PENDEKATAN PERANCANGAN TERSTRUKTUR

Terdapat beberapa pendekatan untuk mengembangkan sistem, yaitu :

1. Pendekatan klasik (***classical approach***) vs pendekatan terstruktur (***structured approach***), dipandang dari **metodologi** yang digunakan.
2. Pendekatan sepotong (***piecemal approach***) vs pendekatan sistem (***system approach***), dipandang dari **sasaran** yang akan dicapai.
3. Pendekatan ***bottom-up*** vs pendekatan ***top-down***, dipandang dari **cara menentukan kebutuhan sistem**.
4. Pendekatan sistem menyeluruh (***total-system approach***) vs pendekatan moduler (***modular approach***), dipandang dari **cara mengembangkan**nya.
5. Pendekatan lompatan jauh (***great loop approach***) vs pendekatan berkembang (***evolutionary approach***), dipandang dari **teknologi** yang akan digunakan.

PENDEKATAN KLASIK

Pendekatan klasik disebut juga pendekatan tradisional atau pendekatan konvensional adalah pendekatan dalam pengembangan sistem yang mengikuti tahapan-tahapan pengembangan sistem (*system life cycle*) tanpa dibekali dengan alat-alat dan teknik-teknik yang memadai. Pendekatan klasik tidak cukup digunakan untuk mengembangkan sistem informasi yang kini semakin kompleks, dan dapat menimbulkan permasalahan, seperti :

1. **Pengembangan perangkat lunak menjadi sulit.**

   Pendekatan klasik tidak menyediakan alat-alat dan teknik-teknik dalam mengembangkan sistem dan sebagai akibatnya proses pengembangan perangkat lunak menjadi tidak terarah dan sulit untuk dikerjakan oleh programer.

2. **Biaya perawatan atau pemeliharaan sistem menjadi lebih mahal.**

   Mahalnya biaya perawatan sistem pada pendekatan klasik adalah karena dokumentasi pengembangan sistem yang kurang lengkap dan kurang terstruktur.

3. **Kemungkinan kesalahan sistem besar.**

   Beberapa penelitian menunjukkan bahwa sistem yang tidak diuji selama tahap pengembangannya merupakan sumber utama dari kesalahan-kesalahan sistem. Pendekatan klasik tidak menyediakan cara untuk menguji sistem, sehingga kemungkinan kesalahan sistem akan menjadi lebih besar.

4. **Keberhasilan sistem kurang terjamin.**

   Penekanan pendekatan klasik adalah kerja dari para personil pengembang sistem bukan pada pemakai sistem. Padahal dukungan dan pemahaman pemakai sistem terhadap sistem yang sedang dikembangkan merupakan hal yang sangat penting dalam keberhasilan proyek pengembangan sistem.

5. **Masalah dalam penerapan sistem.**

   Karena kurangnya keterlibatan pemakai sistem dalam tahapan pengembangan maka pemakai sistem hanya akan mengenal sistem pada saat penerapannya. Dan ini dapat membuat pemakai menjadi kaget dan tidak terbiasa dengan sistem baru dan dapat menimbulkan frustasi bila mereka tidak dapat mengoperasikannya dengan baik.

PENDEKATAN TERSTRUKTUR

Karena terjadi banyak permasalahan pada pendekatan klasik, maka dibutuhkan pendekatan pengembangan sistem yang lebih baik yang tidak hanya mengikuti tahapan *system life cycle* namun juga dilengkapi dengan beberapa alat dan teknik. Pendekatan ini kemudian dikenal dengan pendekatan terstruktur telah dimulai dari awal tahun 1970-an. Pendekatan terstruktur dilengkapi dengan alat-alat (*tools*) dan teknik-teknik (*tehniques*) yang dibutuhkan dalam pengembangan sistem sehingga didapatkan hasil akhir berupa sistem yang strukturnya didefinisikan dengan baik dan jelas.

Beberapa metodologi pengembangan sistem yang terstruktur telah diperkenalkan secara luas. Metodologi pengembangan sistem adalah kesatuan metode-metode, prosedur-prosedur, konsep-konsep pekerjaan, aturan-aturan dan postulat-postulat yang digunakan dalam mengembangkan suatu sistem informasi. Sedangkan metode adalah suatu cara, teknik yang sistematik untuk mengerjakan sesuatu. Sebagian besar metodologi diperuntukkan bagi tahap desain saja, namun banyak juga yang dapat digunakan untuk tahap analisis. Metodologi yang dibahas berikut ini dapat digunakan pada tahap analisis dan disain dan menggunakan pendekatan pengembangan sistem terstruktur. Metodologi-metodologi tersebut dapat diklasifikasikan kedalam tiga kelompok, yaitu :

1. **Metodologi pemecahan fungsional (*functional decomposition methodologies*).**

   Metodologi ini menekankan pada pemecahan sistem ke dalam subsistem-subsistem yang lebih kecil, sehingga akan lebih mudah untuk dipahami, dirancang dan diterapkan. Yang termasuk dalam kelompok ini adalah :

   - HIPO (*Hierarchy plus Input-Process-Output*)
   - *Stepwise refinement* (SR) atau *Iterative stepwise refinement* (ISR)
   - *Information-hiding*

2. **Metodologi berorientasi data (*data-oriented methodologies*).**

   Metodologi ini menekankan pada karakteristik data yang akan diproses. Metodologi ini dikelompokkan kembali ke dalam dua kelas, yaitu :

   a. ***Data-flow oriented methodologies.***

      Metodologi ini secara umum didasarkan pada pemecahan sistem ke dalam modul-modul berdasarkan tipe elemen data dan tingkah laku logika modul tersebut dalam sistem. Dengan metodologi ini, sistem secara logika digambarkan dari arus data dan hubungan antar fungsinya di dalam modul-modul sistem. Yang termasuk dalam metodologi ini adalah :

      - SADT (*Structured Analysis and Design Tehniques*).
      - *Composite design*.
      - *Structured System Analysis and Design* (SSAD).

b. ***Data structured oriented methodologies.***

Metodologi ini menekankan struktur dari input dan output sistem. Struktur ini kemudian akan digunakan sebagai dasar struktur sistemnya. Hubungan fungsi antar modul atau elemen-elemen sistem kemudian dijelaskan dari struktur sistemnya tersebut. Yang termasuk dalam kelompok metodologi ini adalah :

- JSD (Jackson's *System Development*)
- W/O (Warnier / Orr)

3. ***Prescriptive methodologie*s.**

Metodologi ini merupakan metodologi yang dikembangkan oleh *system house* dan pabrik-pabrik perangkat lunak dan tersedia secara komersial dalam paket-paket program. Yang termasuk dalam metodologi ini adalah :

- ISDOS (*Information System Design and Optimization System*)

  ISDOS dikembangkan oleh University of Michigan. Kegunaan ISDOS adalah mengotomatisasi proses pengembangan sistem, dan terdiri dari dua komponen, yaitu :

  - **PSL :** merupakan komponen utama yang berupa suatu bahasa untuk mencatat kebutuhan pemakai dalam bentuk yang dapat dibaca oleh mesin. Outputnya dapat dianalisis oleh PSA. PSL merupakan bahasa untuk menggambarkan sistem dan bukan merupakan bahasa pemrograman prosedural.
  - **PSA :** merupakan paket perangkat lunak yang menyerupai kamus data dan digunakan untuk mengecek data yang dimasukkan, yang disimpan, yang dianalisis, dan yang dihasilkan sebagai output laporan. PSA memanfaatkan DBMS untuk menyimpan data. PSA akan menganalisis PSL untuk kesalahan-kesalahan sintak dan akan menghasilkan sejumlah laporan seperti kamus data (*data dictionary*), kamus fungsi (*function dictionary*), analisis hubungan-hubungan proses, analisis jaringan untuk mengecek kelengkapan semua hubungan data dan proses-proses, analisis

hubungan ketergantungan waktu dari data, dan analisis spesifikasi volume.

- **PLEXSYS :** merupakan komponen tambahan ISDOS, untuk melakukan transformasi statement bahasa tingkat tinggi komputer ke dalam suatu *executable code* untuk suatu konfigurasi perangkat keras yang diinginkan. Jika ISDOS digunakan pada aspek penentuan kebutuhan, PLEXSYS digunakan pada aspek penghasil program otomatis.

- PRIDE

  PRIDE ditawarkan oleh sebuah perusahaan Amerika yaitu M Bryce & Associates. PRIDE merupakan suatu perangkat lunak terpadu yang digunakan untuk analisis dan desain sistem terstruktur, manajemen data, manajemen proyek, dan pendokumentasian. PRIDE juga menyediakan alat CAD (*Computer Aided Design*) untuk pengembangan sistem.

- SDM/70

  SDM/70 (*System Development Methodology*/70) dikembangkan dan dipasarkan oleh suatu perusahaan Amerika, yaitu Atlantic Software, Inc. SDM/70 merupakan suatu perangkat lunak berisi kumpulan metode, estimasi, dokumentasi dan petunjuk administrasi untuk membantu pemakai mengembangkan dan merawat sistem secara efektif.

- SPECTRUM

  Merupakan metodologi pengembangan sistem yang dikembangkan dan dipasarkan oleh sebuah perusahaan Amerika yaitu Spectrum Internasional, Inc. Perangkat lunak ini memiliki beberapa versi untuk keperluan yang berbeda, yaitu SPECTRUM-1 untuk tahapan pengembangan konvensional, SPECTRUM-2 untuk sistem manajemen proyek terstruktur, dan SPECTRUM-3 untuk estimator interaktif online.

- SRES dan SREM

  SRES (*Software Requirement Engineering System*) dikembangkan oleh TRW untuk SDS (*Software Development System*) dari Angkatan udara Amerika Serikat. Pada SRES, kebutuhan pemakai dinyatakan dalam RSL (*Requirement Statement Language*). Definisi RSL kemudian dianalisis menggunakan REVS (*Requirement Engineering and Validation System*) yang juga digunakan untuk memelihara database. Metodologi yang mendasari perangkat lunak ini disebut dengan SREM (*Software Requirement Engineering Methodology*).
- Beberapa *prescriptive methodology* yang lain diantaranya adalah Chapin's approach, DBO (*Design By Objective*), PAD (*Program Analysis Diagram*), HOS (*Higher Order Software*), MSR (*Meta Stepwise Refinement*), dan PDL (*Program Design Language*).

Sedangkan untuk dapat melakukan langkah-langkah sesuai dengan metodologi pengembangan sistem terstruktur, maka dibutuhkan alat-alat dan teknik-teknik untuk melaksanakannya. Alat-alat yang digunakan dalam suatu metodologi umumnya berupa gambar, diagram atau grafik karena lebih mudah dipahami. Namun ada pula alat yang tidak berupa gambar atau grafik. Alat-alat pengembangan sistem berbentuk grafik diantaranya adalah :

a. HIPO diagram, digunakan pada metodologi HIPO dan metodologi lainnya.
b. Data Flow Diagram (DFD), digunakan pada metodologi *Structured System Analysis and Design*.
c. Structured chart, digunakan pada metodologi *Structured System Analysis and Design*.
d. SADT diagram, digunakan pada metodologi SADT.
e. Warnier / Orr diagram, digunakan pada metodologi Warnier / Orr.
f. Jackson's diagram digunakan pada metodologi Jackson's *System Development.*

Disamping alat-alat berbentuk grafik yang digunakan pada suatu metode tertentu, terdapat pula beberapa alat berbentuk grafik yang sifatnya umum dapat

digunakan pada semua metodologi yang ada. Alat-alat ini berupa bagan yang diklasifikasikan sebagai berikut :

1. Bagan untuk menggambarkan aktivitas (*activity charting*)
   a. Bagan alir sistem (*system flowchart*)
   b. Bagan alir program (*program flowchart*)
      - Bagan alir logika progam (*program logic flowchart*)
      - Bagan alir program komputer terinci (*detailed computer program flowchart*)
   c. Bagan alir kertas kerja (*paperwork flowchart*) atau yang disebut juga dengan bagan alir formulir (*form flowchart*).
   d. Bagan alir hubungan database (*database relationship flowchart*)
   e. Bagan alir proses (*process flowchart*)
   f. Gantt chart
2. Bagan untuk menggambarkan tata letak (*layout charting*)
3. Bagan untuk menggambarkan hubungan personil (*personal relationship charting*)
   a. Bagan distribusi kerja (*working distribution chart*)
   b. Bagan organisasi (*organization chart*)

Teknik yang tersedia untuk pengembangan sistem biasanya tidak khusus untuk suatu metodologi tertentu namun dapat digunakan untuk semua metodologi yang ada. Teknik-teknik tersebut adalah :

a. Teknik manajemen proyek, yaitu CPM (*Critical Path Method*) dan PERT (*Program Evaluation and Review Technique*) yang digunakan untuk penjadualan proyek.
b. Teknik menemukan fakta (*fact finding technique*) yang digunakan untuk mengumpulkan data dan fakta-fakta dalam kegiatan mempelajari sistem yang ada. Tehnik-tehnik ini diantaranya adalah :
   - Wawancara
   - Observasi
   - Daftar pertanyaan / kuesioner
   - Pengumpulan sampel (*sampling*)
c. Teknik analisis biaya/manfaat (*cost-effectiveness analysis* atau *cost-benefit analysis*)

d. Teknik untuk menjalankan rapat

e. Teknik inspeksi / *walkthrough*.

PIECEMAL APPROACH VS SYSTEM APPROACH

Piecemal approach merupakan pendekatan pengembangan sistem yang menekankan pada suatu kegiatan atau aplikasi saja. Kegiatan atau aplikasi yang dipilih tersebut, dikembangkan tanpa memperhatikan posisinya di sistem informasi atau tanpa memperhatikan sasaran organisasi secara keseluruhan.

System approach memperhatikan sistem informasi sebagai satu kesatuan terintegrasi dari masing-masing kegiatan atau aplikasinya dan menekankan pada pencapaian sasaran keseluruhan.

BOTTOM-UP APPROACH VS TOP-DOWN APPROACH

Pendekatan ***bottom-up*** dimulai dari level bawah organisasi, yaitu level operasional tempat transaksi dilakukan. Pendekatan ini dimulai dari perumusan kebutuhan-kebutuhan untuk menangani transaksi dan naik ke level atas dengan merumuskan kebutuhan informasi berdasarkan transaksi tersebut. Pendekatan ini merupakan ciri-ciri pendekatan klasik. Jika pendekatan ini digunakan pada tahap analisis, disebut dengan ***data analysis***, karena yang menjadi fokus adalah data yang akan diolah terlebih dahulu.

Sedangkan pendekatan ***top-down*** sebaliknya dimulai dari level atas organisasi yaitu level perencanaan strategis. Pendekatan ini dimulai dengen mendefinisikan sasaran dan kebijakan organisasi. Selanjutnya, dilakukan analisis kebutuhan informasi kemudian ke penentuan input, output, basis data, prosedur-prosedur operasi, dan kontrol. Pendekatan ini merupakan ciri-ciri dari pendekatan terstruktur. Jika pendekatan ini digunakan pada tahap analisis, disebut dengan ***decision analysis***, karena yang menjadi fokus adalah informasi yang dibutuhkan untuk pengambilan keputusan oleh manajemen terlebih dahulu.

TOTAL-SYSTEM APPROACH VS MODULAR APPROACH

Total-system approach merupakan pendekatan pengembangan sistem serentak secara menyeluruh. Pendekatan ini sulit dilakukan untuk sistem yang kompleks, karena menjadi sulit untuk dikembangkan.

Modular approach berusaha memecah sistem yang rumit menjadi beberapa bagian atau modul yang sederhana sehingga akan lebih mudah dipahami dan dikembangkan. Sistem juga akan dapat dikembangkan sesuai dengan waktu yang direncanakan, mudah dipahami oleh pemakai dan mudah untuk dipelihara.

GREAT LOOP APPROACH VS EVOLUTIONARY APPROACH

Great loop approach menerapkan perubahan menyeluruh secara serentak menggunakan teknologi canggih. Hal ini mengandung resiko karena teknologi komputer begitu cepat berkembang dan tahun-tahun mendatang sudah menjadi usang, investasinya juga mahal dan terlalu kompleks.

Evolutionary approach menerapkan teknologi canggih hanya untuk aplikasi yang memerlukan saja saat itu dan akan terus dikembangkan untuk masa-masa selanjutnya mengikuti kebutuhan dan sesuai dengan perkembangan teknologi yang ada.

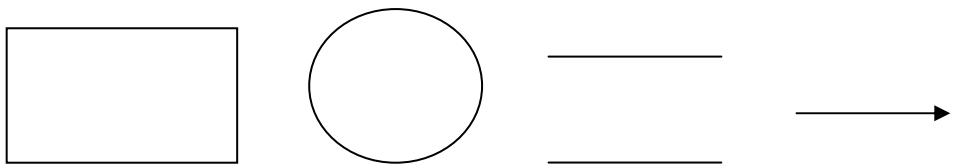
DATA FLOW DIAGRAM

Penggunaan bagan dan notasi untuk mewakili arus data dalam suatu sistem telah banyak dilakukan. Penggunaan notasi dalam diagram arus data ini sangat membantu dalam memahami suatu sistem pada semua tingkat kompleksitasnya. Pada tahap analisis, notasi ini membantu dalam komunikasi dengan pemakai sistem untuk memahahi sistem secara logika. Diagram ini dikenal dengan diagam arus data (*Data Flow Diagram* = DFD).

DFD merupakan alat pengembangan sistem yang berorientasi pada alur data dengan konsep dekomposisi yang dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program. DFD dapat digunakan untuk menggambarkan sistem yang telah ada maupun sistem baru secara logika tanpa mempertimbangkan lingkungan fisik dari data yang mengalir (misalnya lewat telepon, surat, dan sebagainya) maupun tempat data tersebut disimpan (misalnya file, kartu, hardisk, tape, disket, dan sebagainya).

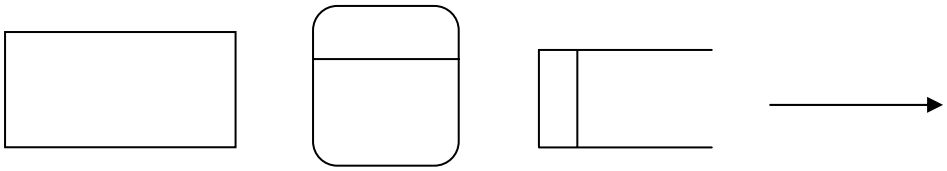
KOMPONEN DFD

Menurut Yourdan dan DeMarco



Terminator Proses Data Store Alur Data

Menurut Gene dan Sarson



Terminator Proses Data Store Alur Data

❶ **Terminator / Entitas Luar (*External Entity*) / Batas Sistem (*Boundary*)**

Terminator adalah entitas di luar sistem yang berkomunikasi / berhubungan langsung dengan sistem. Entitas luar ini dapat berupa orang, sekelompok orang, organisasi, perusahaan, departemen atau sistem lainnya yang berada di lingkungan luar sistem yang akan memberikan input atau menerima output dari sistem.

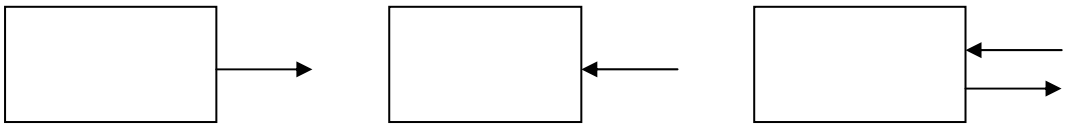
Terdapat 2 jenis terminator :

1. **Terminator sumber**

   Merupakan Terminator yang menjadi sumber

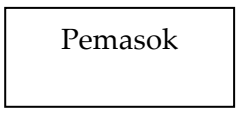
2. **Terminator Tujuan**

   Merupakan terminator yang menjadi tujuan data / informasi sistem.



terminator sumber | terminator tujuan | terminator tujuan dan sumber

Sebagai identifikasi, terminator diberi nama dan biasanya menggunakan kata benda. Contoh : Dosen, Mahasiswa, Pemasok, Langganan, dan sebagainya.



Hal yang perlu diperhatikan tentang terminator :

1. Alur data yang menghubungkan terminator dengan sistem, menunjukkan hubungan sistem dengan dunia luar.
2. Profesional sistem tidak dapat mengubah isi / cara kerja, prosedur yang berkaitan dgn terminator.
3. Hubungan yang ada antar terminator tidak digambarkan dalam DFD.

❷ Proses

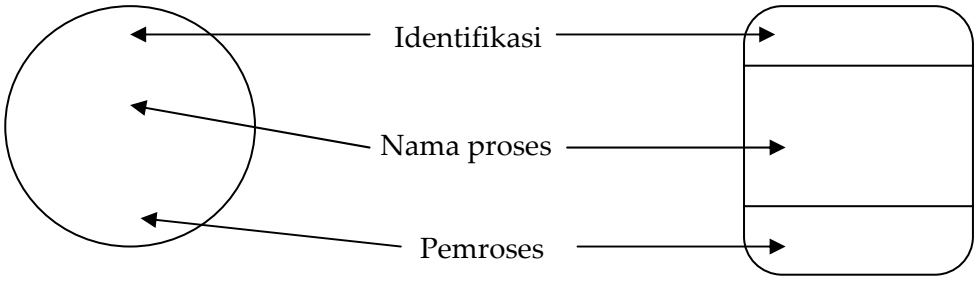
Komponen proses menggambarkan kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari suatu arus data yang masuk ke dalam proses (input) untuk menghasilkan arus data yang keluar dari proses (output). Untuk *physical data flow diagram* (PDFD), proses dapat dilakukan oleh orang, mesin atau komputer sedangkan untuk *logical data flow diagram* (LDFD), suatu proses hanya menunjukkan proses dari komputer.

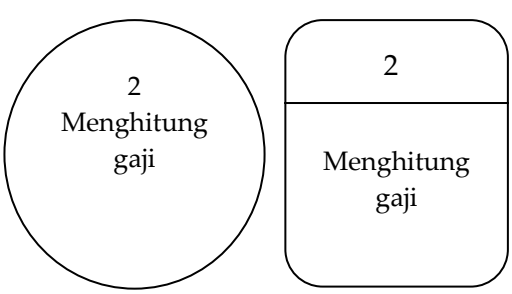
Setiap proses harus diberikan penjelasan lengkap yang meliputi :

1). Identifikasi proses yang umumnya berupa angka pada bagian atas simbol proses.

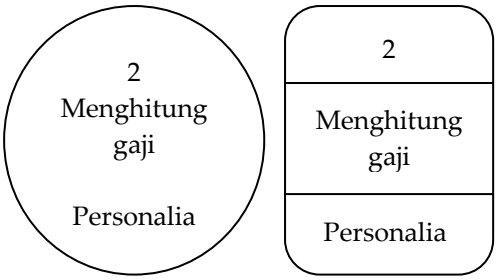
2). Nama proses yang menunjukkan kegiatan apa yang sedang dikerjakan oleh proses tersebut.

3). Pemroses, pada PDFD, proses dapat dilakukan oleh komputer maupun manual seperti oleh orang, mesin, dan sebagainya sehingga perlu ditunjukkan pemrosesnya. Sedangkan pada LDFD, proses hanya dilakukan oleh komputer sehingga tidak perlu disebutkan pemrosesnya.



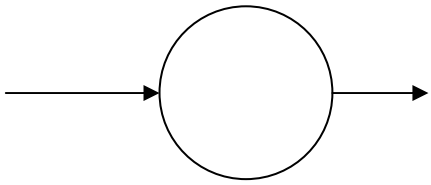


Komponen proses LDFD

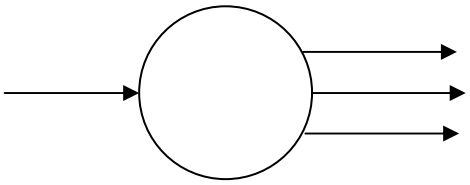


Komponen proses PDFD

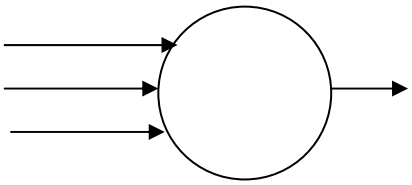
Ada 4 kemungkinan yang dapat terjadi dalam proses sehubungan dengan input dan output :



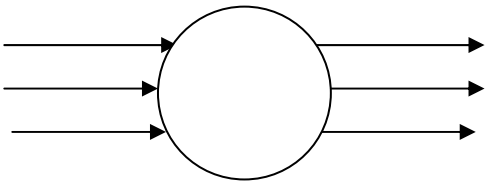
1 input dan 1 output



1 input dan banyak output



Banyak input dan 1 output



Banyak input dan banyak output

Ada beberapa hal yang perlu diperhatikan tentang proses :

1. Proses harus memiliki input dan output.
2. Proses dapat dihubungkan dengan komponen terminator, simpanan data atau proses melalui alur data.
3. Sistem/bagian/divisi/departemen yang sedang dianalisis oleh profesional sistem digambarkan dengan komponen proses.

❸ *Data Store* / Simpanan Data

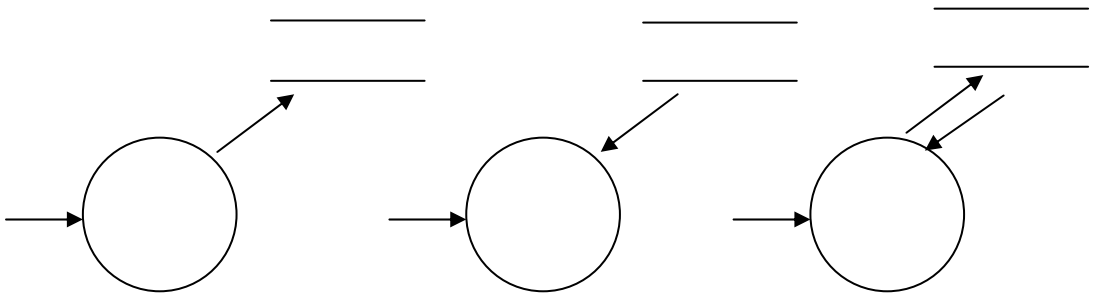
Komponen ini digunakan untuk membuat model sekumpulan paket data. Simpanan data dapat berupa file atau database yang tersimpan dalam disket, harddisk atau bersifat manual seperti arsip / catatan manual, agenda / buku, kotak tempat data / file folder. Komponen simpanan data diberi nama dengan kata benda. Untuk PDFD, selain nama simpanan data perlu dicantumkan penjelasan mengenai media dari simpana data tersebut. Sedangkan pada LDFD, cukup identifikasi dan namanya saja.

D1	Penjualan

Yang perlu diperhatikan tentang simpanan data:

1. Hanya proses saja yang berhubungan dengan simpanan data, karena yang menggunakan atau mengubah data pada simpanan data adalah suatu proses.
2. Alur data dari proses menuju simpanan data, hal ini berarti simpanan data berfungsi sebagai tujuan / tempat penyimpanan dari suatu proses (proses *write*).
3. Alur data dari simpanan data ke proses, hal ini berarti simpanan data berfungsi sebagai sumber/ proses memerlukan data (proses *read*).
4. Alur data dari proses menuju simpanan data dan sebaliknya berarti berfungsi sebagai sumber dan tujuan.

Lihat gambar berikut :



Proses Write Proses Read Proses Update

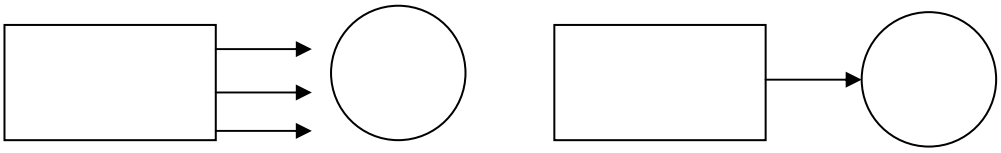
❹ Alur Data / Data Flow

Alur data digunakan untuk menerangkan perpindahan data / paket data yang terjadi di antara proses, simpanan data dan terminator. Alur data dapat berupa kata, pesan, formulir / dokumen, laporan, informasi, surat / memo, dan sebagainya. Alur data diberi nama yang jelas dan mempunyai arti.

Ada 4 konsep tentang alur data :

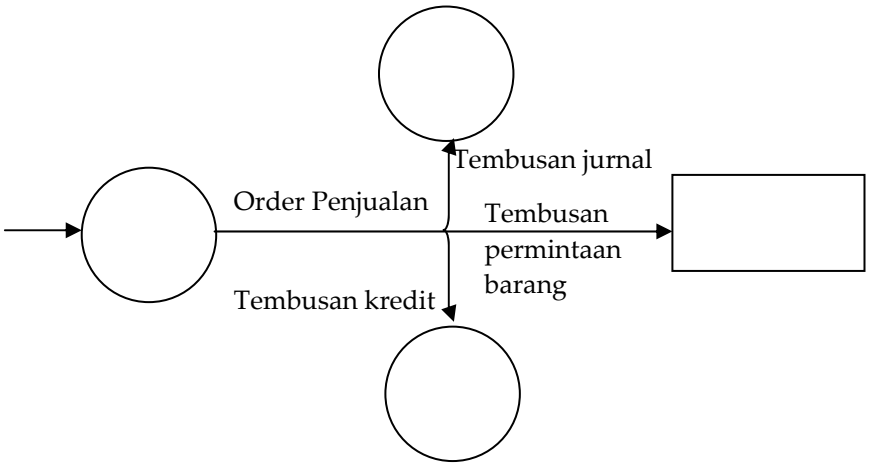
1. Konsep paket data (Packets of data)

Apabila ada 2 data atau lebih yg mengalir dari 1 sumber yang sama menuju pada tujuan yang sama dan mempunyai hubungan digambarkan dengan 1 alur data.



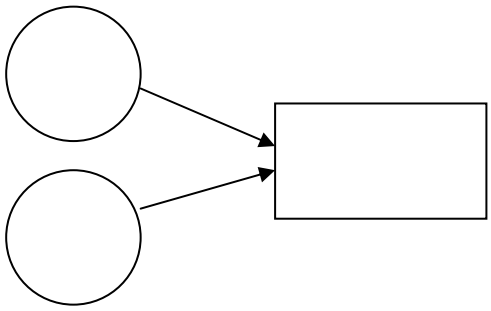
2. Konsep arus data menyebar (Diverging data flow)

Apabila ada sejumlah paket data yang berasal dari sumber yang sama menuju pada tujuan yang berbeda atau paket data yang kompleks dibagi menjadi beberapa elemen data yang dikirim ke tujuan yang berbeda.



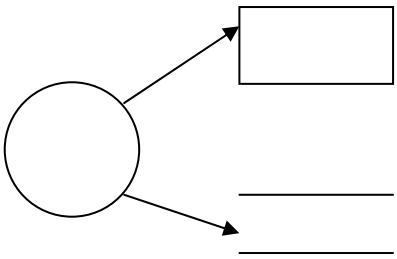
3. Konsep arus data mengumpul (Converging data flow)

Apabila ada beberapa alur data yang berbeda sumber menuju ke tujuan yang sama.

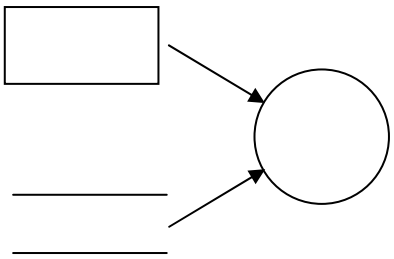


4. Sumber dan Tujuan

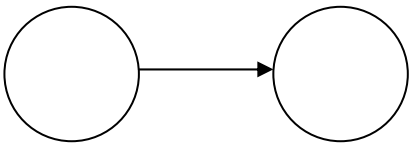
Semua arus data harus dihubungkan dengan proses, baik yang dihasilkan maupun yang menuju ke suatu proses.



Dari proses ke bukan proses



Dari bukan proses menuju proses



Dari proses ke proses

LEVELISASI DFD

Diagram Konteks

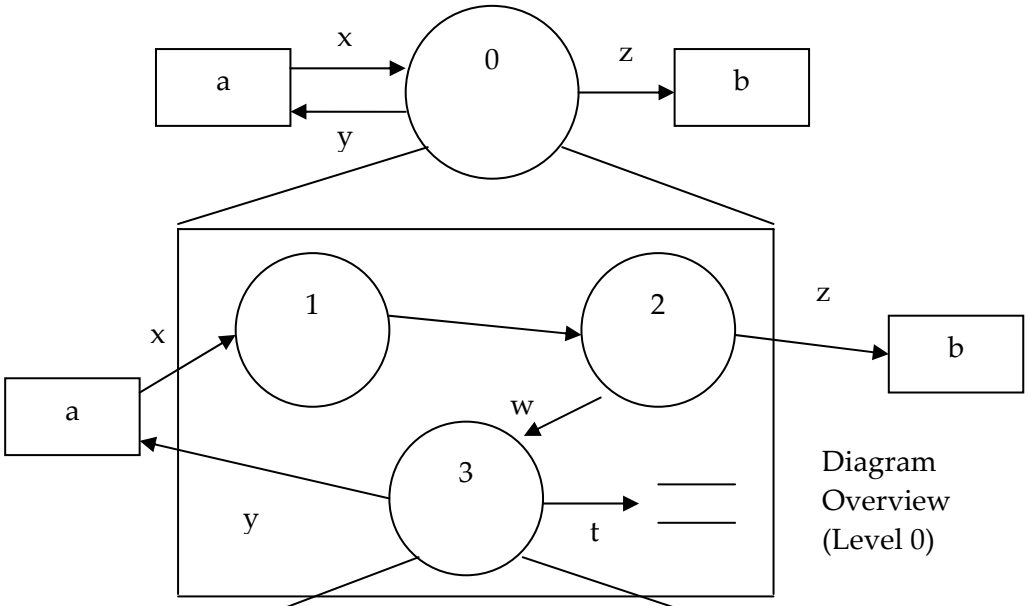


Diagram Overview (Level 0)

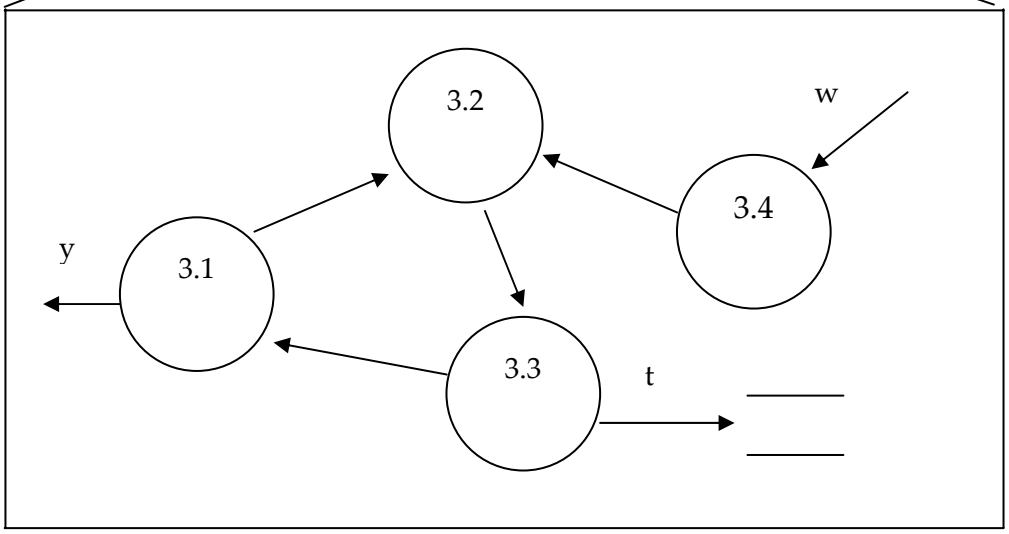


Diagram Level 1

BENTUK DFD

Terdapat 2 bentuk DFD, yaitu DFD fisik (*physical data flow diagram* = PDFD) dan DFD logika (*logical data flow diagram* = LDFD). DFD fisik lebih menekankan pada bagaimana proses sistem diterapkan sedangkan DFD logika lebih menekankan pada proses-proses apa yang terdapat pada sistem.

1. DFD Fisik (*physical data flow diagram* = PDFD)

DFD fisik lebih tepat digunakan untuk menggambarkan sistem yang sudah ada (sistem lama). Penekanan DFD fisik adalah bagaimana proses-proses sistem diterapkan (dengan cara apa, oleh siapa, dan di mana), termasuk proses-proses manual. Dengan DFD fisik, bagaimana proses sistem berajalan dapat lebih digambarkan dan dikomunikasikan kepada pemakai sistem, sehingga analis sistem dapat memperoleh gambaran yang jelas bagaimana sistem tersebut bekerja.

DFD fisik harus memuat :

- Proses-proses manual
- Nama arus data harus menunjukkan penerapannya seperti nomor formulir dan medianya, waktu mengalirnya, dsb. Dengan kata lain, nama arus data harus memuat keterangan yanng cukup terinci untuk menunjukkan bagaimana pemakai sistem memahami kerja sistem.
- Simpanan data dapat menunjukkan simpanan non komputer.
- Nama dari simpanan data harus menunjukkan tipe penerapannya apakah manual atau terkomputerisasi.
- Proses harus menunjukkan nama pemroses, yaitu orang, departemen, sistem komputer, atau nama program komputer yang mengeksekusi proses tersebut.

2. DFD Logika (*logical data flow diagram* = PDFD)

DFD logika lebih tepat digunakan untuk menggambarkan sistem yang akan diusulkan (sistem baru). DFD logika menekankan hanya pada logika dari kebutuhan-kebutuhan sistem, yaitu proses-proses apa secara logika yang dibutuhkan oleh sistem. Karena sistem yang diusulkan belum tentu diterima dan

biasanya terdiri dari beberapa alternatif, maka penggambaran secara logika terlebih dahulu tanpa melihat penerapannya secara fisik akan lebih mengena dan menghemat waktu. Untuk sistem komputerisasi, penggambaran DFD logika hanya akan menunjukkan kebutuhan proses sistem, dan umumnya yang digambarkan hanya proses-proses secara komputer saja.

PENGGAMBARAN DFD

Tidak ada aturan baku untuk menggambarkan DFD, tapi dari berbagai referensi yang ada, secara garis besar :

1. Buat diagram konteks

   Diagram ini adalah diagram level tertinggi dari DFD yg menggambarkan hubungan sistem dengan lingkungan luarnya.

   Cara :

   - Tentukan nama sistemnya.
   - Tentukan batasan sistemnya.
   - Tentukan terminator apa saja yang ada dalam sistem.
   - Tentukan apa yang diterima / diberikan terminator dari / pada sistem.
   - Gambarkan diagram konteks.

2. Buat diagram level zero

   Diagram ini adalah dekomposisi dari diagram konteks

   Cara :

   - Tentukan proses utama yang ada pada sistem.
   - Tentukan apa yang diberikan / diterima masing-masing proses pada / dari sistem sambil memperhatikan konsep keseimbangan (alur data yang keluar / masuk dari suatu level harus sama dengan alur data yang masuk / keluar pada level berikutnya)
   - Apabila diperlukan, munculkan simpanan data (master) sebagai sumber maupun tujuan alur data.

- Gambarkan diagram level zero.
  - Hindari perpotongan arus data dengan membuat duplikat dari simpanan data atau terminator. Duplikasi pada terminator dapat disimbolkan dengan garis miring (/) atau asteriks (*). Sedangkan pada simpanan data dapat digunakan asteriks (*) atau garis vertikal (|).
  - Beri nomor pada proses utama (nomor tidak menunjukkan urutan proses).

3. Buat diagram level Satu

   Diagram ini merupakan dekomposisi dari diagram level zero.

   Cara :

   - Tentukan proses yang lebih kecil (sub-proses) dari proses utama yang ada di level zero.
   - Tentukan apa yang diberikan / diterima masing-masing sub-proses pada / dari sistem dan perhatikan konsep keseimbangan.
   - Apabila diperlukan, munculkan simpanan data (transaksi) sebagai sumber maupun tujuan alur data.
   - Gambarkan DFD level Satu
     - Hindari perpotongan arus data dengan membuat duplikat dari simpanan data atau terminator..
     - Beri nomor pada masing-masing sub-proses yang menunjukkan dekomposisi dari proses sebelumnya. Contoh : 1.1, 1.2, 2.1

4. DFD level dua, tiga, ..

   Diagram ini merupakan dekomposisi dari level sebelumnya. Proses dekomposisi dilakukan sampai dengan proses siap dituangkan ke dalam program. Aturan yang digunakan sama dengan level satu.