain, model spiral bukanlah sebuah obat mujarab. Mungkin akan sangat sulit untuk meyakinkan konsumen (khususnya dalam situasi kontrak) bahwa pendekatan evolusioner bisa dikontrol. model spiral memerlukan keahlian penaksiran risiko yang masuk akal, dan sangat bertumpu pada keahlian ini untuk mencapai keberhasilan. Jika sebuah risiko mayor tidak ditemukan dan diatur, pasti akan terjadi masalah. Akhirnya, model itu sendiri masih baru dan belum dipergunakan secara luas seperti paradigma sekuensial linier dan prototipe. Model ini membutuhkan waktu bertahun-tahun sampai kehandalan paradigma baru yang penting ini bisa dipertimbangkan dengan kepastian absolut.

3.7 Model Rakitan Komponen

Teknologi objek memberikan kerangka kerja teknis untuk sebuah model proses berbasis komponen bagi rekayasa perangkat lunak. Paradigma orientasi objek menekankan kreasi kelas yang mengenkapsulasi data dan algoritma yang dipakai untuk memanipulasi data. Bila data dirancang dan diimplementasi dengan baik, kelas orientasi objek menjadi bisa dipakai lagi pada aplikasi serta arsitektur sistem berbasis komputer yang berbeda-beda.

Model rakitan komponen (Gambar 3.10) menggabungkan beberapa karakteristik model spiral. Model ini bersifat evolusioner, sehingga membutuhkan pendekatan iteratif untuk menciptakan perangkat lunak. Akan tetapi, model rakitan komponen merangkai aplikasi dari komponen perangkat lunak sebelum dipaketkan (kadang-kadang disebut “kelas”).



Gambar 3.10 Model rakitan komponen

Aktivitas perangkat lunak dimulai dengan identifikasi calon kelas. Hal ini dipenuhi dengan mengamati data yang akan dimanipulasi oleh aplikasi dan algoritma-algoritma yang akan diaplikasikan untuk melakukan manipulasi. Data dan algoritma yang berhubungan dikemas ke dalam sebuah kelas.

Kelas-kelas (disebut komponen di dalam Gambar 3.10) yang diciptakan di dalam proyek perangkat lunak disimpan di dalam pustaka kelas (*class library*) atau tempat penyimpanan. Sekali kelas-kelas kandidat diidentiftkasi, pustaka-pustaka kelas diamati untuk menentukan apakah kelas-kelas itu ada. Bila ada, kelas itu kemudian diekstraksi dari pustaka dan digunakan lagi. Jika sebuah kelas kandidat tidak ada di dalam pustaka, dia direkayasa dengan menggunakan metode yang berorientasi objek. Iterasi aplikasi pertama yang akan dibangun kemudian dikomposisi dengan mempergunakan kelas yang diekstraksi dari pustaka dan kelas-kelas baru lain yang dibangun untuk memenuhi kebutuhan-kebutuhan unik dari aplikasi. Kemudian aliran proses kembali ke spiral dan akan masuk lagi ke iterasi rakitan komponen selama subsekuen melewati aktivitas perekayasaan.

Model rakitan komponen membawa kepada penggunaan kembali perangkat lunak, dan kegunaan kembali tersebut memberi sejumlah keuntungan yang bisa diukur pada perekayasa perangkat lunak. Berdasarkan studi reusabilitas (keadaan bisa dipakai lagi) ini, QSM Associates, Inc. melaporkan bahwa rakitan komponen telah memberikan reduksi sebanyak 70% di dalam siklus waktu

perkembangan, 84% pada biaya proyek, serta memberikan indeks produktivitas sebesar 26,2 bila dibandingkan dengan nilai normal yang besarnya hanya 16,9. Meskipun hasil ini merupakan fungsi utama pustaka komponen, tetap saja ada sedikit pertanyaan apakah model rakitan komponen tersebut memberikan keuntungan yang signifikan bagi rekayasa perangkat lunak.

3.8 Model Perkembangan Konkuren

Model perkembangan konkuren atau disebut juga rekayasa konkuren digambarkan oleh Davis dan Sitaram dengan cara sebagai berikut:

> Para manajer proyek yang menelusuri status proyek di dalam bentuk fase-fase mayor (dari siklus kehidupan klasik) tidak memiliki ide tentang status proyek mereka. Hal ini merupakan contoh dari cara menelusuri serangkaian aktivitas yang sangat kompleks dengan menggunakan model yang sangat sederhana. Perlu dicatat bahwa meskipun sebuah proyek (yang besar) ada di dalam fase pengkodean, ada individu di dalam proyek yang terlibat di dalam aktivitas yang dihubungkan secara khusus dengan banyak fase perkembangan secara simultan. Contohnya, orang yang menulis kebutuhan, perancangan, pengkodean, pengujian, dan pengujian integrasi (semuanya dilakukan pada saat yang sama). Model proses rekayasa perangkat lunak oleh Humphrey dan Kellner memperlihatkan konkurensi yang ada untuk aktivitas yang terjadi pada satu fase. Usaha Kellnel yang paling akhir menggunakan *statechart* (sebuah notasi yang mewakili keadaan sebuah proses) untuk mencerminkan hubungan konkuren yang ada di antara aktivitas-aktivitas yang dihubungkan dengan sebuah kejadian khusus (seperti perubahan kebutuhan selama perkembangan terakhir), tetapi gagal menangkap manfaat konkurensi pada semua aktivitas perkembangan dan manajemen perangkat lunak di dalam sebuah proyek. Sebagian besar model proses perkembangan perangkat lunak dikendalikan oleh waktu, semakin dia terlambat, maka proses perkembangannya juga semakin terlambat. Model proses konkuren dikendalikan oleh kebutuhan para pemakai, keputusan manajemen, dan hasil pengkajian.

Model proses yang konkuren dapat disajikan secara skematis sebagai sederetan aktivitas teknis mayor, tugas-tugas dan keadaannya yang lain. Contohnya aktivitas rekayasa yang dibatasi untuk model spiral dipenuhi dengan melakukan tugas-tugas sebagai berikut: *prototiping*, dan atau pemodelan analisis, spesifikasi kebutuhan, dan rancangan.

Gambar 3.11 memberikan representasi skematis dari aktivitas di dalam model proses yang konkuren. Aktivitas – analisis – bisa berada dalam salah satu dari keadaan-keadaan yang dicatat pada saat tertentu. Dengan cara yang sama, aktivitas yang lain (desain atau komunikasi pelanggan) dapat direpresentasikan dalam sebuah sikap yang analog. Semua aktivitas ada secara konkuren tetapi dia tinggal di dalam keadaan yang berbeda. Contohnya, di awal proyek aktivitas *komunikasi pelanggan* (tidak diperlihatkan di dalam gambar) telah melengkapi iterasi pertamanya dan berada di dalam keadaan menunggu perubahan. Aktiviras *analisis* (yang ada di dalam *none state* sementara komunikasi pelanggan inisial dilengkapi) sekarang membuat sebuah transisi ke dalam keadaan di bawah perkembangan. Akan tetapi, jika pelanggan menunjukkan bahwa perubahan kebutuhan harus dibuat, maka aktivitas *analisis* bergerak dari keadaan *di dalam perkembangan* ke dalam keadaan *menunggu perubahan*.

Model proses konkuren sering digunakan sebagai paradigma bagi pengembangan aplikasi klien/server. Sistem klien/server dirancang dari serangkaian komponen fungsional. Bila diaplikasikan kepada klien/server, model proses konkuren akan mendefinisikan aktivitas di dalam dua dimensi, yaitu: *dimensi sistem* dan *dimensi komponen*. Isu tingkat sistem dituju dengan menggunakan tiga aktivitas, yaitu: *desain*, *assembly*, dan *pemakaian*. Dimensi komponen dituju dengan dua aktivitas, yaitu: *desain* dan *realisasi*. Konkurensi dicapai dengan dua cara, yaitu: (1) aktivitas sistem dan komponen yang berlangsung secara simultan dan dapat dimodelkan dengan menggunakan pendekatan orientasi keadaan yang digambarkan di atas; (2) aplikasi klien/server khusus diimplementasikan dengan banyak komponen di mana masing-masing bisa dirancang dan direalisasikan secara konkuren.

Gambar 3.11 Elemen model proses konkuren

Kenyataannya model proses kongkuran bisa diaplikasikan ke dalam semua tipe perkembangan perangkat lunak, dan memberikan gambaran akurat mengenai keadaan tertentu dari sebuah proyek. Selain membatasi aktivitas rekayasa perangkat lunak ke dalam sederetan kejadian, model proses juga mendefinisikan jaringan aktivitas. Setiap aktivitas pada jaringan berada secara simultan dengan aktivitas-aktivitas yang lain. Kejadian-kejadian yang dimunculkan di dalam aktivitas yang diberikan atau pada tempat-

tempat lain di dalam jaringan aktivitas akan memacu transisi di antara keadaan-keadaan sebuah aktivitas.

3.9 Model Formal

Model metode formal mencakup sekumpulan aktivitas yang membawa kepada spesifikasi matematis perangkat lunak komputer. Metode formal memungkinkan perekayasa perangkat lunak untuk mengkhususkan, mengembangkan, dan memverifikasi sistem berbasis komputer dengan menggunakan notasi matematis yang tepat. Variasi di dalam pendekatan ini, disebut juga *clean-room rekayasa perangkat lunak*, sedang diaplikasikan oleh banyak organisasi pengembang perangkat lunak.

Bila metode formal dipakai selama masa pengembangan, metode itu memberikan mekanisme untuk mengeliminasi banyak masalah yang sulit dipecahkan dengan menggunakan paradigma perangkat lunak yang lain. Ambiguitas, ketidaklengkapan, dan ketidak-konsistenan bisa ditemukan dan diperbaiki secara lebih mudah – tidak melalui kajian *ad hoc* tetapi melalui aplikasi analisis matematis. Jika metode formal dipakai selama masa perancangan, mereka berfungsi sebagai dasar bagi verifikasi program sehingga memungkinkan perekayasa perangkat lunak untuk menemukan dan memperbaiki kesalahan yang mungkin saja tidak terdeteksi.

Meskipun belum menjadi pendekatan utama, model metode formal sudah menawarkan janji perangkat lunak yang bebas cacat/kesalahan. Akan tetapi, perhatian tentang kemampuan aplikasinya di dalam lingkungan bisnis sudah mulai disuarakan seperti berikut ini.

1. Pengembangan model formal banyak memakan waktu dan mahal.
2. Karena beberapa pengembang perangkat lunak perlu mempunyai latar belakang yang diperlukan untuk mengaplikasikan metode formal, maka diperlukan pelatihan yang ekstensif.
3. Sulit untuk menggunakan model-model sebagai sebuah mekanisme komunikasi bagi pemakai yang secara teknik belum canggih.

Meskipun demikian, sepertinya metode formal ini akan memperoleh banyak penganut di antara pengembang perangkat lunak yang harus membangun perangkat lunak yang kritis untuk keselamatan (misalnya pengembang perangkat medis dan

penerbangan pesawat), serta di antara pengembang yang harus menderita karena faktor ekonomis yang harus dialami oleh perangkat lunak.

Bentuk "teknik generasi keempat" (4GT) mencakup serangkaian bantu perangkat lunak yang luas yang secara umum memiliki satu hal. Masing-masing memungkinkan perekayasa perangkat lunak untuk mengkhususkan beberapa karakteristik perangkat lunak pada suatu tingkat yang tinggi. Alat-alat bantu tersebut kemudian secara otomatis memunculkan kode sumber yang berdasarkan pada spesifikasi perekayasa. Ada sedikit perdebatan bila tingkatnya lebih tinggi, di mana perangkat lunak bisa ditetapkan untuk sebuah mesin, sehingga suatu program bisa dibangun dengan lebih cepat. Paradigma 4GT untuk rekayasa perangkat lunak berfokus kemampuan spesiflkasi perangkat lunak dengan menggunakan bentuk bahasa yang dikhususkan atau sebuah notasi grafik yang menggambarkan masalah yang akan dipecahkan ke dalam bentuk yang dapat dipahami oleh pelanggan.

Sekarang ini lingkungan perkembangan perangkat lunak yang mendukung paradigma 4GT menyangkut beberapa atau semua alat bantu yang meliputi: bahasa nonprosedural untuk *query* basis data, generasi laporan, manipulasi data, interaksi layar dan definisi, dan penghasilan kode, kemampuan grafls tingkat tinggi, dan kemampuan *spreadsheet*. Pada dasarnya dulu alat-alat bantu yang ditulis di atas bisa diperoleh hanya untuk domain aplikasi tertentu saja, tetapi sekarang lingkungan 4GT telah diperluas untuk menjangkau sebagian besar kategori aplikasi perangkat lunak.

Seperti paradigma-paradigma yang lain, 4GT dimulai dengan langkah pengumpulan kebutuhan. Secara ideal, pemakai akan menggambarkan kebutuhan dan ini akan diterjemahkan secara langsung ke dalam sebuah prototipe operasional. Akan tetapi, hal ini tidak berhasil. Pemakai mungkin tidak yakin tentang apa yang dibutuhkan, mungkin ragu untuk mengkhususkan kenyataan yang diketahui, dan mungkin tidak mampu atau tidak mau menentukan informasi dengan cara tertentu yang bisa dikonsumsi oleh alat-alat bantu 4GT. Untuk versi ini dialog pelanggan dan pengembang yang digambarkan untuk model proses yang lain tetap menjadi bagian dasar pendekatan 4GT.

Untuk aplikasi yang kecil, mungkin untuk bergerak secara langsung dari langkah pengumpulan kebutuhan ke implementasi, digunakan bahasa generasi keempat nonprosedural (4GL -- *fourth*

generation language). Untuk usaha yang lebih besar, diperlukan pengembangan strategi desain bagi sistem tersebut, sekalipun sebuah 4GL akan digunakan. Pemakaian 4GT tanpa desain (untuk proyek besar) akan menyebabkan terjadinya kesulitan-kesulitan yang sama (kualitas yang buruk, kemampuan pemeliharaan yang buruk, penerimaan pelanggan yang buruk) yang telah kita alami pada saat mengembangkan perangkat lunak dengan menggunakan pendekatan *ad hoc*.

Implementasi dengan menggunakan 4GL memungkinkan pengembang perangkat lunak merepresentasikan *output* yang diinginkan dengan cara tertentu yang menghasilkan generasi kode yang otomatis untuk menghasilkan *output*. Jelas dengan demikian suatu struktur data dengan informasi yang relevan harus ada dan siap untuk diakses oleh 4GL.

Untuk mentransformasi implementasi 4GT ke dalam sebuah produk, pengembang harus melakukan pengujian secara teliti, mengembangkan dokumentasi yang bermakna, dan melakukan semua kegiatan integrasi pemecahan yang lain yang diperlukan di dalam paradigma rekayasa perangkat lunak yang lain. Sebagai tambahan, perangkat lunak 4GL yang dikembangkan harus dibangun dengan cara tertentu yang memungkinkan pemeliharaan dilakukan secara cepat.

Seperti semua paradigma rekayasa perangkat lunak yang lain, model 4GT pun memiliki keuntungan dan kerugian. Orang yang sepaham mengklaim masalah penghematan waktu yang dramatis di dalam pengembangan perangkat lunak, serta produktivitas yang dikembangkan dengan sangat baik dari orang-orang yang membangun perangkat lunak. Kelompok yang berlawanan mempermasalahkan alat-alat bantu 4GT yang tidak bisa dengan lebih mudah diterapkan dibanding bahasa pemrograman yang lain, sehingga kode sumber hasil yang dihasilkan oleh alat-alat bantu tersebut menjadi "tidak efisien". Selain itu, kemampuan pemeliharaan bagi sistem perangkat lunak yang dikembangkan dengan mempergunakan 4GT masih mengundang pertanyaan.

Ada beberapa segi baik dari kedua sisi klaim di atas, dan mungkin untuk menyimpulkan keadaan sekarang dari pendekatan 4GT, yaitu sebagai berikut.

1. Kegunaan 4GT telah disebarkan selama dekade terakhir dan sekarang merupakan pendekatan yang masih berjalan terus bagi berbagai area aplikasi yang berbeda-beda.

2. Data yang dikumpulkan dari perusahaan-perusahaan yang menggunakan 4GT menunjukkan bahwa waktu yang dibutuhkan untuk menghasilkan perangkat lunak sangat berkurang untuk aplikasi kecil dan menengah, serta jumlah desain dan analisis bagi aplikasi kecil juga berkurang.
3. Kegunaan 4GT untuk pengembangan perangkat lunak yang besar membutuhkan analisis, desain dan pengujian yang sangat banyak (aktivitas rekayasa perangkat lunak) untuk memperoleh penghematan waktu yang substansial yang dapat dicapai melalui eliminasi pengkodean.

Kesimpulannya, teknik generasi keempat telah menjadi sebuah bagian yang penting dari perkembangan perangkat lunak. Bila dirangkai dengan pendekatan rakitan komponen, paradigma 4GT bisa menjadi pendekatan yang dominan untuk pengembangan perangkat lunak.

3.10 Teknologi Proses

Model proses yang telah didiskusikan pada bagian terdahulu harus disesuaikan dahulu sebelum digunakan oleh tim proyek perangkat lunak. Untuk melakukannya telah dikembangkan alat-alat bantu teknologi proses untuk membantu organisasi yang sedang berlangsung, mengorganisasi tugas-tugas kerja, mengontrol dan memonitor kemajuan, serta mengatur kualitas teknis.

Alat-alat bantu teknologi proses memperbolehkan organisasi perangkat lunak untuk membangun sebuah model kerangka kerja proses umum otomatis, sejumlah tugas, dan aktivitas pelindung. Model tersebut, yang biasanya diwakilkan sebagai sebuah jaringan, kemudian dapat dianalisis untuk menentukan aliran kerja khusus dan mengamati struktur proses alternatif yang menyebabkan pengurangan waktu dan biaya pengembangan.

Sekali sebuah proses yang bisa diterima diciptakan, alat-alat bantu teknologi proses yang lain dapat dipergunakan untuk mengalokasi, memonitor, dan bahkan mengontrol semua tugas rekayasa perangkat lunak yang didefinisikan sebagai bagian dari model proses. Setiap anggota tim proyek perangkat lunak bisa mempergunakan alat-alat bantu tersebut untuk mengembangkan *checklist* dari tugas-tugas kerja yang akan dilakukan, hasil-hasil kerja yang akan diproduksi, dan aktivitas penjaminan kualitas yang akan dilakukan. Alat-alat bantu teknologi proses juga dapat dipergunakan

untuk mengkoordinasi penggunaan alat-alat bantu perangkat lunak komputer bntuan yang sesuai untuk tugas-tugas kerja khusus.

3.11 Produk dan Proses

Bila proses lemah maka tidak diragukan lagi hasil akhirnya akan buruk. Akan tetapi, ketergantungan yang *obsesive* pada proses juga berbahaya. Secara singkat, Margaret David mengomentari dualitas hasil dan proses sebagai berikut:

> *Sekitar setiap sepuluh tahun lebih atau kurang dari lima, komunitas perangkat lunak kembali mendefinisikan "masalah" dengan menggeser fokusnya dari isu produk ke isu proses. Demikianlah, kita telah mempergunakan bahasa program terstruktur (produk) diikuti dengan metode analisis terstruktur (proses) diikuti dengan enkapsulasi data (produk) diikuti dengan penekanan pada Rekayasa perangkat lunak Institute's Software Development Capability Maturity Model (proses).*

Sementara tendensi natural dari pendulum adalah kembali lagi ke titik tengah di antara dua titik ujung, fokus komunitas perangkat lunak bergeser secara konstan karena gaya baru diaplikasikan ketika ayunan yang terdahulu gagal. Ayunan-ayunan tersebut menjadi pengganggu di antara mereka sendiri karena mereka meragukan pelaksana perangkat lunak rata-rata dengan mengubah secara radikal apa artinya melakukan pekerjaan. Belum lagi untuk melakukannya dengan baik. Ayunan itu juga tidak memecahkan "masalah", karena gagal selama produk dan proses diperlakukan seperti membentuk sebuah dikotomi dan bukan dualitas.

Ada preseden di dalam komunitas ilmiah untuk mengajukan dugaan bahwa dualitas pada saat kontradiksi di dalam penelitian tidak dapat dijelaskan sepenuhnya oleh satu teori atau yang lainnya. Sifat mendua dari bahaya, yang secara simultan seperti gelombang dan partikel, telah diterima sejak tahun 1920 pada saat Louis de Broglie mengusulkannya. Saya percaya bahwa penelitian-penelitian terhadap perkakas perangkat lunak dan perkembangannya telah mendemonstrasikan dualitas yang mendasar antara produk dan proses. Anda tidak akan pernah bisa menarik atau memahami perkakas tersebut secara penuh, konteksnya, artinya, terlebih lagi jika Anda melihatnya hanya sebagai sebuah proses atau produk saja.

Semua kegiatan manusia bisa menjadi sebuah proses, tetapi masing-masing dari kita menarik diri dari kegiatan-kegiatan yang menghasilkan representasi atau contoh yang dapat dipergunakan atau dihargai oleh satu orang atau lebih tersebut, yaitu bahwa kita tidak merasa puas terhadap penggunaan kembali produk kita oleh kita sendiri atau orang lain.

Demikianlah, asimilasi cepat terhadap tujuan kegunaan kembali ke dalam perkembangan perangkat lunak, secara potensial menambah kepuasan pelaksana perangkat lunak yang ditarik dari kerja mereka. Asimilasi juga menambah urgensi penerimaan dualitas produk dan proses. Pemikiran tentang perkakas perangkat lunak hanya sebagai produk atau hanya sebagai proses akan mengaburkan konteks dan cara penggunaannya, atau mengaburkan kenyataan bahwa masing-masing kegunaan menghasilkan produk yang pada gilirannya akan dipergunakan sebagai input ke berbagai aktivitas pengembangan perangkat lunak yang lain. Dengan mengambil satu sudut pandang tertentu kesempatan untuk penggunaan kembali berkurang secara dramatis dan kesempatan untuk menambah kepuasan kerja akan hilang.

Orang menarik sebanyak mungkin (atau lebih) kepuasan dari proses kreatif seperti yang mereka lakukan pada produk akhir. Seorang artis menikmati goresan kuas sebanyak dia menikmati hasil yang sudah dibingkai. Seorang penulis menikmati pencarian gaya bahasa yang sesuai seperti menikmati buku yang sudah selesai. Seorang profesional perangkat lunak yang kreatif juga akan mendapat kepuasan dari proses sebanyak dia menggunakan produk/hasil akhir.

Kerja masyarakat perangkat lunak akan berubah di dalam tahun ini dan yang akan datang. Dualitas produk dan proses merupakan elemen yang penting di dalam menjaga manusia-manusia kreatif agar tetap terjalin, sementara rekayasa pemrograman dan perangkat lunak diselesaikan.

Bab 4

Analisis dan Desain Sistem Informasi Akademik Perguruan Tinggi

Pada bagian ini akan dijelaskan berbagai analisis dan desain aplikasi Sistem Informasi Akademik. Mengingat kompleksitas dan luasnya lingkup bidang akademik, pada bahasan ini dibatasi pada proses perkuliahan saja. Adapun teknik pengembangan aplikasi menggunakan model RAD (*Rapid Application Development*).

4.1 Analisis Sistem

Pada tahap ini akan dijabarkan analisis sistem yang meliputi kebutuhan sistem dan spesifikasi sistem yang baru.

4.1.1 Analisis Kebutuhan Sistem

Berikut adalah beberapa kebutuhan sistem yang perlu diakomodir sebelum melakukan pengembangan sistem.

- Aplikasi sistem informasi harus bisa diakses multi user dalam waktu yang bersamaan, mengingat kegiatan akademik yang melibatkan banyak entitas, seperti mahasiswa, staf, administrator, dan lain sebagainya.
- Sistem *database* yang diimplementasikan harus mampu menangani akses multi user dengan tingkat keamanan yang bisa diandalkan, cepat dalam pemrosesan, dan bisa diimplementasikan di berbagai *platform* sistem operasi, mengingat mungkinnya terjadi perbedaan penggunaan sistem operasi pada komputer server dan klien.
- Aplikasi sistem informasi harus bisa diakses melalui berbagai tempat yang terhubung jaringan internet.
- Aplikasi sistem informasi di masa yang akan datang mudah diintegrasikan dengan teknologi baru yang dirasa layak diterapkan, seperti akses via GPRS, SMS, atau yang lain.

- Biaya pengembangan yang relatif murah.

4.1.2 Spesifikasi Sistem yang Baru

Berdasarkan anlaisis kebutuhan yang dijabarkan di atas, berikut adalah spesifikasi sistem baru yang akan dibuat.

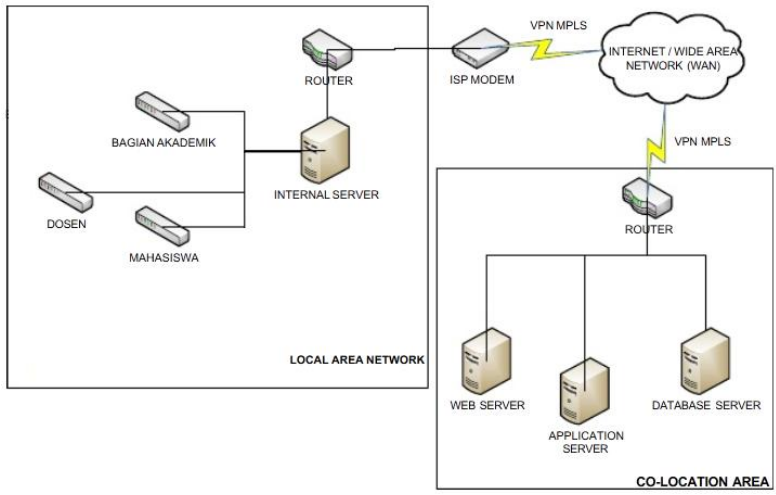
- Aplikasi sistem informasi dikembangkan dengan bahasa pemrograman web, dalam hal ini PHP dengan berbagai teknik dan *tool* yang bisa diintegrasikan, seperti penggunaan CMS (*content management system*) dan *framework*.
- Menggunakan MySQL atau PostgreSQL sebagai sistem *database* penyusun sistem informasi akademik.
- Penggunaan bahasa pemrograman berbasis web dan server yang on-line memungkin aplikasi bisa diakses dari berbagai penjuru dunia yang terhubung dengan jaringan internet.
- Disediakan manajemen server dan berbagai *tool* pendukung agar ke depan bisa diintegrasikan dengan teknologi baru.
- Digunakan sistem operasi berbasis Linux, bahasa pemrograman PHP, dan server *database* yang serba *open source* menjadikan aplikasi ini dikembangkan dengan biaya yang sangat murah.

4.2 Desain Sistem

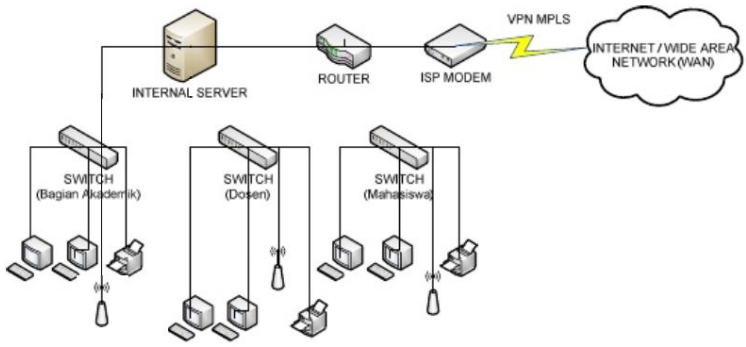
Desain sistem dimaksudkan untuk memudahkan pemahaman tentang struktur sistem informasi yang akan dibangun, mulai dari desain sistem komunikasi data, desain sistem dilihat dari sisi bisnis, desain data, dan desain proses, serta aliran informasi.

4.2.1 Struktur Sistem Komunikasi Data

Sistem Komunikasi Data yang dimaksud adalah arsitektur jaringan komputer di mana aplikasi sistem informasi akademik yang dikembangkan diimplementasikan. Gambar 4.1 berikut ini merupakan sistem komunikasi data aplikasi sistem informasi akademik yang akan dikembangkan sesuai dengan spesifikasi sistem baru hasil analisis.



(a)

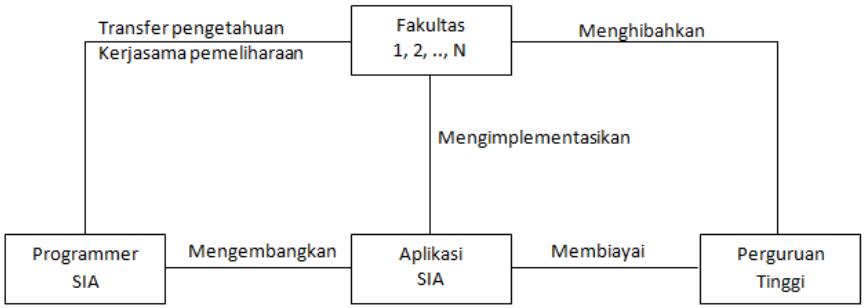


(b)

Gambar 4.1 (a) Arsitektur Jaringan komputer dan Sistem Komunikasi Data
(b) Arsitektur *Local Area Network* (*bawah*)

4.2.2 Pemodelan Bisnis

Pemodelan Bisnis digunakan untuk membuat menjelaskan hubungan antar entitas dunia nyata yang terlibat dalam aplikasi Sistem Informasi Akademik dilihat dari sisi bisnisnya.



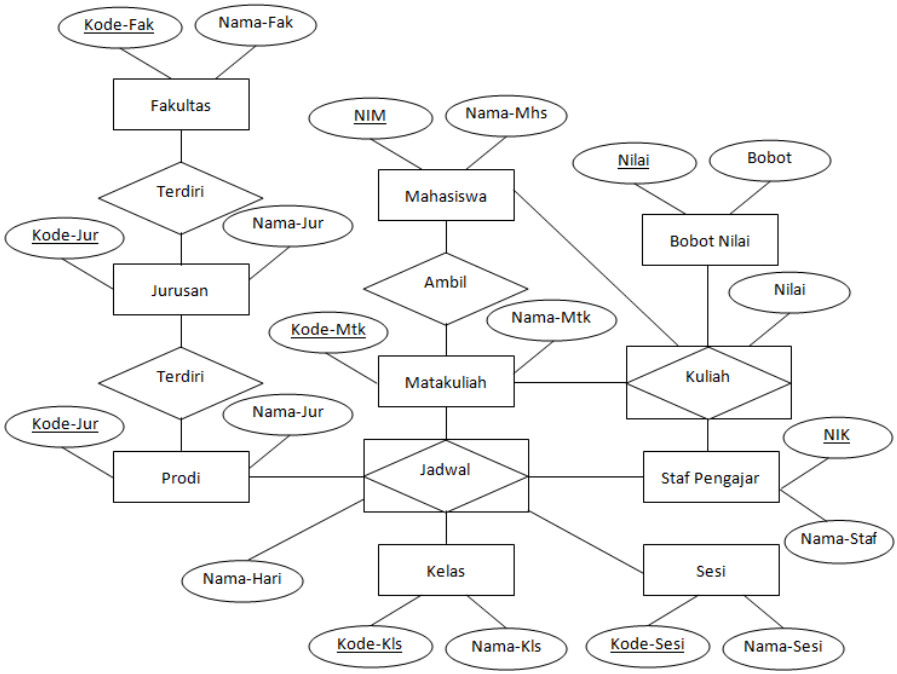
Gambar 4.2 Pemodelan Bisnis Sistem Informasi Akademik

Hal-hal yang dapat dijelaskan mengenai desain Sistem Informasi Akademik (SIA) pada level bisnis adalah sebagai berikut.

- Perguruan tinggi membiayai pengembangan Aplikasi SIA dengan memperkerjakan beberapa *programmer*.
- *Programmer* mengembangkan Aplikasi SIA.
- Aplikasi SIA oleh pihak Perguruan Tinggi dihibahkan ke masing-masing fakultas.
- Masing-masing fakultas mengimplementasikan Aplikasi SIA untuk kepentingan akademik.
- *Programmer* dan masing-masing fakultas melakukan kerjasama untuk Transfer Pengetahuan dan Kerjasama Pemeliharaan mengenai Aplikasi SIA.

4.2.3 Pemodelan Data

Pemodelan data digunakan untuk menjelaskan struktur *database* yang akan dibuat untuk aplikasi Sistem Informasi Akademik. Terdapat beberapa cara untuk merpresentasikan pemodelan data, diantaranya adalah Normalisasi dan Diagram ER (*Entity Relationship*). Pada penelitian ini tahap pemodelan data diselesaikan dengan Diagram ER.



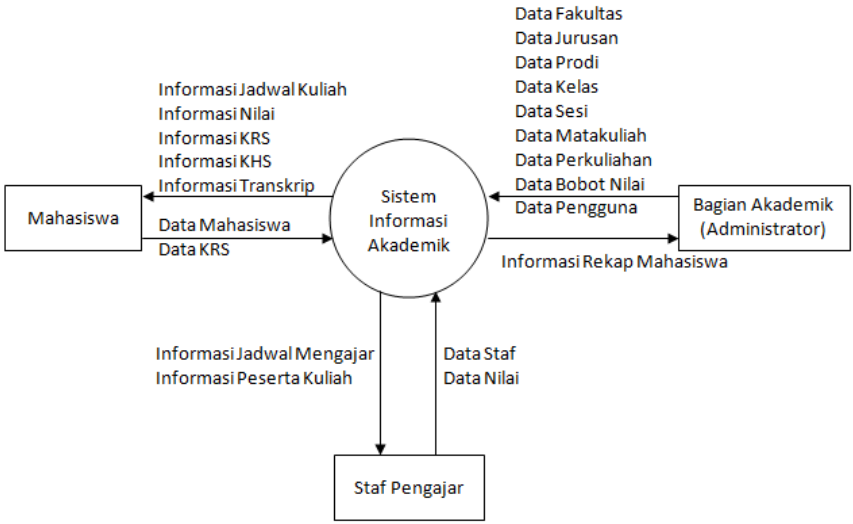
Gambar 4.3 Diagram ER Sistem Informasi Akademik

Keterangan:

- Atribut pada masing-masing entitas hanya diwakili beberapa atribut utama, sedangkan atribut penunjang tidak diikutkan dengan tujuan memperjelas hubungan antar entitas, mengingat banyaknya entitas yang terlibat.
- Atribut yang diberi tanda garis bawah (*underscorer*) merupakan atribut *primary key*.

4.2.4 Pemodelan Proses dan Aliran Informasi

Pemodelan Proses digunakan untuk menjelaskan proses-proses apa saja yang terlibat dalam sistem dan bagaimana aliran data yang diperlukan pada masing-masing proses. Pada penelitian ini digunakan DAD (Diagram Alir Data) atau DFD (*Data Flow Diagram*) untuk merepresentasikannya.



Gambar 4.4 Diagram Konteks Sistem Informasi Akademik

Hal-hal yang dapat dijelaskan mengenai desain Sistem Informasi Akademik (SIA) pada level proses adalah sebagai berikut.

1) Mahasiswa
   - Masukan ke sistem: Data Mahasiswa dan Data KRS (Kartu Rencana Studi).
   - Keluaran dari sistem: Informasi Jadwal Kuliah, Informasi Nilai, Informasi KRS, Informasi KHS (Kartu Hasil Studi), dan Informasi transkrip.
2) Staf
   - Masukan ke sistem: Data Staf dan Data Nilai.
   - Keluaran dari sistem: Informasi Jadwal Mengajar dan Informasi Peserta Kuliah.
3) Bagian Akademik (Administrator)
   - Masukan ke sistem: Data Fakultas, Data Jurusan, Data Prodi, Data Kelas, Data Sesi, Data Matakuliah, Data Perkuliahan, Data Bobot Nilai, dan Data Pengguna.
   - Keluaran dari sistem: Informasi Rekap Mahasiswa.

4.3 Desain Masukan

Desain masukan dalam hal ini berupa form-form yang digunakan sebagai antarmuka pengguna untuk berinteraksi dengan sistem. Interaksi yang dimaksud berupa memberikan data sebagai masukan sistem untuk selanjutnya oleh sistem tersebut diproses

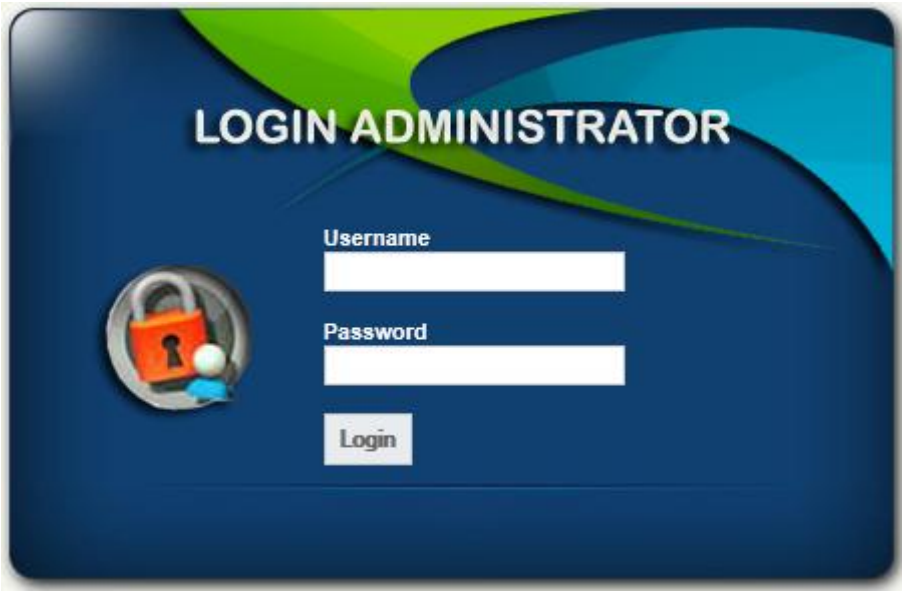
menjadi keluaran, baik keluran bagi proses lain ataupun keluaran bagi pengguna (informasi atau laporan).

4.3.1 Aplikasi Admin

Aplikasi Admin digunakan administrator sistem untuk melakukan berbagai administrasi data yang dibutuhkan oleh Sistem Informasi Akademik. Berikut akan disajikan desain masukan untuk Aplikasi Admin.

4.3.1.1 Halaman Login Admin

Halaman Login digunakan melakukan autentifikasi pengguna untuk masuk ke halaman utama Aplikasi Admin.



Gambar 4.5 Halaman Login Admin

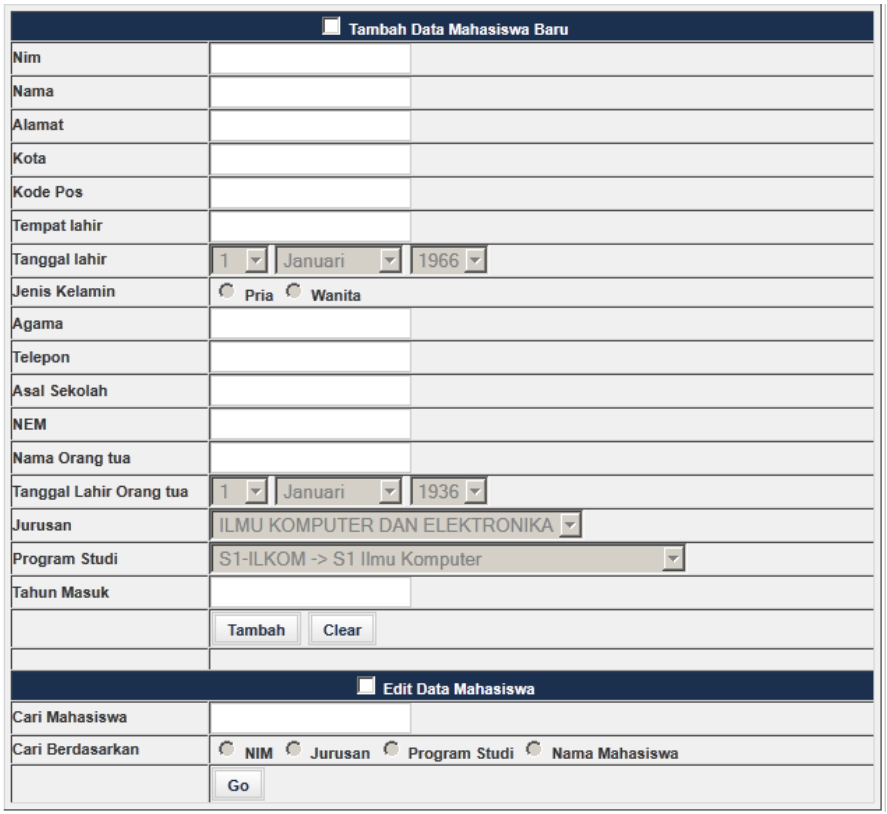
4.3.1.2 Halaman Utama Admin

Halaman Utama akan muncul bila data pengguna (*username* dan *password*) yang dimasukkan melalui Halaman Login valid. Halaman utama berisi beberapa menu untuk melakukan navigasi Aplikasi Admin.

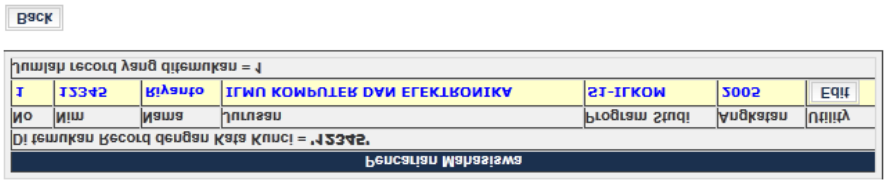
Gambar 4.6 Halaman Utama Admin

4.3.1.3 Halaman Administrasi Data Mahasiswa

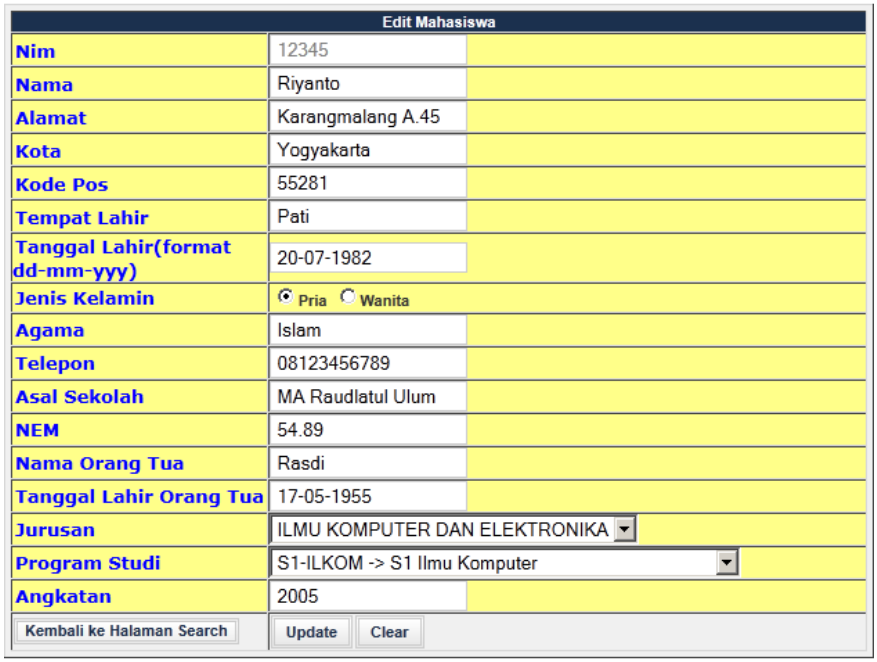
Halaman Administrasi Data Mahasiswa digunakan untuk melakukan administrasi data mahasiswa, seperti: tambah data, pencarian data, perbaharuan data, dan penghapusan data.



Gambar 4.7 Halaman Tambah dan Pencarian Data Mahasiswa



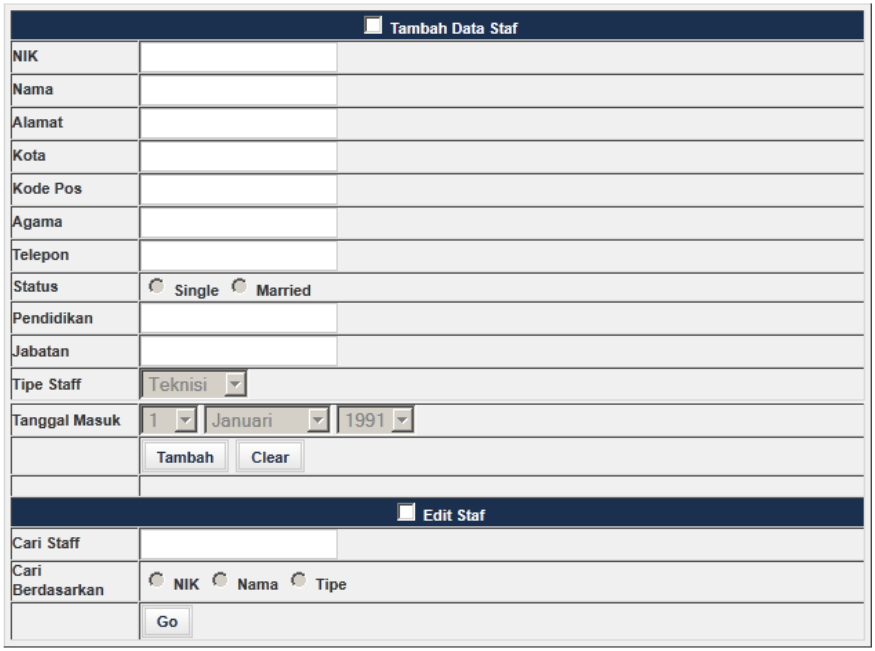
Gambar 4.8 Halaman Hasil Pencarian Data Mahasiswa



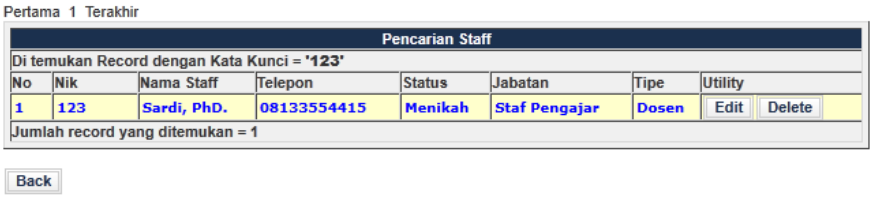
Gambar 4.9 Halaman Edit Data Mahasiswa

4.3.1.4 Halaman Administrasi Data Staf

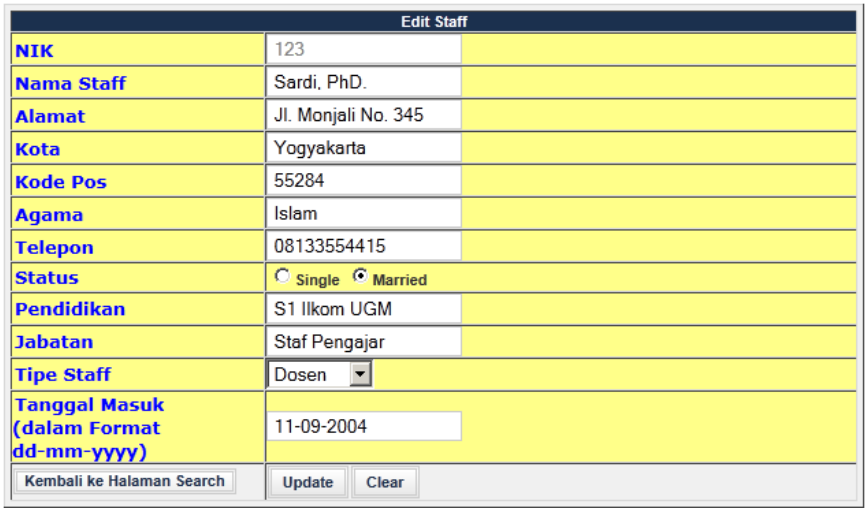
Halaman Administrasi Data Staf digunakan untuk melakukan administrasi data staf pengajar (dosen). Administrasi yang dimaksud adalah tambah data, pencarian data, perbaharuan data, dan penghapusan data staf.



Gambar 4.10 Halaman Tambah dan Pencarian Data Staf



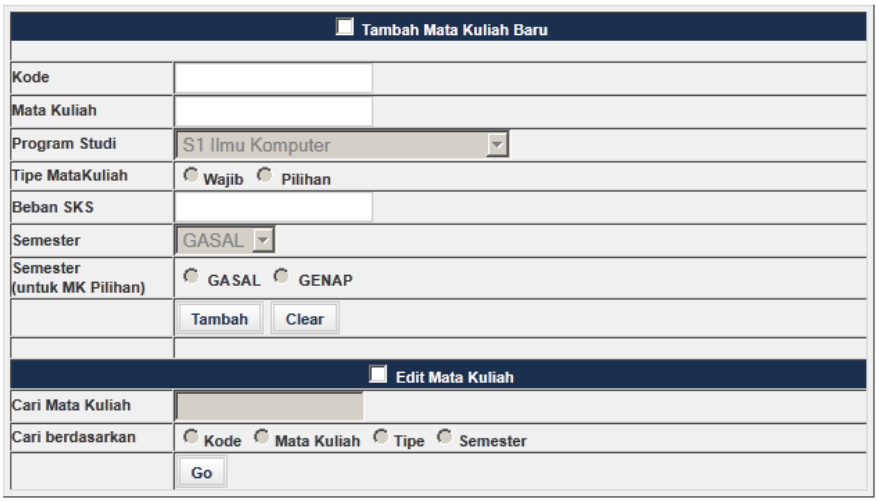
Gambar 4.11 Halaman Hasil Pencarian Data Staf



Gambar 4.12 Halaman Edit Data Staf

4.3.1.5 Halaman Administrasi Data Matakuliah

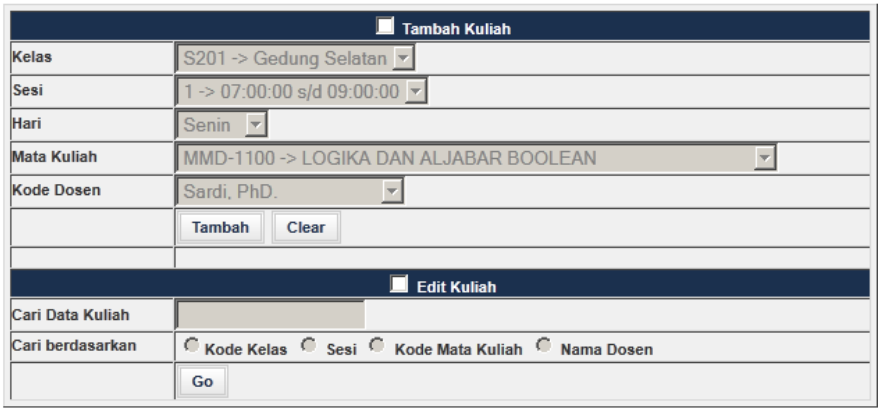
Halaman Administrasi Data Matakuliah digunakan untuk melakukan administrasi data matakuliah. Administrasi yang dimaksud adalah tambah data, pencarian data, perbaharuan data, dan penghapusan data matakuliah.



Gambar 4.13 Halaman Tambah dan Pencarian Data Matakuliah

4.3.1.6 Halaman Administrasi Data Perkuliahan

Halaman Administrasi Data Perkuliahan digunakan untuk melakukan administrasi data perkuliahan. Administrasi yang dimaksud adalah tambah data, pencarian data, perbaharuan data, dan penghapusan data perkuliahan.



Gambar 4.14 Halaman Tambah dan Pencarian Data Perkuliahan

4.3.1.7 Halaman Administrasi Data Kelas

Halaman Administrasi Data Kelas digunakan untuk melakukan administrasi data kelas. Administrasi yang dimaksud adalah tambah data, pencarian data, perbaharuan data, dan penghapusan data kelas.

Gambar 4.15 Halaman Tambah dan Pencarian Data Kelas

4.3.1.8 Halaman Administrasi Data Sesi Perkuliahan

Halaman Administrasi Data Sesi digunakan untuk melakukan administrasi data sesi. Administrasi yang dimaksud adalah tambah data, pencarian data, perbaharuan data, dan penghapusan data sesi.

Gambar 4.16 Halaman Tambah dan Pencarian Data Sesi

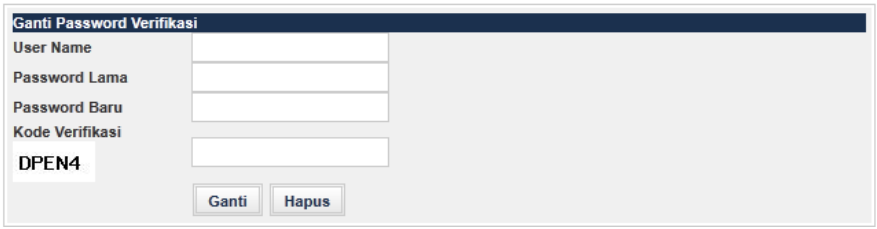
4.3.1.9 Halaman Administrasi Data Aktivasi KRS

Halaman Administrasi Data Aktivasi KRS digunakan untuk melakukan administrasi data jadwal KRS pada semester tertentu, dalam hal ini berupa rentang tanggal. Administrasi yang dimaksud adalah mengaktifkan jadwal KRS pada rentang tanggal pada semester tertentu.

Gambar 4.17 Halaman Aktivasi Jadwal Pengisian KRS

4.3.1.10 Halaman Ganti *Password* Admin

Halaman Ganti *Password* Admin digunakan untuk melakukan administrasi data pengguna untuk masuk ke halaman admin. Administrasi yang dimaksud adalah perbaharuan data *username* dan *password* untuk masuk ke aplikasi admin.

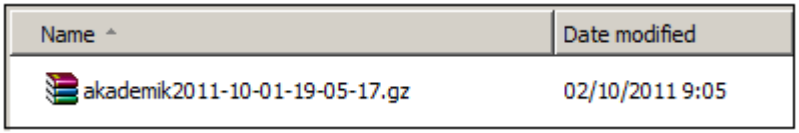


Gambar 4.17 Halaman Ganti Password Admin

4.3.1.11 Halaman *Backup Database*

Halaman *Backup Database* digunakan untuk melakukan proses *backup* data akademik pada waktu tertentu sesuai kebutuhan admin.

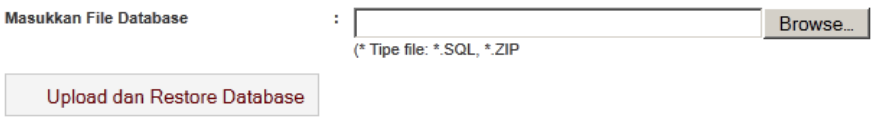
Gambar 4.18 Halaman Backup Database



Gambar 4.19 File Hasil Backup berupa *.gz

4.3.1.12 Halaman *Restore Database*

Halaman *Restore Database* digunakan untuk melakukan proses *restore database*. Tujuannya bila terjadi kesalahan data saat proses administrasi, maka dengan mudah dikembalikan seperti semula melalui halaman ini.



Gambar 4.20 Halaman *Restore Database*

4.3.2 Aplikasi Mahasiswa

Aplikasi Mahasiswa digunakan mahasiswa untuk melakukan berbagai administrasi data yang dibutuhkan oleh

Mahasiswa. Berikut akan disajikan desain masukan untuk Aplikasi Mahasiswa.

4.3.2.1 Halaman Login Mahasiswa

Halaman Login digunakan melakukan autentifikasi pengguna untuk masuk ke halaman utama Aplikasi Mahasiswa.

Gambar 4.21 Halaman Login Mahasiswa

4.3.2.2 Halaman Aktivasi Mahasiswa

Halaman Aktivasi digunakan untuk memperoleh PIN yang akan digunakan untuk masuk ke aplikasi **Mahasiswa** melalui halaman **Login**. Untuk bisa mendapatkan PIN, mahasiswa harus mengisi data-data yang diperlukan, seperti: NIM, NEM, Email, dan Tahun lahir Orang Tua.

Gambar 4.22 Halaman Aktivasi Mahasiswa

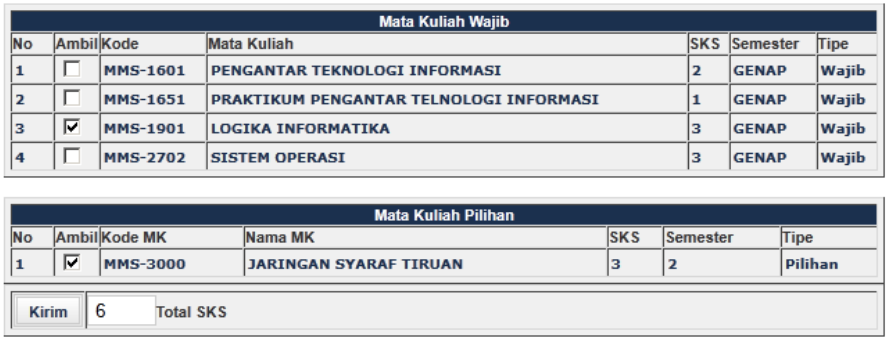
4.3.2.3 Halaman Utama Mahasiswa

Halaman Utama akan muncul bila data pengguna (*username* dan *password*) yang dimasukkan melalui Halaman Login valid. Halaman utama berisi beberapa menu untuk melakukan navigasi Aplikasi Mahasiswa.

Gambar 4.23 Halaman Utama Mahasiswa

4.3.2.4 Halaman KRS

Halaman KRS (Kartu Rencana Studi) digunakan mahasiswa untuk melakukan pengisian dan perubahan KRS sesuai jadwal yang ditentukan.



Gambar 4.24 Halaman Pengisian dan Perubahan KRS

4.3.2.5 Halaman Ganti *Password*

Halaman Ganti *Password* Mahasiswa digunakan untuk melakukan administrasi data pengguna untuk masuk ke Halaman Mahasiswa. Administrasi yang dimaksud adalah perbaharuan data *username* dan *password* untuk masuk ke Aplikasi Mahasiswa.

Gambar 4.25 Halaman Ganti Password Mahasiswa

4.3.3 Aplikasi Staf

Aplikasi Staf digunakan staf pengajar (dosen) untuk melakukan berbagai administrasi data yang dibutuhkan oleh Staf. Berikut akan disajikan desain masukan untuk Aplikasi Staf.

4.3.3.1 Halaman Login Staf

Halaman Login digunakan melakukan autentifikasi pengguna untuk masuk ke halaman utama Aplikasi Staf.

Gambar 4.26 Halaman login Staf Pengajar

4.3.3.2 Halaman Aktivasi Staf

Halaman Aktivasi Staf digunakan untuk memperoleh PIN yang akan digunakan untuk masuk ke Aplikasi Staf melalui Halaman Login Staf. Untuk bisa mendapatkan PIN, Staf harus mengisi data-data yang diperlukan, seperti: NIK, Email, dan Kode yang dikirim Admin melalui Email.

Gambar 4.27 Halaman Aktivasi Staf

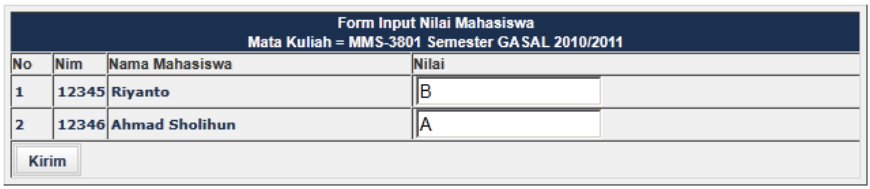
4.3.3.3 Halaman Utama Staf

Halaman Utama Staf akan muncul bila data pengguna (*username* dan *password*) yang dimasukkan melalui Halaman Login Staf valid. Halaman utama berisi beberapa menu untuk melakukan navigasi Aplikasi Staf.

Gambar 4.28 Halaman Utama Staf

4.3.3.4 Halaman Administrasi Data Nilai

Halaman Administrasi Data Nilai digunakan untuk melakukan input dan edit data nilai mahasiswa sesuai dengan matakuliah yang diampu oleh staf.



Gambar 4.29 Halaman Input Nilai

Gambar 4.30 Halaman Edit Nilai

4.3.3.5 Halaman Ganti Password

Halaman Ganti Password Staf digunakan untuk melakukan administrasi data staf untuk masuk ke Halaman Utama Staf. Administrasi yang dimaksud adalah perbaharuan data *username* dan *password* untuk masuk ke Aplikasi Staf.

Gambar 4.31 Halaman Ganti Password Staf

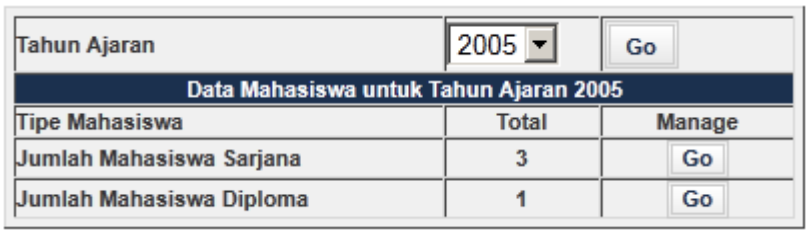
4.4 Desain Keluaran

Jika desain masukan digunakan untuk memasukkan data yang diperlukan sistem, maka desain keluaran merupakan tampilan keluaran yang dihasilkan sistem. Keluaran yang dimaksud dapat berupa data sebagai masukan untuk proses lainnya, atau informasi/laporan untuk entitas yang terlibat, dalam hal ini administrator (bagian akademik), mahasiswa, dan dosen.

4.4.1 Aplikasi Admin

Halaman Rekap Data Mahasiswa

Halaman rekap berisi informasi rekap data mahasiswa berdasarkan jenjang pendidikan, dalam hal ini sarjana dan diploma. Pada halaman ini, administrator dapat mengklik tombol [Go] untuk mengakses data mahasiswa yang diinginkan.

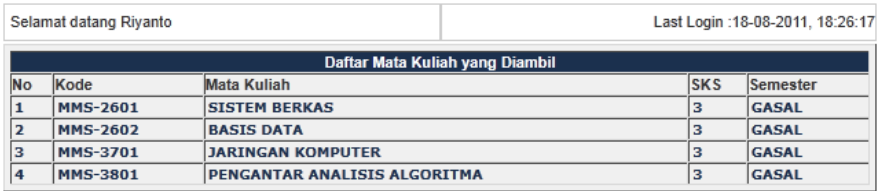


Gambar 4.32 Halaman Rekap Data Mahasiswa

4.4.2 Aplikasi Mahasiswa

4.4.2.1 Halaman KRS

Halaman KRS berisi informasi daftar mata kuliah yang diambil mahasiswa saat melakukan pengisian KRS.



Selamat datang Riyanto — Last Login :18-08-2011, 18:26:17

Daftar Mata Kuliah yang Diambil

No	Kode	Mata Kuliah	SKS	Semester
1	MMS-2601	SISTEM BERKAS	3	GASAL
2	MMS-2602	BASIS DATA	3	GASAL
3	MMS-3701	JARINGAN KOMPUTER	3	GASAL
4	MMS-3801	PENGANTAR ANALISIS ALGORITMA	3	GASAL

Gambar 4.33 Halaman KRS

4.4.2.2 Halaman Jadwal Perkuliahan

Halaman Jadwal Perkuliahan berisi informasi jadwal perkuliahan yang disajikan oleh bagian akademik (administrator). Informasi ini dapat dimanfaatkan mahasiswa sebagai bahan pertimbangan sebelum melakukan proses pengisian KRS.

Selamat datang Riyanto — Last Login :18-08-2011, 18:26:17

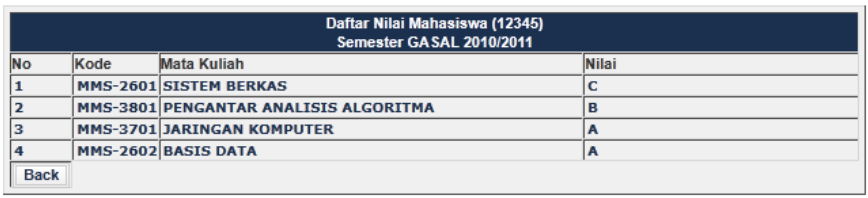
Jadwal Kuliah untuk S1-ILKOM

Kode	Mata Kuliah	Kelas	Hari	Lokasi	Sesi	Dosen Pengampu
MMS-1601	PENGANTAR TEKNOLOGI INFORMASI	S201	Senin	Gedung Selatan	4	Sahid, M.Cs.
MMS-2601	SISTEM BERKAS	S202	Senin	Gedung Selatan	1	Drs. Supirso, M.Eng.
MMS-3000	JARINGAN SYARAF TIRUAN	S202	Senin	Gedung Selatan	5	Sampan, S.Si., M.Kom.
MMS-3801	PENGANTAR ANALISIS ALGORITMA	U201	Senin	Gedung Utara	4	Sampan, S.Si., M.Kom.
MMS-1601	PENGANTAR TEKNOLOGI INFORMASI	S201	Senin	Gedung Selatan	3	Sardi, PhD.
MMS-1901	LOGIKA INFORMATIKA	S201	Senin	Gedung Selatan	5	Mas'ud, Drs, M.Kom
MMS-2602	BASIS DATA	S202	Senin	Gedung Selatan	2	Drs. Supirso, M.Eng.
MMS-3701	JARINGAN KOMPUTER	U201	Senin	Gedung Utara	1	Mas'ud, Drs, M.Kom

Gambar 4.34 Halaman Jadwal Perkuliahan

4.4.2.3 Halaman Nilai

Halaman Nilai berisi informasi daftar nilai yang diperoleh mahasiswa berdasarkan matakuliah yang diambil melalui pengisian KRS pada semester tertentu.



Daftar Nilai Mahasiswa (12345)
Semester GASAL 2010/2011

No	Kode	Mata Kuliah	Nilai
1	MMS-2601	SISTEM BERKAS	C
2	MMS-3801	PENGANTAR ANALISIS ALGORITMA	B
3	MMS-3701	JARINGAN KOMPUTER	A
4	MMS-2602	BASIS DATA	A

Back

Gambar 4.35 Halaman Nilai

4.4.2.4 Halaman KHS

Halaman KHS (Kartu Hasil Studi) berisi informasi rekap nilai keseluruhan matakuliah yang diambil pada periode semester tertentu.

KARTU HASIL STUDI

Nama : Riyanto
NIM : 12345
Program Studi : S1 Ilmu Komputer
Semester : GASAL 2010/2011

NO	KODE	MATA KULIAH	NILAI	SKS	BOBOT
1	MMS-2601	SISTEM BERKAS	C	3	6
2	MMS-3801	PENGANTAR ANALISIS ALGORITMA	B	3	9
3	MMS-3701	JARINGAN KOMPUTER	A	3	12
4	MMS-2602	BASIS DATA	A	3	12

Prestasi Akademik		
Total SKS	:	12
Total Mata Kuliah	:	4
IP Semester	:	3.25

Keterangan :
Kartu Hasil Studi merupakan fasilitas yang dapat digunakan untuk melihat hasil studi mahasiswa persemester. Selain dapat dilihat secara online, hasil studi ini juga dapat dicetak.

Gambar 4.36 Halaman KHS

4.4.2.5 Halaman Transkrip

Halaman Transkrip berisi informasi rekap keseluruhan nilai mulai dari semester awal kuliah sampai semester saat ini.

TRANSKRIP NILAI

Nama : Riyanto
NIM : 12345
Program Studi : S1 Ilmu Komputer

NO.	KODE	MATA KULIAH	NILAI	SKS	SEMESTER
1	MMS-2601	SISTEM BERKAS	C	3	GASAL 2010/2011
2	MMS-3801	PENGANTAR ANALISIS ALGORITMA	B	3	GASAL 2010/2011
3	MMS-3701	JARINGAN KOMPUTER	A	3	GASAL 2010/2011
4	MMS-2602	BASIS DATA	A	3	GASAL 2010/2011
5	MMS-1601	PENGANTAR TEKNOLOGI INFORMASI	A	2	GENAP 2009/2010
6	MMS-1651	PRAKTIKUM PENGANTAR TELNOLOGI INFORMASI	B	1	GENAP 2009/2010
7	MMS-1901	LOGIKA INFORMATIKA	A	3	GENAP 2009/2010
8	MMS-2702	SISTEM OPERASI	B	3	GENAP 2009/2010
9	MMS-3702	PEMROGRAMAN WEB	B	3	GENAP 2009/2010

Prestasi Akademik		
Total SKS	:	24
Total Mata Kuliah	:	9
IP Komulatif	:	3.33

Gambar 4.37 Halaman Transkrip

4.4.3 Aplikasi Staf

4.4.3.1 Halaman Jadwal Mengajar

Halaman Jadwal Mengajar berisi informasi jadwal mengajar berdasarkan matakuliah yang diampu. Pada halaman ini juga dilengkapi informasi hari, kelas, dan sesi mengajar staf tertentu.

Jadwal Perkuliahan
Dosen Pengampu: Sampan, S.Si., M.Kom.

No	Kode	Mata Kuliah	Kelas	Sesi	Hari	Jam Masuk	Jam Keluar
1	MMS-3801	PENGANTAR ANALISIS ALGORITMA	U201	4	Senin	13:01:00	15:00:00
2	MMS-3000	JARINGAN SYARAF TIRUAN	S202	5	Senin	15:01:00	17:00:00
3	MMS-1911	BAHASA INGGRIS I	S201	1	Selasa	07:00:00	09:00:00

Gambar 4.38 Halaman Jadwal Mengajar

4.4.3.2 Halaman Peserta Kuliah

Halaman Peserta Kuliah berisi informasi daftar mahasiswa peserta kuliah berdasarkan matakuliah yang diampu oleh staf tertentu pada semester tertentu.

Daftar Peserta Mata Kuliah MMS-3801
Semester GASAL Tahun Ajaran 2010/2011

No	Nim	Nama Mahasiswa	Alamat	Kota
1	12345	Riyanto	Karangmalang A.45	Yogyakarta
2	12346	Ahmad Sholihun	Karangjati RT/RW:5/37 No. 52	Sleman

Gambar 4.40 Halaman Peserta Kuliah

Lampiran-Lampiran

Lampiran 1 - Teknik Pemodelan Data

A. Normalisasi (*Normal Form Based on Primary Keys*)

Normalisasi adalah proses langkah-langkah yang fleksibel untuk menggantikan hubungan yang ditentukan oleh koleksi yang berurutan dimana hubungan mempunyai suatu struktur yang lebih reguler dan sederhana. Proses normalisasi merupakan proses pengelompokan data elemen menjadi tabel-tabel yang menunjukkan *entity* dan relasinya (Kristanto, 1994). Adapun tahap-tahap normalisasi adalah sebagai berikut.

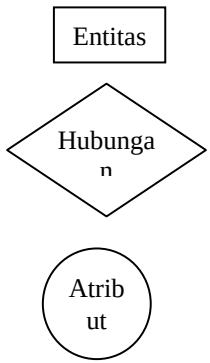
- ☑ **Bentuk Normal Pertama (Firt Normal Form / 1st NF)**
  Bentuk normal pertama mempunyai ciri, yaitu setiap data dibentuk dalam *flat file*, data dibentuk dalam satu record dan nilai dari *field-field* yang menyatakan data tunggal dan bukan gabungan dari nilai-nilai. Tidak ada *set* atribut yang berulang-ulang atau atribut bernilai ganda (*multivalue*).
- ☑ **Bentuk Normal Kedua (Second Normal Form/ 2nd NF)**
  Bentuk normal kedua mempunyai syarat, yaitu bentuk data memenuhi kriteria bentuk pertama. Atribut bukan kunci haruslah bergantung secara fungsional pada kunci utama (*primary key*), sehingga untuk membentuk normal kedua haruslah sudah ditentukan kunci-kunci *field*. Kunci *field* haruslah unik dan dapat mewakili atribut lain yang menjadi anggotanya.
- ☑ **Bentuk Normal Ketiga (Third Normal Form/ 3th NF)**
  Untuk menjadi bentuk ketiga maka relasi haruslah dalam bentuk normal kedua dan semua atribut bukan primer tidak mempunyai hubungan transitif. Dengan kata lain, setiap atribut bukan kunci haruslah bergantung hanya pada *primery key* dan *primery key* secara menyeluruh.
- ☑ **Bentuk Normal Boyce-Codd (Boyce-Codd Norm Form/BCNF)**
  Boyce-Codd Normal Form mempunyai paksaan yang lebih kuat dari bentuk normal ketiga. Untuk menjadi *BNCF*, relasi harus dalam bentuk normal kesatu dan setiap atribut harus bergantung fungsi pada atribut *superkey.*

B. Diagram ER (*Entity Relationship Diagram*)

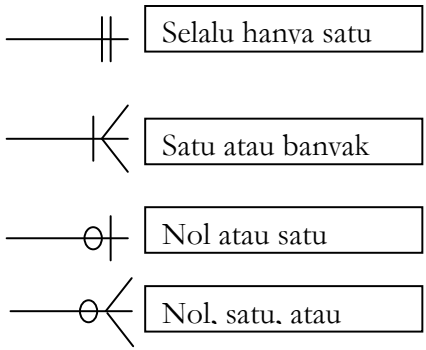
Berikut adalah gambaran singkat mengenai Diagram ER (sering disebut ERD) beserta arti simbol yang digunakan.

1. ERD merupakan model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak.
2. ERD berupa model data konseptual, yang merepresentasikan data dalam suatu organisasi..
3. ERD menekankan pada struktur dan hubungan (*relationship*) data, berbeda dengan DFD (Data Flow Diagram) yang merupakan model jaringan fungsi yang akan dilaksanakan sistem.
4. Biasanya digunakan oleh profesional sistem untuk berkomunikasi dengan pemakai eksekutif tingkat tinggi dalam perusahaan yang tidak tertarik pada pelaksanaan operasi sistem sehari-hari, namun lebih kepada:
   - Data apa saja yang diperlukan untuk bisnis mereka?
   - Bagaimana data tersebut berelasi dengan data lainnya?
   - Siapa saja yang diperbolehkan mengakses data tersebut?

Notasi atau simbol yang digunakan pada perancangan Diagram ER adalah sebagai berikut.



Kardinalitas:



Lampiran 2 - Teknik Pemodelan Proses

A. Bagan Alir (*Flow Chart*)

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. Flowchart menolong analis dan programmer untuk memecahkan masalah ke dalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. Flowchart biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut.

PEDOMAN MEMBUAT FLOWCHART

Bila seorang analis dan programmer akan membuat flowchart, ada beberapa petunjuk yang harus diperhatikan, seperti:

1. Flowchart digambarkan dari halaman **atas** ke **bawah** dan dari **kiri** ke **kanan**.
2. Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini harus dapat dimengerti oleh pembacanya.
3. Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas.
4. Setiap langkah dari aktivitas harus diuraikan dengan menggunakan deskripsi kata kerja, misalkan **MENGHITUNG PAJAK PENJUALAN**.
5. Setiap langkah dari aktivitas harus berada pada urutan yang benar.
6. Lingkup dan range dari aktifitas yang sedang digambarkan harus ditelusuri dengan hati-hati. Percabangan-percabangan yang memotong aktivitas yang sedang digambarkan tidak perlu digambarkan pada flowchart yang sama. Simbol konektor harus digunakan dan percabangannya diletakan pada halaman yang terpisah atau hilangkan seluruhnya bila percabangannya tidak berkaitan dengan sistem.
7. Gunakan simbol-simbol flowchart yang standar.

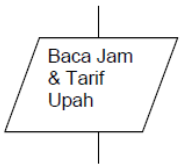
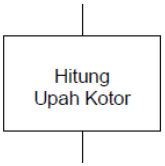
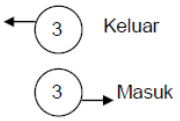
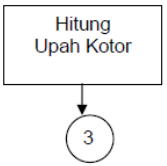
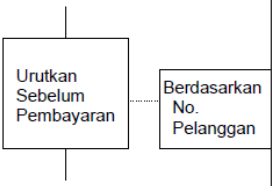
JENIS-JENIS *FLOWCHART*

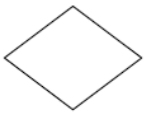
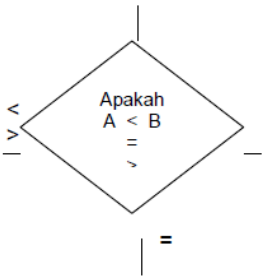
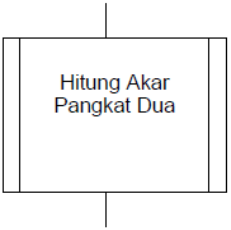
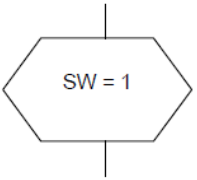
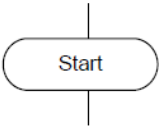
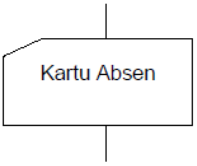
- *Flowchart* terbagi atas lima jenis, yaitu:
- *Flowchart* Sistem (*System Flowchart*).
- *Flowchart* Paperwork/*Flowcart* Dokumen (*Document Flowchart*).
- *Flowchart* Skematik (*Schematic Flowchart*).

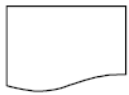
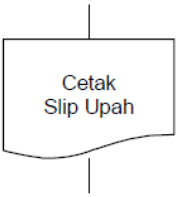
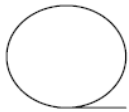
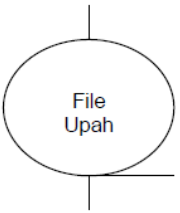
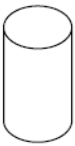
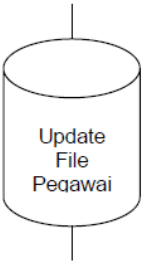
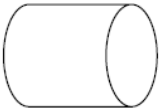
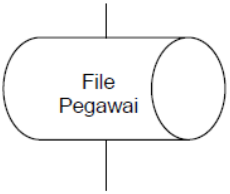
- *Flowchart* Program (*Program Flowchart*).
- *Flowchart* Proses (*Process Flowchart*).

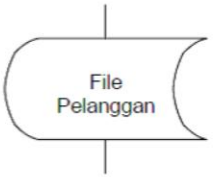
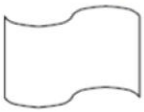
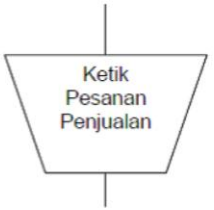
SIMBOL-SIMBOL *FLOWCHART*

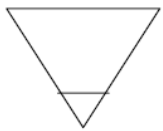
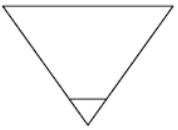
Simbol-simbol *flowchart* yang biasanya dipakai adalah simbol-simbol flowchart standar yang dikeluarkan oleh ANSI dan ISO. Simbol-simbol ini dapat dilihat pada tabel berikut ini.

SIMBOL	ARTI	CONTOH
Input / Output 	Merepresentasikan Input data atau Output data yang diproses atau Informasi.	
Proses 	Mempresentasikan operasi	
Penghubung 	Keluar ke atau masuk dari bagian lain flowchart khususnya halaman yang sama	
Anak Panah	Merepresentasikan alur kerja	
Penjelasan 	Digunakan untuk komentar tambahan	

SIMBOL	ARTI	CONTOH
Keputusan 	Keputusan dalam program	
Predefined Process 	Rincian operasi berada di tempat lain	
Preparation 	Pemberian harga awal	
Terminal Points 	Awal / akhir flowchart	
Punched card 	Input / outuput yang menggunakan kartu berlubang	

SIMBOL	ARTI	CONTOH
Dokumen 	I/O dalam format yang dicetak	
Magnetic Tape 	I/O yang menggunakan pita magnetik	
Magnetic Disk 	I/O yang menggunakan disk magnetik	
Magnetic Drum 	I/O yang menggunakan drum magnetik	

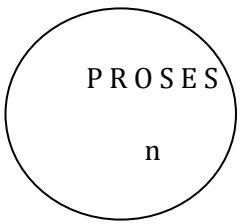
SIMBOL	ARTI	CONTOH
On-line Storage 	I/O yang menggunakan penyimpanan akses langsung	
Punched Tape 	I/O yang menggunakan pita kertas berlubang	
Manual Input 	Input yang dimasukkan secara manual dari keyboard	
Display 	Output yang ditampilkan pada terminal	
Manual Operation 	Operasi Manual	

SIMBOL	ARTI	CONTOH
Communication Link 	Transmisi data melalui channel komunikasi, seperti telepon	Terminal Komputer 
Off-line Storage 	Penyimpanan yang tidak dapat diakses oleh komputer secara langsung	

B. Diagram Alir Data (*Data Flow Diagram*)

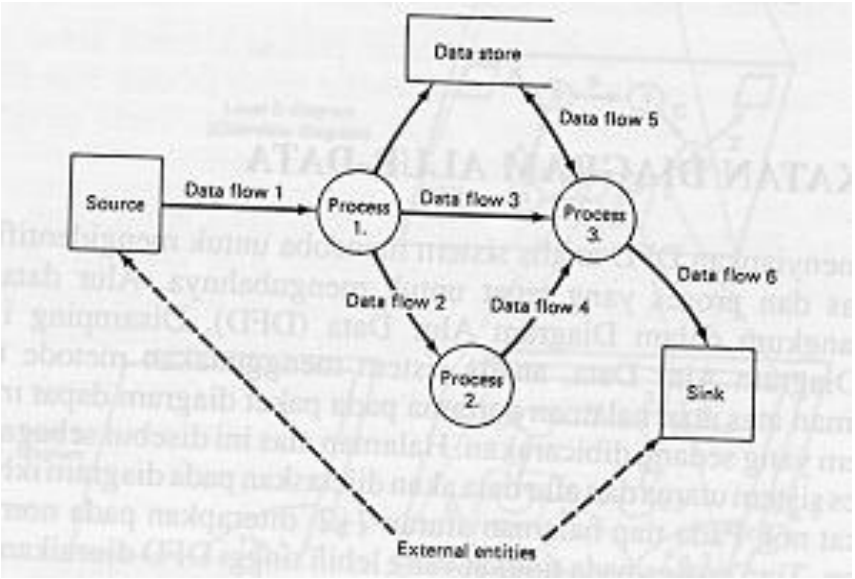
Dengan DAD kita dapat menjelaskan sistem yang ada atau sistem baru yang akan diperkenalkan pada tingkatan logis tanpa harus mempertimbangkan lingkungan fisik tempat data disimpan (misalnya disket atau pita).

SIMBOL-SIMBOL DAD

1. Simbol ALUR DATA ***N a m a*** 	❖ Menunjukkan alur data (informasi/objek) yang mengalir. ❖ Nama alur data menunjukkan nama dari data yang mengalir tersebut, dan bisa lebih dari satu.
2. Simbol PROSES 	❖ Menunjukkan tugas atau proses yang dilakukan baik secara manual atau otomatis. ❖ Simbol Proses ini tidak hanya menunjukkan alur data yang keluar dari proses tersebut, tetapi juga menunjukkan alur data yang masuk dalam proses ini. ❖ Nama proses hendaknya berupa kalimat perintah yang berupa kata

	kerja aktif dan diikuti oleh klausa objek untuk menjelaskan proses tersebut. ❖ n menunjukkan angka referensi dari proses tersebut.
3. Terminator atau Entitas Eksternal N a m a	❖ Merupakan simbol entitas eksternal untuk menunjukkan tempat asal data (sumber) atau tempat tujuan data (Tujuan). ❖ Nama entitas eksternal (terminator) ditulis dalam bentuk tunggal.
4. Penyimpanan Data (Data Store) atau	❖ Terlepas dari media penyimpanan fisik, simbol ini menunjukkan gudang informasi atau data. ❖ Sangat sering terjadi bahwa unsur-unsur data tidak berjalan dari suatu proses ke proses berikutnya secara langsung, melainkan disimpan terlebig dahulu, sementara operasi lainnya atau penyusunan ulang unsur-unsur data berlangsung. ❖ Bila data store hanya diperbaharui selama atau sesudah proses tertentu maka untuk menunjukkan arah alur data ke gudang dibuat gambar anak panah yang mengarah pada gudang data tersebut. ❖ Bila data dari gudang dipakai pada proses itu, maka kita gunakan satu anak panah yang mempunyai dua arah.

Contoh Penggunaannya secara umum :



Gambar 1: Contoh Penggunaan Simbol-simbol DAD

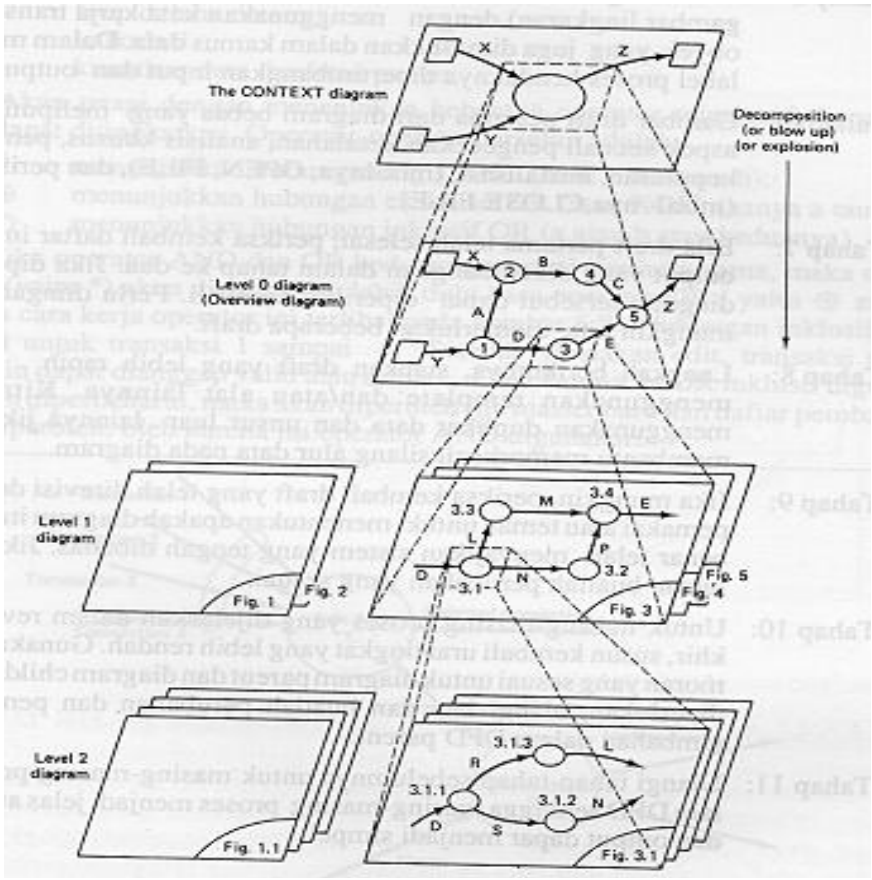
TINGKATAN DALAM DAD

Tingkatan pertama disebut dengan Diagram Konteks (*Context Diagram*), yang menggambarkan mengenai sistem secara global. Dalam hal ini ditetapkan Entitas-entitas eksternal yang terlibat dalam proses, baik sebagai sumber maupun tujuan.

Tingkatan berikutnya dikatakan sebagai Digram level nol (*Zero Diagram* atau *Overview Diagram*), yakni memberikan gambaran mengenai proses-proses apa saja yang akan dilakukan dan melibatkan entitas-entitas eksternal yang ada serta data store –data store tertentu.

Diagram level 1, merupakan penjabaran rinci dari setiap proses yang ada pada diagram level nol, secara khusus. Dimungkin akan muncul proses-proses detilnya. Diagram level 2, merupakan penjabaran rinci dari setiap proses yang baru muncul pada diagram level 1, secara khusus. Dalam hal ini juga dimungkinkan akan muncul proses-proses detilnya.

Tingkatan berikutnya akan kita definisikan sesuai dengan keadaan dari level sebelumnya, dengan harapan diagram ini akan memberikan pemahaman secara detil atau rinci mengenai sistem yang sedang akan dikerjakan. Secara ringkas tingkatan dalam DAD dapat disajikan dalam Gambar 2.



Gambar 2: Tingkatan Diagram Alur Data

KELEBIHAN DAN KELEMAHAN DAD

Berdasarkan uraian sebelumnya, bahwa DAD menggambarkan hal-hal sebagai berikut :

- Adanya pembagian sistem ke dalam sub-sub sistem alur data pada sistem.
- Adanya data store dan alur data (masuk atau keluar) pada sistem.
- Adanya unsur-unsur eksternal, yaitu sumber dan tujuan dari sistem.

Akan tetapi, pada umumnya, DAD tidak menunjukkan:

- Komposisi alur data dalam sistem.
- Syarat akses data dari data store.
- Keputusan dalam sistem.

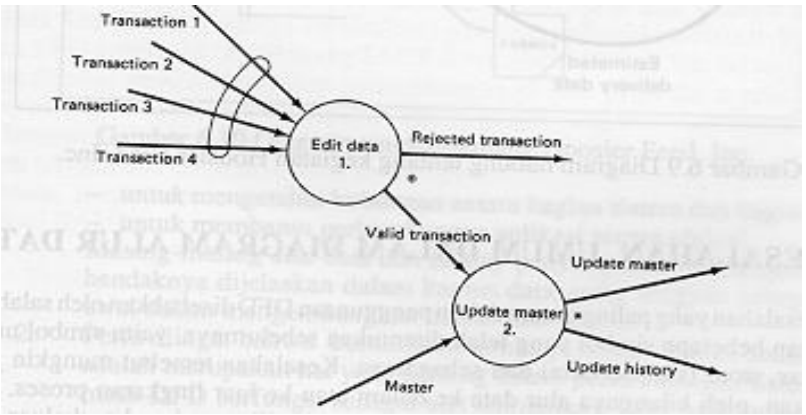
- Loop dalam sistem.
- Kalkulasi.
- Kuatitas data dan atau proses.

OPERATOR LOGIKA DALAM DAD

Dengan menggunakan beberapa operator relasional, kemampuan DAD dapat ditingkatkan. Oprator yang dimaksud yaitu:

- Operator * : menunjukkan hubungan logik AND (baik a maupun b)
- Operator ⊕ : menunjukkan hubungan XOR (hanya a atau b)
- Operator O : menunjukkan hubungan inklusif OR (a atau b atau keduanya)

Contoh :



Gambar 3: Contoh Simbol Operator Logik pada DAD

Berdasarkan Gambar 3, hubungan **inklusif OR** dipakai untuk transaksi 1 s/d 4. Setelah dilakukan proses edit, transaksi tersebut mungkin dianggap valid atau ditolak. Oleh karenai itu **ekslusif OR** digunakan. Sesudah diperbaharui, maka akan diperoleh file master baru dan daftar pembaharuan, untuk itu digunakan operator **AND**.

ALASAN UTAMA PENGGUNAAN DAD

1. Diagram alur data dapat membantu para analis sistem untuk :
   a. meringkas informasi tentang sistem
   b. mengetahui komponen kunci tentang sistem dan membantu dalam menentukan fungsi-fungsi yang dapat dipakai kembali
   c. membantu perkembangan aplikasi secara efektif

2. DAD sebagai alat komunikasi yang baik antara pemakai dengan analis sistem sehingga sangat mudah untuk melakukan kaji ulang secara terus menerus pada perkembangan aplikasi tersebut.
3. DAD menunjukkan syarat-syarat pengaturan waktu dari berbagai macam proses

Daftar Pustaka

Anonim 1. (1999). *Sistem Informasi Manajemen Edisi Kesepuluh*. (Terjemahan Bob Widyohartono) Jakarta: PT. Pustaka Binaman Pressindo.

Anonim 2. (1998). *Management Information System.* Upper Saddl11e River, New Jersey: Prentice-Hall, Inc.

Anonim 3. (2001). *Sistem Informasi Manajemen.* (Terjemahan Hendra Teguh) Jakarta: Pearson Education Asia, PT. Prenhallindo. (Buku asli diterbitkan tahun 1998).

Davis, G. B. (1993). *Kerangka Dasar Sistem Informasi Manajemen*. (Terjemahan Andreas S. Adiwardana) Jakarta: PT. Pustaka Binaman Pressindo.

Jogiyanto, HM. (1999). *Analisis dan Desain Sistem Informasi*. Yogyakarta: Andi Offset.

Laudon, K.C., & Laudon, J.P. (1998). *Managemen Information Systems New Approaches to Organization & Technology.* Upper Sadle River New Jersey: Prentice-Hall.

McLeod, R., Jr. (1995). *Management Information System.* Upper Saddle River, New Jersey: Prentice-Hall, Inc.

Murdick, R. G., Ross, J. E., Clagget, J.R. (1997). *Sistem Informasi untuk Manajemen Modern* edisi ketiga (Terjemahan J. Djamil) Jakarta: Erlangga. (Buku asli diterbitkan tahun 1984).

Onong Uchjana Effendi. (1989). *Sistem Informasi Manajemen.* Bandung: Mandar Maju.

Pressman, RS. (2002). *Rekayasa Perangkat Lunak*. (Terjemahan CN Harnaningrum). Yogyakarta : Andi Offset. (Buku asli "*SOFTWARE ENGINEERING A Practitioner's Approach*" diterbitkan tahun 1997).

Siagian, S.P. (2001). *Sistem Informasi Manajemen untuk Pengambilan Keputusan.* Bandung: Remadja Karya.

Silberschatz, A. etc. (1986). *Database System Concepts*. New York: McGraw-Hill Book Company.

Singh A. (Maret 2005). Telecommunications system & internet communications. *Journal of information technology and libraries.* Diambil pada tanggal 21 Mei 2005 dari http://proquest.umi.com/pqdweb.

Stoner, J.A.F. & Freeman, R.A. (2000). *Management.* Englewood Cliffs, New Jersey: Prentice-Hall International Editions.

Profil Penulis

Dr. Lantip Diat Prasojo. Lahir di Magetan, 25 April 1974. Saat ini tercatat sebagai dosen tetap di Prodi Manajemen Pendidikan Fakultas Ilmu Pendidikan dan Pascasarjana UNY. Penulis menyelesaikan S1 Teknik Elektro di UGM, S2 Manajemen Pendidikan di UNY (memperoleh gelar Magister Pendidikan dalam waktu 19 bulan dengan predikat *Cumlaude*) dan S3 Prodi Administrasi/ Manajemen Pendidikan UPI Bandung (memperoleh gelar Doktor dalam waktu 2 tahun dengan predikat *Cumlaude*). Beberapa mata kuliah yang diampu adalah Manajemen Strategik, TQM, Praktik Manajemen Pendidikan, Manajemen Pendidikan, Sistem Informasi Manajemen (SIM), TIK Manajemen dan Manajemen Perkantoran. Penulis pernah ditugasi UNY sebagai Ketua Laboratorium Jurusan Administrasi Pendidikan, Koordinator ISO Pascasarjana UNY, Manajer LIMUNY Puskom UNY, dan Sekretaris Prodi S2 dan S3 Manajemen Pendidikan Pascasarjana UNY. Selain itu, penulis juga pernah membantu Dirjen PMPTK Kementerian Pendidikan Nasional dalam Proyek BERMUTU, Fasilitator tingkat nasional untuk diklat kepala sekolah, Tim CPD (*Continuous Professional Development*) untuk Kepala Sekolah di seluruh Indonesia dan sebagai Asesor BAN PT Kemdiknas. Penulis dapat dihubungi melalui email: lantip1975@gmail.com.

SISTEM INFORMASI MANAJEMEN PENDIDIKAN

Profil Penulis

Dr. Lantip Diat Prasojo. Lahir di Magetan, 25 April 1974. Saat ini tercatat sebagai dosen tetap di Prodi Manajemen Pendidikan Fakultas Ilmu Pendidikan dan Pascasarjana UNY. Penulis menyelesaikan S1 Teknik Elektro di UGM, S2 Manajemen Pendidikan di UNY (memperoleh gelar Magister Pendidikan dalam waktu 19 bulan dengan predikat Cumlaude) dan S3 Prodi Administrasi/Manajemen Pendidikan UPI Bandung (memperoleh gelar Doktor dalam waktu 2 tahun dengan predikat Cumlaude). Beberapa mata kuliah yang diampu adalah Manajemen Strategik, TQM, Praktik Manajemen Pendidikan, Manajemen Pendidikan, Sistem Informasi Manajemen (SIM), TIK Manajemen dan Manajemen Perkantoran. Penulis pernah ditugasi UNY sebagai Ketua Laboratorium Jurusan Administrasi Pendidikan, Koordinator ISO Pascasarjana UNY, Manajer LIMUNY Puskom UNY, dan Sekretaris Prodi S2 dan S3 Manajemen Pendidikan Pascasarjana UNY. Selain itu, penulis juga pernah membantu Dirjen PMPTK Kementerian Pendidikan Nasional dalam Proyek BERMUTU, Fasilitator tingkat nasional untuk diklat kepala sekolah, Tim CPD (Continuous Professional Development) untuk Kepala Sekolah di seluruh Indonesia dan sebagai Asesor BAN PT Kemdiknas. Penulis dapat dihubungi melalui email: **lantip1975@gmail.com**

ISBN 602-7981-01-6

UNY PRESS

Jl.H.Affandi (Jl.Gejayan), Gg. Alamanda,
Kompleks FT-UNY, Kampus Karangmalang, Yogyakarta,
Kode Pos:55281,Telp.(0274)589346,
unypress.yogyakarta@gmail.com