ANALISIS ALAT MODEL (*MODELING TOOLS*), SEBAGAI VISUALISASI SISTEM INFORMASI

Deliza Henny
Fakultas Ekonomi Universitas Trisakti

Abstract

Technical analysis by using modeling tools is one of the methods for understanding the system. Each of modeling tools visualized the system by diagram, and represented system from many sides.

The purpose of this paper is to elaborate and understanding the modeling tools as a means that used in developing the system, and as a analytical device for understanding the system.

Some of modeling tools those are studied in this paper, i.e. : Data Flow Diagram (DFD), Flowchart, Entity Relationship Diagram (ERD), State Transaction Diagram (STD).

Keywords: Data Flow Diagram, Flowchart, Entity Relationship Diagram

PENDAHULUAN

Pemahaman tentang Komputer dan Sistem Informasi tidak hanya mempelajari *hardware*, *software* dan data, namun juga perlu mengetahui tentang komponen lainnya, yang dikenal dengan 5 komponen CIS (*Computer Information System*). Ke lima komponen tersebut adalah : *people*, *procedures*, *data*, *hardware*, dan *software*.

Satu hal yang terkait dengan ke lima komponen CIS dimaksud adalah *modeling tools*, yang merupakan metodologi pe-modelan sistem. Ada empat macam *modeling tools*, dimana masing-masing mempunyai tujuan yang berbeda

dan merupakan sarana bagi analis dalam upaya memahami kebutuhan pemakai dan menggunakannya untuk membangun sebuah sistem informasi. Berikut ini akan dijelaskan secara singkat mengenai *modeling tools* dimaksud :

a. Model Proses

Data Flow Diagram (DFD) disebut juga model proses karena langsung mem-fokuskan pada elemen yang diperlukan dalam proses transformasi data menjadi informasi. Juga dalam penggambaran data inflow, outflow dan proses. DFD memperlihatkan keterkaitan yang erat antara ketiga komponen CIS, yaitu antara *people, data* dan *procedures*.

b. Model Data

Menjelaskan tentang struktur data yang digunakan dalam sebuah sistem informasi, baik data yang ada didalam, diantara dan yang keluar dari sistem. Model data digunakan sebagai alat untuk mendefinisikan kebutuhan database suatu organisasi. Agar dapat memahami model data ini, perlu mengetahui dua hal yaitu : bagaimana dasar dari proses sebuah file (*file processing fundamental)* dan bagaimana file-file saling terkait melalui sebuah field yang berfungsi sebagai "key" (*relational database*). Entity Relationship Diagram (ERD) adalah alat untuk mendefinisikan kebutuhan tentang database suatu organisasi.

c. Model Sistem

Model ini berkaitan dengan dua komponen sistem yang tidak ada pada dua model sebelumnya (model proses dan model data). Dua komponen sistem tesebut adalah komponen perangkat keras (*hardware*) dan komponen perangkat lunak (*software*). Berbagai model sistem dibuat untuk memberikan gambaran tentang *hardware* dan *software* dengan penjelasan yang mudah dipahami oleh pemakai. Telah diakui bahwa model sistem sangat berperan membantu pemakai (*user*) dalam memahami tahap-

tahap Siklus Daur Hidup Sistem atau *System Development Life Cycle* (SDLC).

d. Model Obyek

Model ini lebih menekankan pada obyek dibandingkan pada *procedures* atau data. Kunci utamanya adalah mengetahui apa yang menjadi obyek perusahaan. Secara umum definisi dari obyek perusahaan adalah karakteristik data dan fungsi-fungsi atau proses-proses yang tampil mewakili kinerja perusahaan. Jadi data dan komponen proses dari suatu sistem informasi bergabung bersama dalam suatu obyek perusahaan.

PEMBAHASAN

Data Flow Diagram (DFD)

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau lingkungan fisik dimana data tersebut akan disimpan. DFD merupakan alat yang digunakan pada metodelogi pengembangan sistem yang terstruktur. DFD juga merupakan alat yang dapat menggambarkan aliran data (data flow) didalam sistem dengan struktur yang jelas. Lebih lanjut DFD juga merupakan dokumentasi sistem yang baik.

a. Notasi Baku DFD

DFD menyajikan suatu gambaran tentang bagaimana *people* dan *procedures* mentransformasi data menjadi informasi. Data digambarkan mengalir dalam sistem dengan cara yang dikenal sebagai *input-proses-output* (IPO). Berikut ini penjelasan 4 simbol DFD.

- Kesatuan luar (*External entity)* atau terminator .

  Setiap sistem pasti mempunyai batas sistem (*boundary*) yang memisahkan suatu sistem dengan lingkungan luarnya. Sistem akan menerima input dan menghasilkan output kepada lingkungan luarnya.

Kesatuan luar (*external entity*) merupakan kesatuan (*entity*) di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan input atau menerima output dari sistem. Umumnya terminator adalah orang atau sekelompok orang, contoh : pelanggan yang melakukan order suatu produk atau manajer yang melakukan evaluasi laporan penjualan mingguan. Dapat dijelaskan juga bahwa terminator adalah sumber data atau penerima informasi, (*data sources* atau *sinks*).
Terminator bisa berbentuk suatu organisasi atau sistem informasi lain, sepanjang ia sebagai sumber data atau penerima informasi. *External entity* atau terminator digambarkan dengan bentuk kotak atau suatu kotak dengan sisi kiri dan atasnya berbentuk garis tebal dan diberi label nama data elemen lingkungan yang dimaksud.

Gambar 1. External Entity

Beberapa hal yang perlu diketahui berkenaan dengan terminator, yaitu :

a) Terminator berada diluar sistem yang dimodelkan, aliran menghubungkan terminator dengan berbagai proses atau penyimpanan dan juga merupakan *interface* antara sistem dengan dunia luar.
b) Terminator berada diluar jangkauan perubahan sistem yang didesain. Jadi desain sistem yang dibuat harus fleksibel dan bebas untuk memiliki pilihan yang terbaik (efisien, handal dan sebagainya)

- ***Data Flow* (aliran data)**

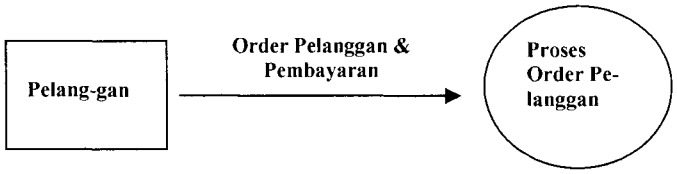
Data *flow* atau aliran data digambarkan sebagai sebuah anak panah menuju atau meninggalkan suatu proses. Aliran data digunakan untuk

menjelaskan perpindahan sekelompok informasi dari suatu bagian sistem ke bagian lainnya. Jadi aliran data menunjukkan pengarahan data. Lambang aliran data adalah seperti ini :

Gambar 2. Data Flow

Aliran data diberi nama jelas dan mempunyai arti, yang dituliskan disamping garis panahnya. Beberapa konsep dalam penggambaran aliran data di DFD :

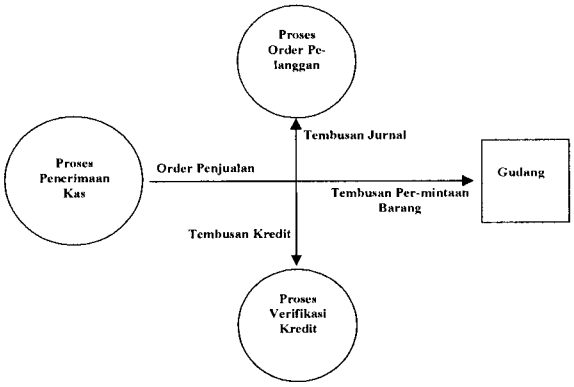
a) Konsep Paket Data (*packet of data*)
Bila dua atau lebih data mengalir dari suatu sumber yang sama ke tujuan yang sama, maka harus dianggap sebagai suatu arus data tunggal, karena data tersebut mengalir bersama sebagai suatu paket. Data yang mengalir bersama-sama harus ditunjukkan sebagai suatu aliran data, walaupun misalnya terdiri dari beberapa dokumen. Contoh : dua aliran data Order pelanggan dan Pembayaran, harus ditunjukkan sebagai aliran data yang tunggal, yaitu sebagai aliran data "Order Pelanggan dan Pembayaran" yang menuju ke proses : "Proses Order Pelanggan".



Gambar 3. Pocket of Data Flow

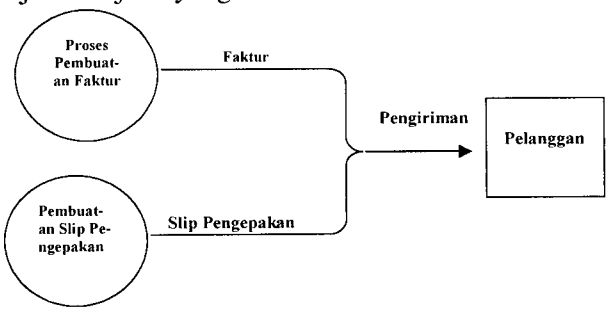
b) Konsep Aliran Data Menyebar (*diverging data flow*).
Aliran data yang menyebar menunjukkan sejumlah tembusan dari aliran data yang sama dari sumber data yang sama ketujuan yang berbeda. Contoh : aliran data "order penjualan" mempunyai 3 tembusan, yaitu tembusan untuk jurnal yang mengalir (menuju) ke

proses pembuatan faktur, tembusan permintaan barang yang mengalir ke *external entity* gudang, dan tembusan kredit yang mengalir ke proses veifikasi kredit. Konsep ini menunjukkan bahwa 3 tembusan (3 *copy*) aliran data tersebut mempunyai struktur elemen sama, karena merupakan hasil dari tembusan aliran data order penjualan.



Gambar 4. Diverging Data Flow

c) Konsep Aliran Data Mengumpul (*converging data flow*)
Aliran data yang mengumpul menunjukkan beberapa aliran data yang berbeda dari sumber yang berbeda bergabung bersama menuju ke tujuan yang sama.

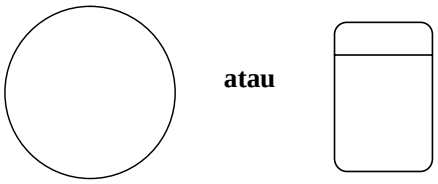


Gambar 5. Converging Data Flow

d) Konsep Sumber Dan Tujuan Arus Data
Semua aliran data harus dihasilkan dari suatu proses atau menuju ke suatu proses. Atau bisa saja satu atau kedua-duanya, yaitu berasal dari suatu proses menuju ke bukan suatu proses atau berasal dari bukan suatu proses tetapi menuju ke suatu proses, atau berasal dari suatu proses dan menuju ke suatu proses. Konsep ini penting karena aliran data adalah salah satu dari hasil suatu proses atau akan digunakan untuk melakukan suatu proses.

- **Proses (*Process*)**

Proses yang biasa disebut dengan fungsi, atau sebuah transformasi. Proses menunjukkan bagaimana satu atau lebih *input* menjadi *output*. Proses dilambangkan dengan lingkaran atau persegi panjang yang diberi label nama proses yang dimaksud.



Gambar 6. Proses

Setiap proses harus diberi kejelasan yang lengkap meliputi berikut ini :

a) Identifikasi Proses
Identifikasi ini umumnya berupa suatu angka yang menunjukkan nomor acuan dari proses dan ditulis pada bagian atas di simbol proses.

b) Nama Proses
Nama proses menunjukkan apa yang dikerjakan oleh proses tersebut. Nama proses harus jelas, lengkap dan menggambarkan kegiatan prosesnya, yang biasanya diawali dengan

kata kerja (misal : membuat, membandingkan, menghitung, mempersiapkan). Nama proses diletakkan dibawah nomor proses.

c) Pemroses
Pemroses ini menunjukkan siapa atau dimana suatu proses dilakukan.

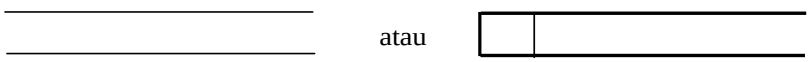
Keterangan simbol proses dapat dicantumkan dibawah nama proses. Berbagai kemungkinan aliran data dalam suatu proses :

a) Proses menerima satu aliran data dan menghasilkan satu aliran data
b) Proses menerima lebih dari satu aliran data dan menghasilkan satu aliran data
c) Proses menerima satu aliran data dan menghasilkan lebih dari satu aliran data.

Jadi dapat disimpulkan bahwa suatu proses harus menerima aliran data dan menghasilkan aliran data. Bisa terjadi kesalahan dalam proses pada pembuatan DFD, yang umumnya adalah sebagai berikut :

a. Proses mempunyai *input* tetapi tidak menghasilkan *output* . Kesalahan ini disebut lubang hitam (*black hole)*, karena data masuk kedalam proses dan lenyap tidak berbekas seperti masuk kedalam lubang hitam yang dalam.
b. Proses menghasilkan output tetapi tidak pernah menerima input dan kesalahan ini disebut dengan ajaib (*miracle*), karena secara ajaib dihasilkan output tanpa pernah menerima input.

- *Data Stores* (penyimpanan data)
Data stores adalah tempat penyimpanan data yang sedang tidak aktif. Notasi dari penyimpanan ini berupa 2 garis paralel :



Gambar 7. Data Stores

Secara umum pada proses data, penyimpanan seringkali merujuk kepada *file* atau *database*, seperti : *file* pita *magnetik* atau *file disk* yang diorganisir oleh sistem manajemen *database*. Dalam penggambaran penyimpanan pada DFD perlu memperhatikan beberapa hal sebagai berikut :

a. Hanya proses saja yang berhubungan dengan penyimpanan, karena yang menggunakan atau merubah data pada penyimpanan adalah proses
b. Aliran data yang menuju ke penyimpanan data dari suatu proses menunjukkan proses *update* terhadap data yang tersimpan.
c. Aliran data yang berasal dari penyimpanan ke suatu proses menunjukkan bahwa
d. proses tersebut menggunakan data dalam penyimpanan tersebut.
e. Suatu proses yang melakukan kedua-duanya, yaitu menggunakan dan update data dari penyimpanan data.

Bentuk *Data Flow Diagram* (DFD)

Ada dua bentuk DFD : *physical* DFD dan *logical* DFD, penjelasannya berikut ini :

a. Physical DFD (PDFD)

Physical DFD lebih menekankan pada bagaimana proses dari sistem diterapkan
(dengan cara apa, oleh siapa dan dimana). PDFD lebih tepat digunakan untuk menggambarkan sistem lama, dimana dari kerangka PDFD ini lebih dapat dikomunikasikan kepada pengguna sistem, dan bagi analis sistem akan dapat gambaran yang jelas bagaimana sistem bekerja.

***b. Logical* DFD (LDFD)**

LDFD menekankan proses-proses apa yang terdapat di sistem, sehingga lebih sesuai digunakan untuk menggambarkan sistem yang akan diusulkan. LDFD tidak menekankan pada bagaimana sistem diterapkan, tapi lebih mementingkan hanya pada logika dari kebutuhan-kebutuhan sistem, yaitu proses apa yang secara logika dibutuhkan sistem.

Pedoman Pembuatan DFD

Beberapa hal yang perlu diperhatikan dalam pembuatan DFD, yaitu :

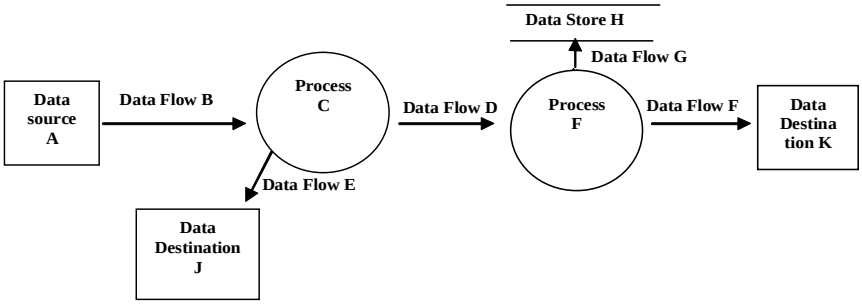
a. *Data flow* tidak boleh menghubungkan secara langsung *external entity* dengan *data store* atau dengan *external entity* lain, harus melalui sebuah proses.
b. *Data flow* tidak boleh menghubungkan *data store* dengan *data store* lain, dalam hal ini harus melalui sebuah proses.
c. *Data store* harus punya paling tidak satu *entry data flow* dan satu *exit data flow*, dan bisa saling berhubungan melalui suatu proses.
d. Setiap proses harus punya satu *entry data flow* dan satu *exit data flow.*
e. Pemilihan nama proses, aliran data, *data store* dan *terminator*, harus memiliki pengertian yang sesuai.
f. Penggambaran ulang DFD sebanyak yang diperlukan (level).
g. Menghindari DFD yang kompleks.
h. Menjaga DFD agar konsisten, baik terhadap DFD tersebut maupun dengan DFD lain yang berhubungan.

Berikut ini adalah tahapan dalam pembuatan DFD :

a. Identifikasikan semua *external entity* atau terminator yang terlibat di sistem, terminator merupakan sumber aliran data ke sistem dan tujuan penerima arus data dari hasil proses sistem.
b. Identifikasikan semua input dan output yang terlibat dengan *external entity*
c. Menggambarkan bagan berjenjang untuk keseluruhan proses yang ada pada sistem, yang disebut *Event Decomposition Diagram*. Tujuan penggambaran diagram ini adalah untuk kepentingan penggambaran DFD pada level bawah.
d. Membuat diagram *event*, yang dimaksud *event* adalah kejadian yang terjadi dalam lingkungan sistem dan berkaitan dengan respon yang keluar dari sistem.
e. Membuat diagram sistem, yang bertujuan untuk menggambarkan secara lengkap seluruh kejadian yang terjadi pada sistem, yang merupakan gabungan dari semua diagram event yang sebelumnya telah dibuat.

f. Menyusun diagram konteks (*context diagram*), yang merupakan tahap awal dalam penggambaran DFD, yaitu diagram yang menggambarkan keseluruhan sistem dalam suatu *single process* yang terhubung hanya dengan *external entities*[1]. Penggambaran sistem pertama kali secara garis
g. besar yang tertuang pada diagram konteks menurut pendekatan terstruktur
h. disebut juga *top level*, dan penggambarannya akan memberikan pemahaman mengenai batasan sistem (*system boundary*). Dalam suatu kontrak kerja desain sistem, diagram konteks sangat relevan terutama dalam penentuan cakupan sistem yang menjadi tanggung jawab *system designer*.
i. Kelanjutan dari diagram konteks adalah penggambaran sistem secara lebih rinci (disebut *lower level*). Jadi dilakukan suatu dekomposisi diagram konteks, penggambaran secara terinci ini disebut juga *overview diagram.* Penggambaran DFD *level* 1 (secara garis besar) dibuat sebagai rincian dari sistem. *Leveling process* dapat terus dijalankan dengan penggambaran lebih detail sampai DFD l*evel* 2, *level* 3, dan seterusnya. Secara teoritis jumlah level yang disusun tidak terbatas, namun bila dekomposisi DFD tidak lagi memberikan detail sistem yang lebih jelas, maka selanjutnya beralih ke tugas atau model lain. Berarti pada keadaan tersebut dekomposisi dari DFD telah selesai.

Contoh sederhana DFD yang menggambarkan suatu proses data :



Gambar 8. Contoh DFD

- Input untuk proses C adalah data flow B yang berasal dari data source A
- Output proses C adalah data flow D dan E
- Data flow E dikirim ke data destination J
- Data flow D digunakan sebagai input bagi proses F
- Proses F menggunakan juga input data flow G
- Output dari proses F adalah data flow I dan data flow G
- Data flow G berasal dari data store H
- Data flow I dikirim ke data destination K

***Entity Relationship Diagram* (ERD)**

Entity Relationship Diagram (ERD) merupakan alat model data untuk menggambarkan obyek dalam bentuk entitas dan relasinya berdasarkan keinginan dari pemakai sistem. Sehingga keterkaitan obyek data dalam suatu sistem akan terlihat jelas oleh pemakai dan perancang sistem.

a. Bagian dari ERD

ERD mencakup empat hal yaitu : *entities*, *attributes*, *relationships,cardinality*, yang penjelasannya akan dibahas dibawah ini.

- *Entities*

  Entiti adalah suatu file data atau sesuatu tempat untuk menyimpan data, sebutan lainnya yaitu *entity type*, *entity class* atau *file.* Simbol entiti adalah bentuk segi empat dan dapat berupa orang, tempat, obyek, kejadian maupun konsep. Isi dari suatu entiti disebut *instance*, contoh : entiti mahasiswa mempunyai *multiple instances* yaitu Ina, Hesy, Tuty, Ratna.

- *Attributes*

  Attributes adalah karakteristik suatu entiti, sebutan lainnya *elemen*, *property*, *field.* Beberapa atribut digabung menjadi superatribut, yang disebut sebagai *compound attribute.* Contoh : nama mahasiswa adalah salah satu atribut entiti mahasiswa, yang mencakup nama belakang, nama depan, nama panggilan.

- *Relationships*

  Relationships adalah hubungan yang ada diantara satu atau lebih entiti. Contoh : hubungan antara entiti mahasiswa dengan kurikulum (kelas) yaitu :

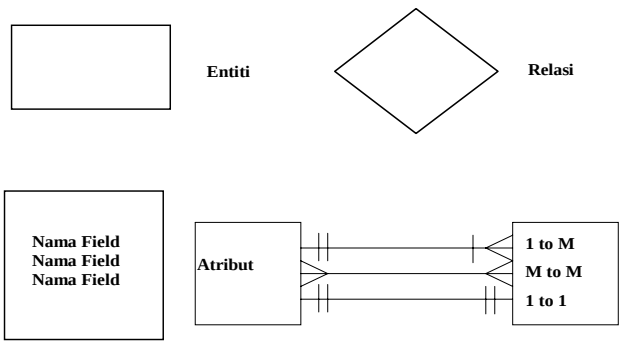
  - satu mahasiswa terdaftar pada satu atau lebih kurikulum
  - satu kurikulum diikuti oleh nol, satu atau lebih mahasiswa

- *Cardinality*

  Cardinality menjelaskan jumlah minimum dan maksimum kejadian (*occurrences)* dari satu entiti dengan entiti lain yang terkait. Karena semua hubungan adalah dua arah (*bi-directional*), maka kardinaliti harus ada pada kedua arah tersebut. Contoh : banyak dosen (entiti) mempunyai relasi mengajar pada banyak mahasiswa (entiti), kardinalitinya adalah *Many to Many* (M to M).

b. Simbol dari ERD

Ada beberapa macam simbol ERD, salah satunya adalah menurut David Harris dalam bukunya System Analysis & Design (2nd edition, 2000) simbol ERD adalah :



Gambar 9. ERD Symbols

c. Relational Database

Relational database adalah kumpulan dari file yang berhubungan oleh karena adanya suatu field umum (*common fields*), atau suatu struktur data yang mempunyai hubungan dengan elemen data lain, baik dalam suatu file atau dalam file lain.

Relational database dapat digambarkan dalam bentuk tabel (contoh di bawah). Kolom tabel menunjukkan atribut dari file, dimana atribut ini menunjukkan item dari data (*field*). Kumpulan field disebut *domain*. Sedangkan setiap baris pada tabel disebut *tuple (record)*. *Tuple* yang mempunyai dua domain disebut 2 *tuple* (*record*), jika mempunyai tiga domain disebut 3 tuple, demikian seterusnya.

Setiap record dalam tabel mempunyai suatu field yang unik, dengan cara mana record ini dapat di-identifikasikan, dan field yang unik ini disebut *Primary key* .

Contoh relational database dalam bentuk tabel :

TABEL 1
RELATIONAL DATA BASE

Nilai MHS (nama file)

No_mhs	Kode_mk	Sem	Tahun	Nilai
9901	MKA781	1	99/00	A
9902	MKA782	1	99/00	B
9903	MKP654	2	98/99	B
....	....	....	....	....
....	....	....	....	....

Atribut (field)

tuple (record)

key field

domai

Relational database tidak selalu dalam bentuk tabel, dapat ditulis dalam bentuk notasi berikut ini, contoh : NilaiMHS (no_mhs, kode_mk, sem, tahun, nilai).

Nama paling depan dari notasi ini yang ditulis diluar tanda kurung adalah nama file. Sedangkan nama-nama dalam tanda kurung yang ditulis dengan koma sebagai pemisah, adalah nama field. Nama field yang digaris bawahi adalah *primary key* (PK).

File yang berisi atribut atau group field yang berulang-ulang (*repeating group*) akan menyebabkan akses data di file menjadi lambat serta memboroskan tempat (*space*) penyimpanan, karena nilai field tersebut selalu ditulis dalam file. Disain database harus memenuhi kondisi tidak mengandung anomali atau kejanggalan pada penempatan atribut tertentu dari suatu obyek data.

Mengatasi masalah ini diperlukan normalisasi, yang merupakan suatu proses dimana setiap file dalam database di rancang kembali dengan masing-masing atribut yang ada dalam file tersebut hanya tergantung pada key field saja. Beberapa bentuk normalisasi :

a) Normalisasi Bentuk Pertama (*first normal form/ 1NF*)
Database disebut sebagai 1NF jika dan hanya jika setiap atribut data relasi tersebut hanya memiliki nilai tunggal dalam satu baris data atau record, tidak ada lagi repeating group. Cara yang dilakukan adalah memisahkan atribut atau *group field* yang berulang tersebut kedalam file yang terpisah.

b) Normalisasi Bentuk Kedua (*second normal form/ 2NF*)
suatu relasi memenuhi 2NF jika dan hanya jika sudah memenuhi 1NF dan setiap atribut yang bukan *primary key* (PK) tergantung secara fungsional terhadap semua PK

c)* Normalisasi Bentuk Ketiga *(third normal form/ 3NF)
Jika dan hanya jika sudah dalam kondisi 2NF dan file telah memuat semua relasi dependen dimana setiap atribut bukan PK tidak tergantung secara fungsional kepada atribut bukan PK yang lainnya dalam relasi tersebut. Normalisasi tidak diperlukan apabila sudah tidak ada lagi anomali (keganjilan) pada database.

Bagan alir (*Flowchart*)

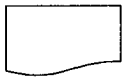
Flowchart adalah suatu teknik analisis yang dipakai untuk menerangkan aspek-aspek suatu sistem informasi secara jelas, ringkas dan menggunakan logika

a. Notasi Baku

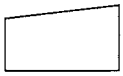
Flowchart digambarkan dengan menggunakan simbol-simbol yang akan memberi kejelasan atas proses/prosedur dan data yang ada dalam suatu sistem. Simbol-simbol *flowchart* dapat dibagi dalam 4 kategori, berikut ini adalah penjelasan dan sebagian dari simbol-simbol tersebut :

- Simbol *Input Output*

  Merupakan simbol bagi data *input* atau *output* dari suatu proses.



Dokumen



Simbol dari entri data dengan *online devices*, misal *CRT terminal* atau *PC*

Gambar 10. Simbol Input-Output

- Simbol Proses

  Menunjukkan media yang digunakan untuk suatu proses data.

Proses komputer (*computer processing*)

Proses manual (*manual processing*)

Gambar 11. Simbol Proses

- Simbol penyimpanan *(storage*)

  Simbol dari media penyimpanan data yang sementara belum dipakai.

Gambar 12. Simbol Penyimpanan

- Simbol Flow & beberapa simbol lain :

  Simbol aliran data, proses, dokumen atau barang. Selain itu menunjukkan kegiatan-kegiatan operasional, misal awal dan akhir suatu flowchart, kapan keputusan diambil, kapan perlu penjelasan tambahan

Gambar 13. Simbol *connector* dan *decision*

b. Beberapa macam Flowchart

- *Flowchart* dokumen

  Flowchart yang menggambarkan arus dokumen dan informasi diantara para pengguna dalam suatu perusahaan/ organisasi. Terlihat pada *flowchart*, antara lain: dari mana asal dokumen, distribusi dokumen, tujuan penggunaan dokumen.

- *Flowchart* sistem komputer

  Flowchart yang menggambarkan kaitan antara *input*, proses dan *output* suatu sistem.

- *Flowchart* program
  Flowchart yang menggambarkan alur (*sequence*) dari logic operasional komputer dalam menjalankan suatu program.
- *Flowchart* konfigurasi komputer
  Flowchart yang menggambarkan perangkat keras (*hardware*) komputer yang dipakai dalam suatu sistem.

c. Perbedaan antara Flowchart dan DFD

Menurut hasil penelitian Kievit & Martin (1989) seperti ditulis dalam buku "*Accounting Information System*" (Romney, Steinbart & Cushing), *flowchart* dan DFD adalah alat (*tools*) yang paling sering dipakai dalam pengembangan dan implementasi suatu sistem. Hasil *survey* mereka menunjukkan bahwa para profesional menggunakan DFD sejumlah 62,5%, sedangkan yang menggunakan *flowchart* sebanyak 97,6%.

Perbedaan-perbedaan keduanya adalah :

a. *Flowchart* lebih menekankan pada aliran dokumen dan *record* / catatan, DFD penekanannya pada aliran data dan keadaan sistem secara keseluruhan

b. *Flowchart* menggambarkan aliran yang lebih bersifat fisik, sedangkan DFD mengambarkan aliran yang bersifat logik.

c. *Flowchart* lebih menekankan kejelasan dari bagaimana data diproses dan disimpan. DFD lebih kepada desain sistem baru dan tidak terlalu mementingkan *physical devices* dalam proses penyimpanan maupun transformasi data.

d. Proses di DFD dapat beroperasi secara paralel, sehingga beberapa proses dapat dilakukan serentak. *Flowchart* cenderung hanya menunjukkan proses yang urut, sedangkan dalam DFD kegiatan proses dapat dilakukan secara tidak urut.

e. DFD tidak menunjukkan proses perulangan (*loop*) dan proses keputusan (*decision*), pada *flowchart* kedua hal tersebut ada.

f. *Flowchart* menggunakan banyak simbol, DFD menggunakan empat simbol.

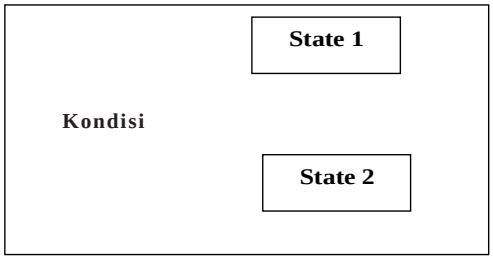
State Transition Diagram (STD) atau Diagram Perubahan Keadaan

State Transition Diagram (STD) adalah teknik pemodelan atas sistem yang bersifat *real time*, misal *telephone switching*. Lebih jelasnya dapat diuraikan *real time system* adalah suatu sistem yang bekerja dengan data misalnya dari sumber diluar sistem, yang harus mengeluarkan respon atau keluaran berkecepatan tinggi dan dapat digunakan kembali oleh lingkungan diluar sistem pada waktu yang relatif sama.

Diagram ini menggunakan notasi persegi panjang sebagai *state* (keadaan), sedangkan panah merepresentasikan perubahan keadaan. Setiap persegi panjang menggambarkan keadaan (*state*) sistem pada saat tertentu, lebih tepatnya jika *state* diasumsikan sebagai kumpulan atribut yang menggambarkan sesuatu pada suatu saat atau kondisi. Contoh dari keadaan (state) antara lain :

a. Menunggu pemakai memasukkan password
b. Pencampuran bahan kimia
c. Menunggu perintah berikutnya
d. Mengisi tangki

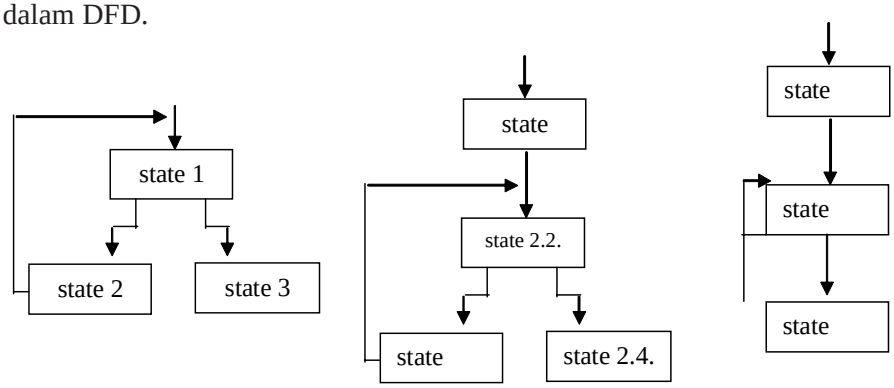
Kelengkapan STD memerlukan dua hal yaitu pertama kondisi penyebab perubahan keadaan, dan kedua aksi yang harus dilakukan ketika akan berubah keadaan.



Gambar 14. Kondisi dan Aksi

Kondisi adalah suatu kejadian pada lingkungan luar yang akan dideteksi sistem, misalnya berupa sinyal atau interupsi atau munculnya sejumlah paket data. Hal ini menyebabkan sistem berubah, dari keadaan menunggu x berubah menjadi keadaan menunggu y, atau melanjutkan pemrosesan x menjadi pemrosesan y. Aksi merupakan respon balik sistem bagi lingkungan di luar sistem atau merupakan bahan masukan bagi keadaan berikutnya.

Pada sistem yang kompleks terdapat lusinan keadaan, sehingga untuk mengelompokkannya dalam suatu diagram tunggal akan sulit dan kadang tidak mungkin. Karena itu digunakan penurunan level, seperti yang dilakukan dalam DFD.



Gambar 15. Penurunan level dalam STD lvl x, lvl x+1, lvl X+2

KESIMPULAN DAN IMPLIKASI

Penulisan ini membahas beberapa perangkat pemodelan, yang bertujuan merepresentasikan sistem melalui diagram, yaitu : Flowchart, *Data Flow Diagram* (DFD), *Entitiy Relationship Diagram* (ERD), dan juga *State Transaction Diagram* (STD).

Pada dasarnya ada tiga alasan yang menyebabkan perlunya melakukan pemodelan sistem, yaitu :

a. Agar dapat memfokuskan perhatian pada hal-hal penting dalam sistem tanpa harus terlibat terlalu jauh

b. Mendiskusikan perubahan dan koreksi terhadap kebutuhan pengguna sistem dengan resiko dan biaya minimal
c. Menguji pengertian penganalisaan sistem terhadap kebutuhan pengguna dan memberi masukan dalam disain sistem dan pemrograman perancangan sistem bila dibutuhkan.

Alat model DFD, ERD, STD, masing-masing memang mempunyai fokus tersendiri pada sistem yang akan dimodelkan. Pada saatnya ada kondisi dimana penganalisa sistem harus menggunakan semua model bersama-sama, untuk melengkapi hasil penelitian yang diharapkan. Namun dalam kasus tertentu penggunaan model-model tersebut dapat mengakibatkan interpretasi yang tidak konsisten dari situasi yang sebenarnya. Hal ini dapat terjadi jika sistem yang dikerjakan sangat besar dimana terlibat personil dan kelompok personil yang mempunyai fokus perhatian yang berbeda. Sehingga penekanan yang perlu ditegaskan disini adalah bahwa model-model yang digunakan harus konsisten satu sama lain, baik itu dalam penggunaan DFD yang difokuskan pada fungsi sistem, atau ERD yang fokusnya pada hubungan antar data, atau STD yang ber fokus pada karekteristik waktu.

DAFTAR PUSTAKA

Arrens, Alvins A dan James K Loebbecke, ***Auditing : An Integrated Approach***, Prentice Hall, Edisi ke-6, 1994

Askelson, Kennenth D., ***Automatic Identification Technologies***, AICPA Info Tec Update, 1994

Bordnar, George H dan Hopwood, William S. ***Accounting Information Systems***, Prentice Hall, Edisi ke-8, 2001

Harris, David. ***System Analysis & Design (for the small enterprise)***, the Dryden Press, Edisi ke-2, 1999

H M Jogiyanto. ***Analisis & Disain Sistem Informasi : Pendekatan Terstruktur Teori & Praktek Aplikasi Bisnis***, Penerbit ANDI Yogyakarta, 1999