DASAR – DASAR PERANCANGAN PERANGKAT LUNAK

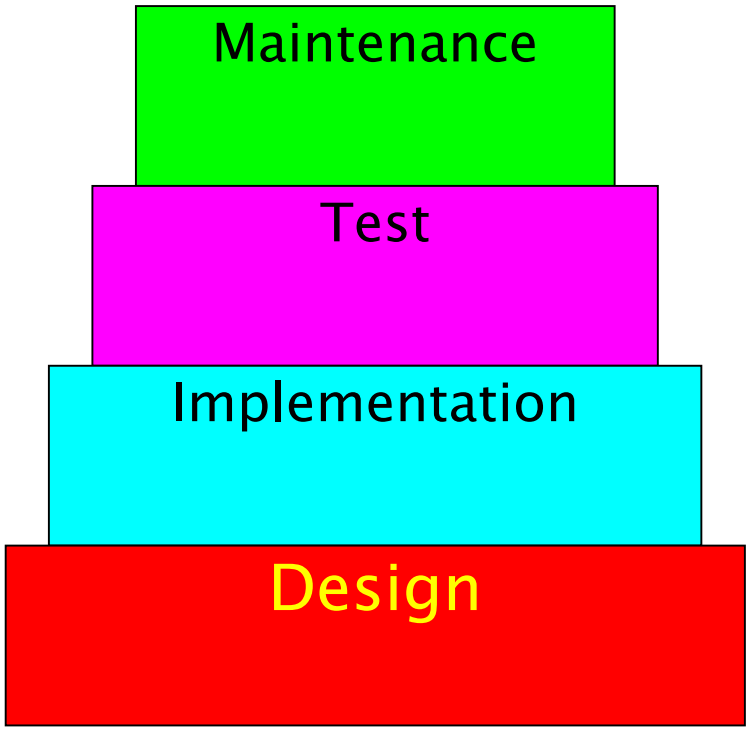
UNIVERSITAS INDRAPRASTA PGRI

PENGANTAR

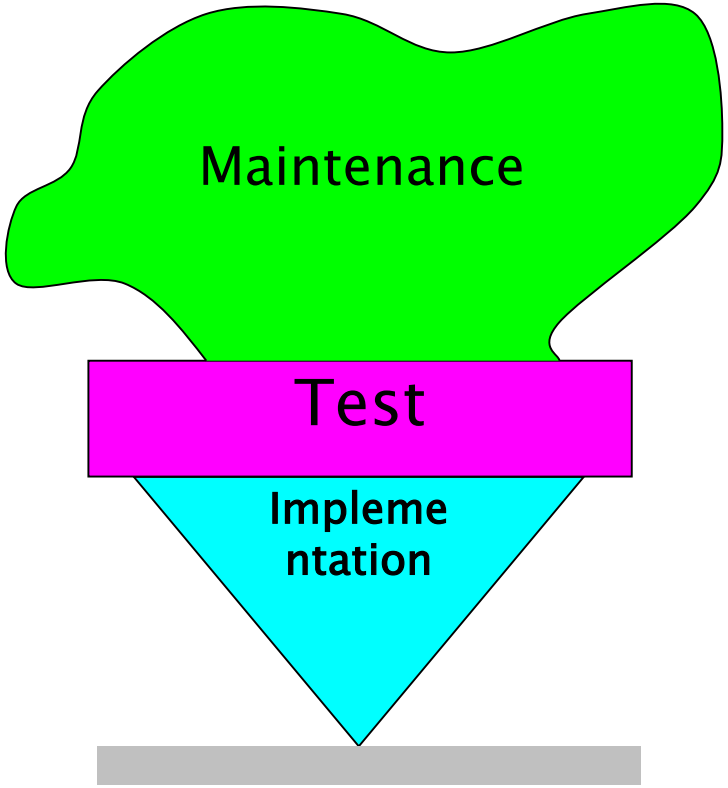
- Perancangan Perangkat Lunak merupakan proses penerjemahan dari kebutuhan menjadi perangkat lunak

- Hasil dari perancangan adalah :
  - Rancangan data yang memetakan model domain informasi pada saat analisis menjadi struktur data yang dibutuhkan untuk implementasi perangkat lunak.
  - Rancangan arsitektural yang mendefinisikan hubungan dari komponen-komponen struktural utama dari program.
  - Rancangan prosedural yang memetakan komponen-komponen struktural ke deskripsi prosedur perangkat lunak

PENGANTAR



Dengan perancangan



Tanpa perancangan

PRINSIP-PRINSIP DASAR PERANCANGAN PERANGKAT LUNAK

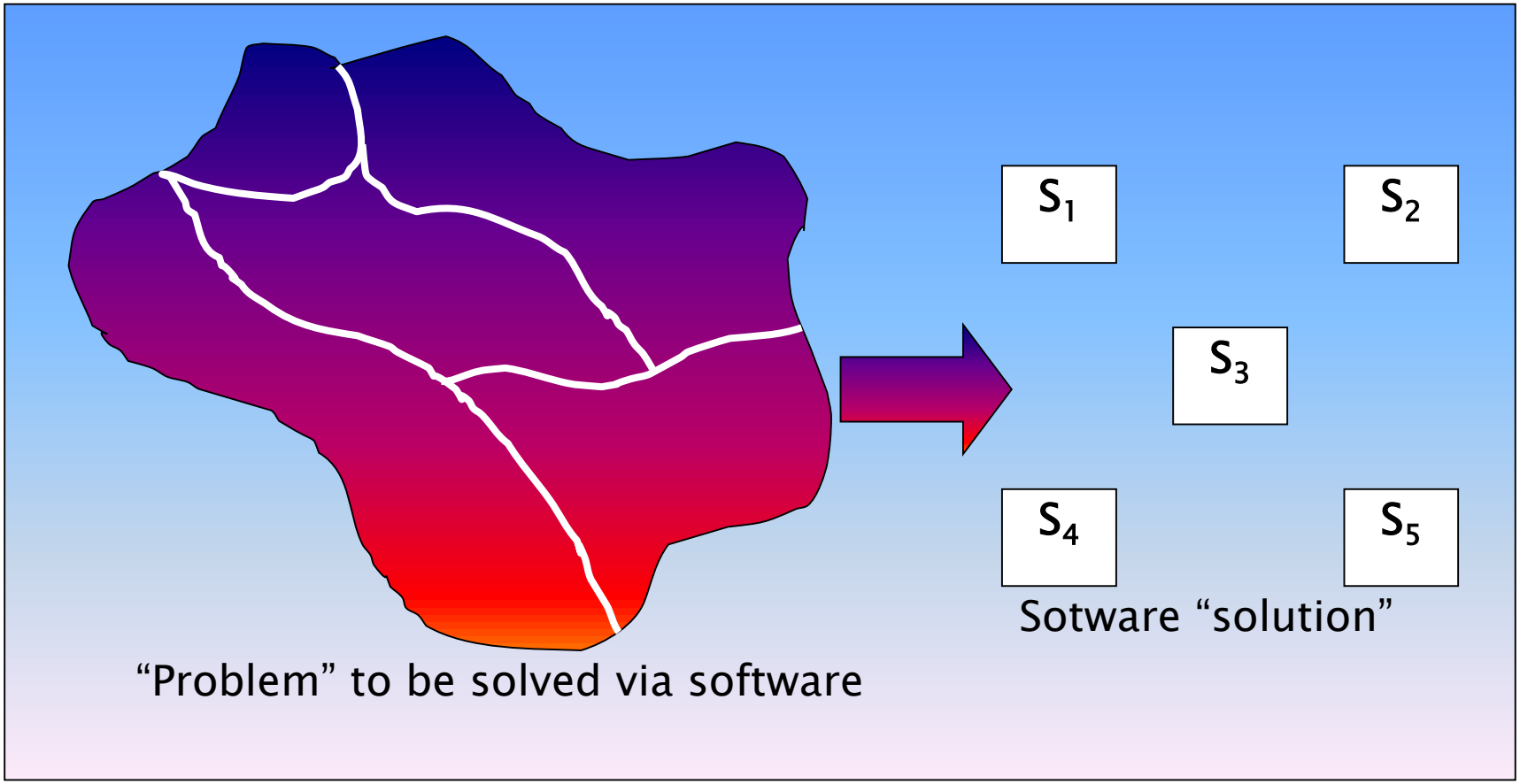
1. ABSTRACTION (Wasserman)
   - Pada rancangan secara modular, beberapa tingkatan abstraksi dapat diperoleh, sehingga perancang dapat mengkonsentrasikan pada setiap tingkatan abstraksi yang lebih terinci.
   - Pada level paling tinggi, solusi dinyatakan secara global dengan bahasa pada lingkungan masalah. Dan pada abstraksi paling bawah, solusi dinyatakan dalam bahasa yang dapat langsung diimplementasikan

PRINSIP-PRINSIP DASAR PERANCANGAN PERANGKAT LUNAK

2. MODULARITY & SOFTWARE ARCHITECTURE

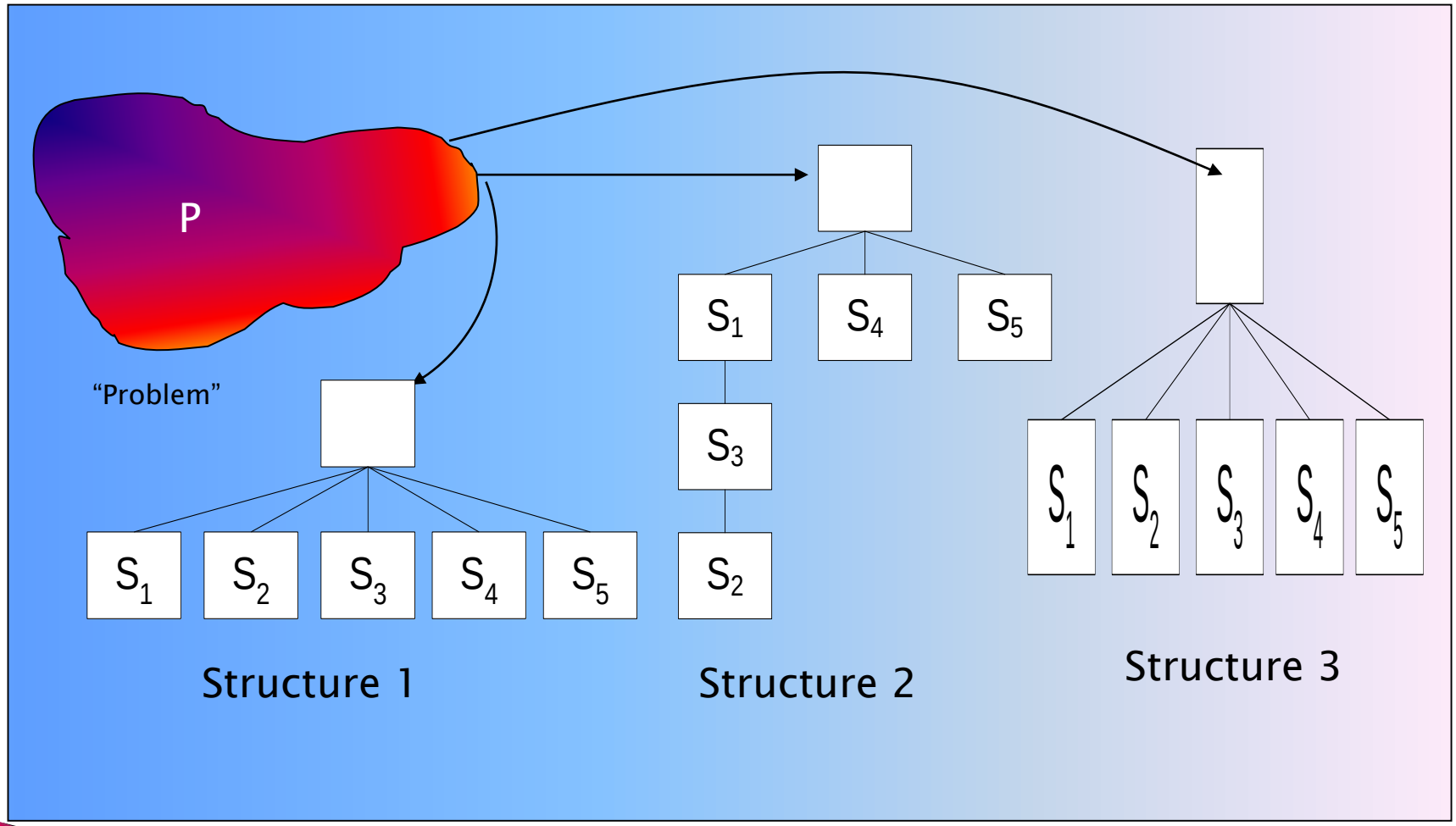
- Perangkat lunak dibagi atas beberapa modul.
- Sebuah modul dapat dibagi lagi atas beberapa sub-modul
- Modul memiliki nama yang unik.
- Sebuah modul dapat memanggil modul lainya.

PRINSIP-PRINSIP DASAR PERANCANGAN PERANGKAT LUNAK



Struktur Evolusi

PRINSIP-PRINSIP DASAR PERANCANGAN PERANGKAT LUNAK



Struktur Berbeda

PRINSIP-PRINSIP DASAR PERANCANGAN PERANGKAT LUNAK

- **HIRARKI KONTROL (STRUKTUR PROGRAM)**
  - Menunjukkan organisasi dari modul-modul program dan menunjukkan hirarki kontrolnya. Tidak merepresentasikan aspek prosedural dari perangkat lunak seperti urutan proses, keputusan, atau perulangan.
  - Kedalaman dan lebar menunjukkan jumlah tingkatan kontrol dan seluruh cakupan kontrol
  - *Fan-out* menunjukkan jumlah modul yang secara langsung dikontrol oleh modul lain
  - *Fan-in* menunjukkan jumlah modul yang mengontrol modul yang bersangkutan
  - Modul yang mengontrol modul yang lain disebut ***superordinate***
  - Modul yang dikontrol modul yang lain disebut ***subordinate***

FAN-OUT

- Fan-out dari sebuah modul adalah banyaknya subordinate langsung dari modul tersebut
- Perluasan kontrol dari sebuah modul sebaiknya tidak melebihi 7 + 2 (kecuali pada pusat-pusat transaksi)
- Hindarkan Fan-out yang bersifat *main-line* (satu boss, dengan modul-modul lain sebagai subordinate)
- Sebuah modul dengan Fan-out yang banyak biasanya sukar dipelihara.
- Untuk memecahkan fan-out yang banyak gunakan modul-modul antara

FAN-IN

- Fan-in dari modul adalah banyaknya modul lain yang (boss) menggunakan/memanggil modul tersebut.
- Jika mungkin Fan-in harus dilakukan sebanyak-banyaknya.
- Fan-in yang banyak menghindari pengulangan pembuatan modul yang sama atau serupa
- Fan-in yang banyak mempermudah pemeliharaan karena menempatkan suatu fungsi yang sama dalam satu modul

PRINSIP-PRINSIP DASAR PERANCANGAN PERANGKAT LUNAK

- **STRUKTUR DATA**
  - Representasi logika dari hubungan antara elemen-elemen data.

- **PROSEDUR PERANGKAT LUNAK**
  - Struktur program hanya mendefinisikan hirarki kontrol tanpa memperhatikan urutan proses. Prosedur perangkat lunak berfokus pada rincian proses dari setiap modul.

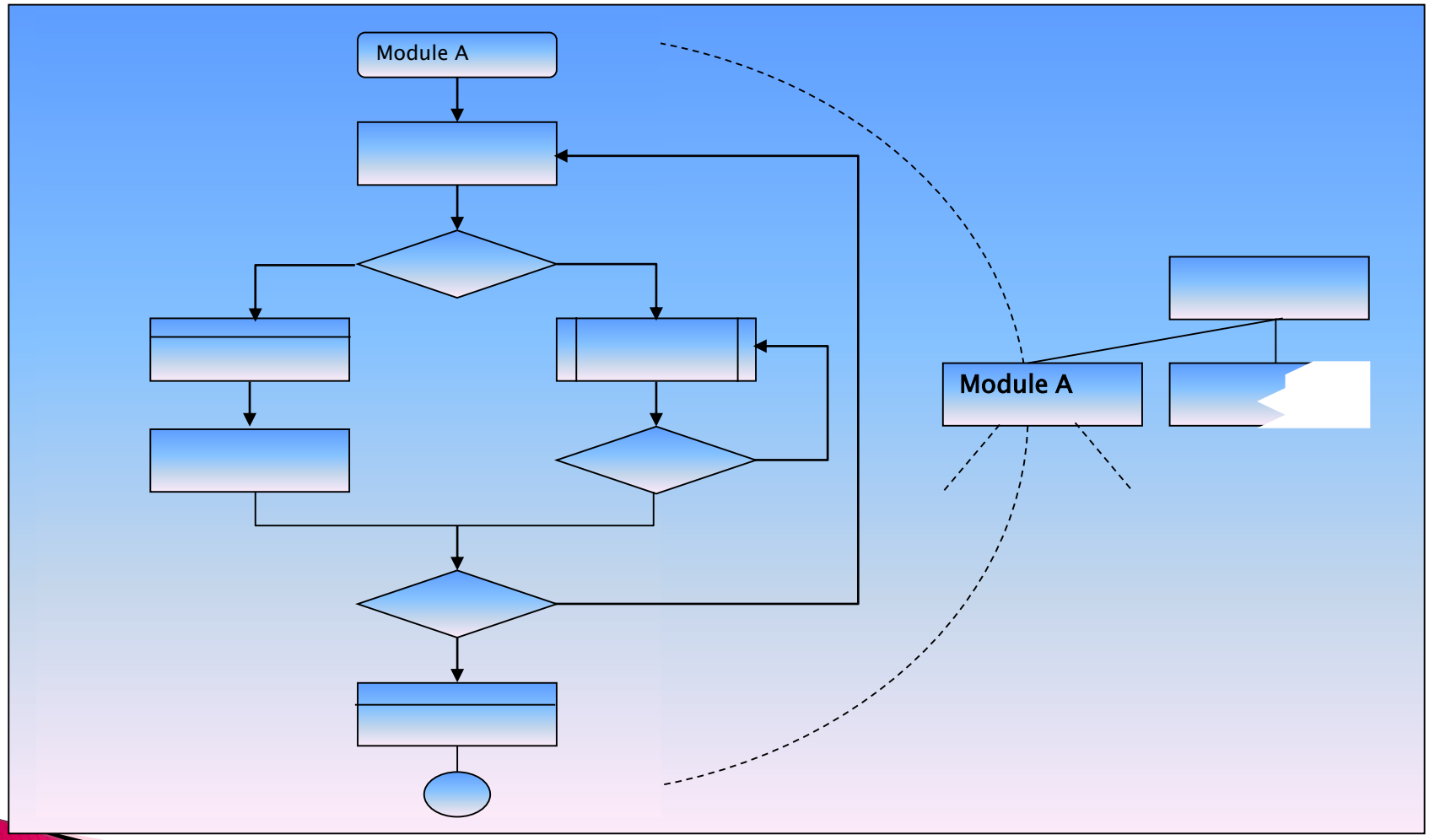
- **INFORMATION HIDING (by Pamas)**
  - Prinsip dasar dalam pembentukan modul dimana hanya data yang benar-benar perlu, yang dikenalkan dan dapat diakses oleh sebuah modul.

PERANCANGAN YANG MODULAR

- Keuntungan :
  - menurunkan kompleksitas
  - mempermudah pengubahan
  - implementasi yang lebih mudah karena bagian-bagian yang berbeda dapat dibuat dengan paralel

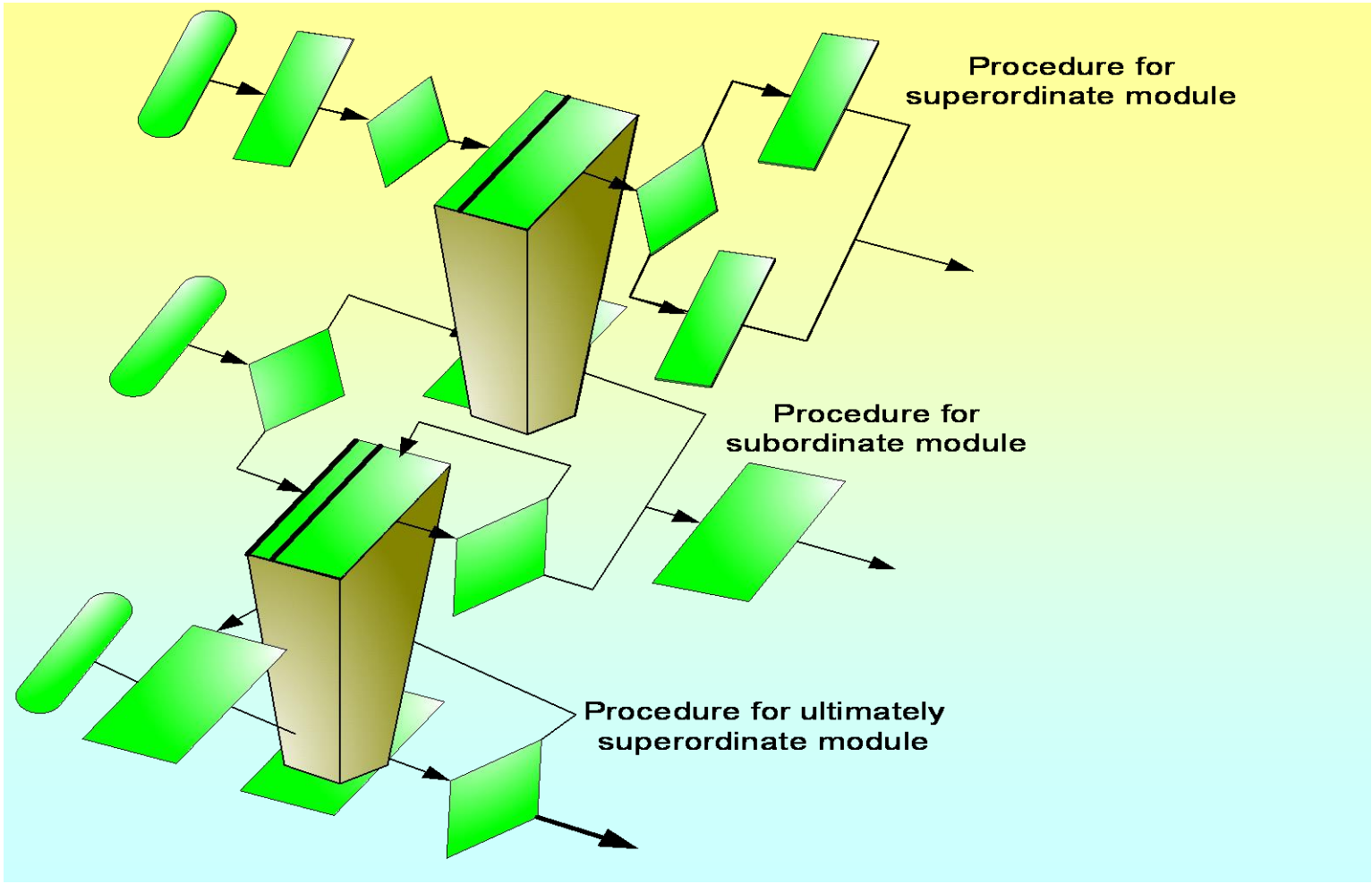
- Evaluasi dari hubungan antar modul dapat dinilai dengan melihat *kohesi* dan *kopling*nya (Steven, Myers, dan Constantine)

PERANCANGAN YANG MODULAR



Prosedur dalam suatu modul

PERANCANGAN YANG MODULAR



Prosedur berlapis

KOPLING

- Kopling adalah tingkat saling ketergantungan antara dua modul.
- Kita menghendaki modul dengan kopling rendah yaitu modul-modul yang sedapat mungkin tidak saling bergantungan.
- Kopling yang rendah adalah tanda dari pembagian sistem yang baik, dimana sesuatu yang tidak berhubungan dipisahkan.
- Jika sedikit atau tidak ada interaksi dua modul disebut ***loosely coupled*** (kopling rendah) dan jika sebaliknya disebut ***tighty coupled*** (kopling tinggi)
- Makin tinggi kopling yang ada makin sulit sebuah program untuk dimengerti.
- Jika dua modul memiliki kopling yang rendah, maka sebuah modul dapat diubah tanpa perlu mengubah modul yang lain.

KOPLING

- Mengapa kopling yang rendah diperlukan ?
  - Untuk menghilangkan ***ripple effect*** (perubahan pada sebuah modul dapat berpengaruh pada modul lain). Sehingga dapat memelihara atau mengubah suatu modul dengan resiko yang minimal untuk mengubah modul lainnya. Jika mungkin kita ingin dapat bekerja dengan modul A tanpa perlu mengetahui tentang apapun dalam modul B.

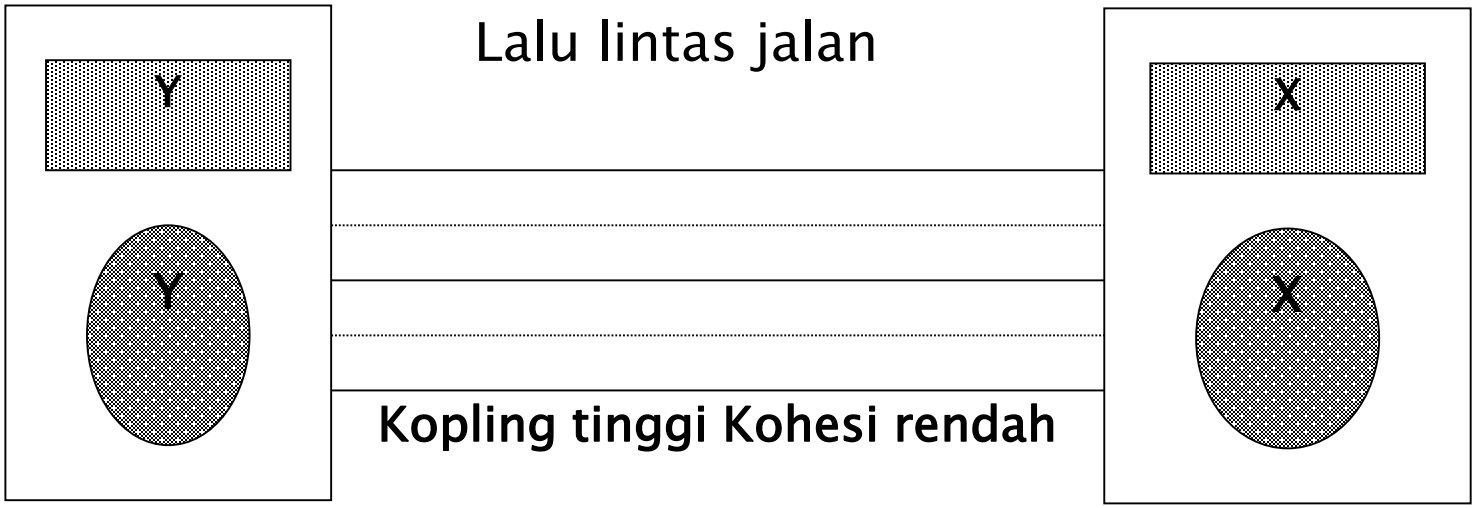
KOPLING

- Faktor-faktor yang berpengaruh pada kopling antara dua modul adalah :

  - Jumlah item data yang disalurkan diantara dua modul (*makin banyak data yg disalurkan makin tinggi kopling yang terjadi*).
  - Jumlah data kontrol yang disalurkan diantara dua modul (*makin banyak data kontrol yang disalurkan makin tinggi kopling yang terjadi*)
  - Jumlah elemen data global yang digunakan bersama-sama oleh beberapa modul (*makin banyak data global yang digunakan, makin tinggi kopling yang terjadi*)

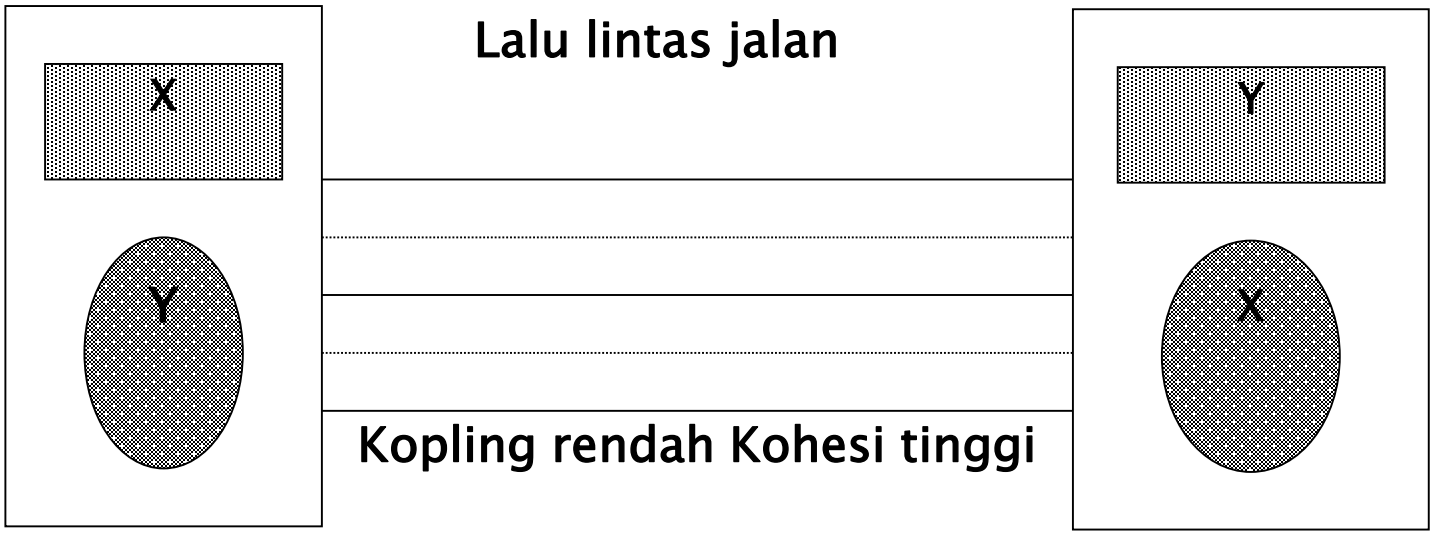
KOHESI

- Apa yang terjadi diantara modul-modul (kopling) dipengaruhi oleh apa yang terjadi dalam modul-modul tersebut secara individual (kohesi)



KOHESI

- Melekatkan hal-hal yang berkaitan didalam modul yang sama, akan mengurangi lalu-lintas diantara modul-modul



KOHESI

- Kohesi adalah ukuran kekuatan asosiasi antar elemen didalam suatu modul.
- Elemen yang dimaksud adalah : sebuah instruksi, sekumpulan intruksi, atau pemanggilan ke modul lain.
- Kohesi tinggi jika sebuah modul hanya bertanggung jawab terhadap satu pekerjaan saja.

RANCANGAN DATA

- Rancangan data yang bagus dapat membuat struktur program lebih baik, modularitas efektif dan menurunkan kompleksitas prosedural

- Beberapa petunjuk :
  - Semua struktur data dan operasi yang mengolahnya harus didefinisikan
  - Selalu gunakan kamus data
  - Rancanagan data yang bersifat low-level harus ditunda sampai akhir dari proses perancangan
  - Bentuk keputusan dan struktur data serta operasi-operasi yang mengolahnya
  - Bahasa pemrograman harus mendukung spesifikasi dan realisasi dari tipe data abstrak.

PERANCANGAN ARSITEKTURAL

- Tujuan dari perancangan arsitektural adalah untuk membangun struktur program yang modular dan membentuk hubungan kontrol antar modul

- Rancangan arsitektural menggabungkan struktur program dan struktur data serta mendefinisikan *interface* yang membuat data melalui program. Dan dinyatakan dalam bentuk Bagan Susunan atau diagram Wamier/Orr.

BAGAN SUSUNAN (BAGAN TERSTRUKTUR)

- Bagan susunan merupakan susunan hirarki dari modul-modul

- Bagan susunan menunjukkan
  - pembagian sistem menjadi modul-modul
  - hirarki dan organisasi modul-modul
  - komunikasi antar modul (masukan dan keluaran)
  - nama modul, yang berarti juga fungsi modul.

- Bagan Susunan tidak menunjukkan :
  - Mekanik didalam modul (seperti ukuran pemanggilan modul lain, *loops* dsb
  - Data internal dari modul

BAGAN SUSUNAN

Simbol-simbol yang digunakan :

1  : Modul

2  : hubungan (Connection)

3  : pengulangan (*loop*)

4  : Seleksi / pemilihan

5  : Kopel Kontrol

6  : Kopel Data

BAGAN SUSUNAN

KOPEL DATA (DATA COUPLE) adalah aliran data dari modul yang memanggil ke modul yang dipanggil.

KOPEL KONTROL (CONTROL COUPLE) adalah elemen yang dikirimkan oleh modul yang dipanggil kepada modul yang memanggil sebagai tanda bahwa proses pemanggilan selesai (biasanya untuk proses pengulangan)

Pengirim Informasi

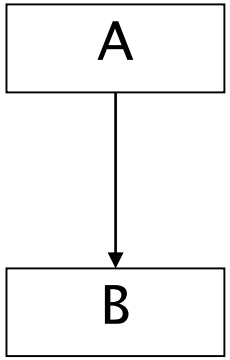
Penerima Informasi

Perbedaan Kopel Data dan Kopel Kontrol :

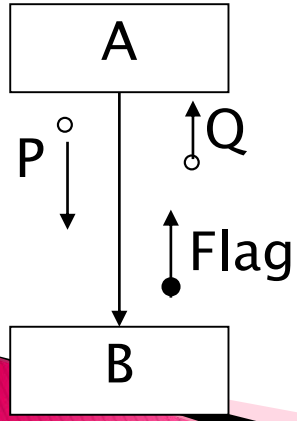
- Data biasanya berhubungan dengan permasalahan (diproses)
- Kontrol adalah alat bantu untuk implementasi tidak diproses

Contoh Bagan Terstruktur

: Suatu modul dengan nama “Hitung Potongan”

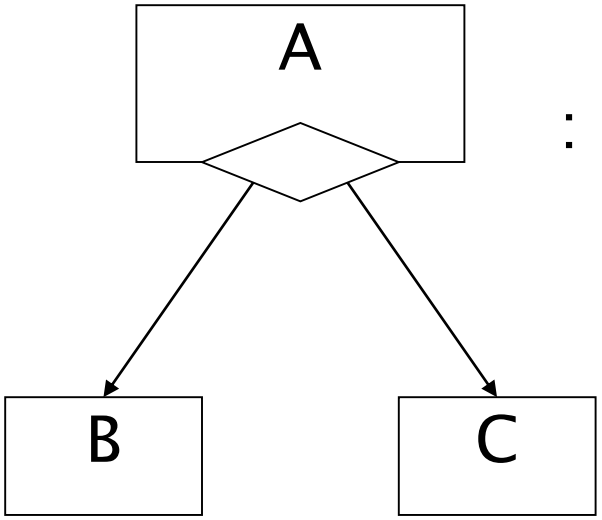


: Modul A memanggil modul B, setelah proses modul B selesai maka proses kembali ke modul A

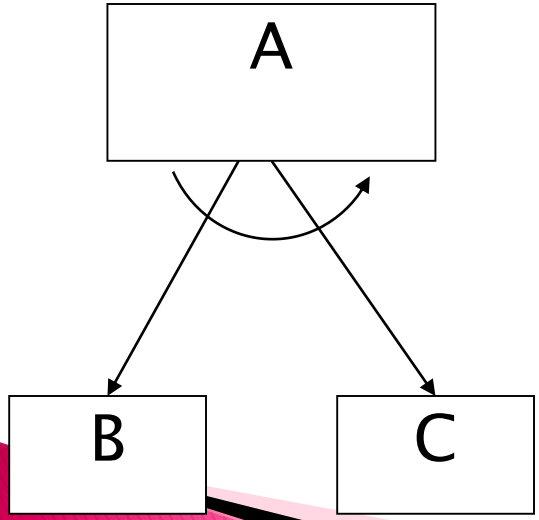


: Modul A memanggil modul B dan elemen data P dikirimkan dari modul A ke modul B. Kemudian modul B mengirimkan hasil proses (*elemen data Q* *dan elemen kontrol* ***Flag***) ke modul A .

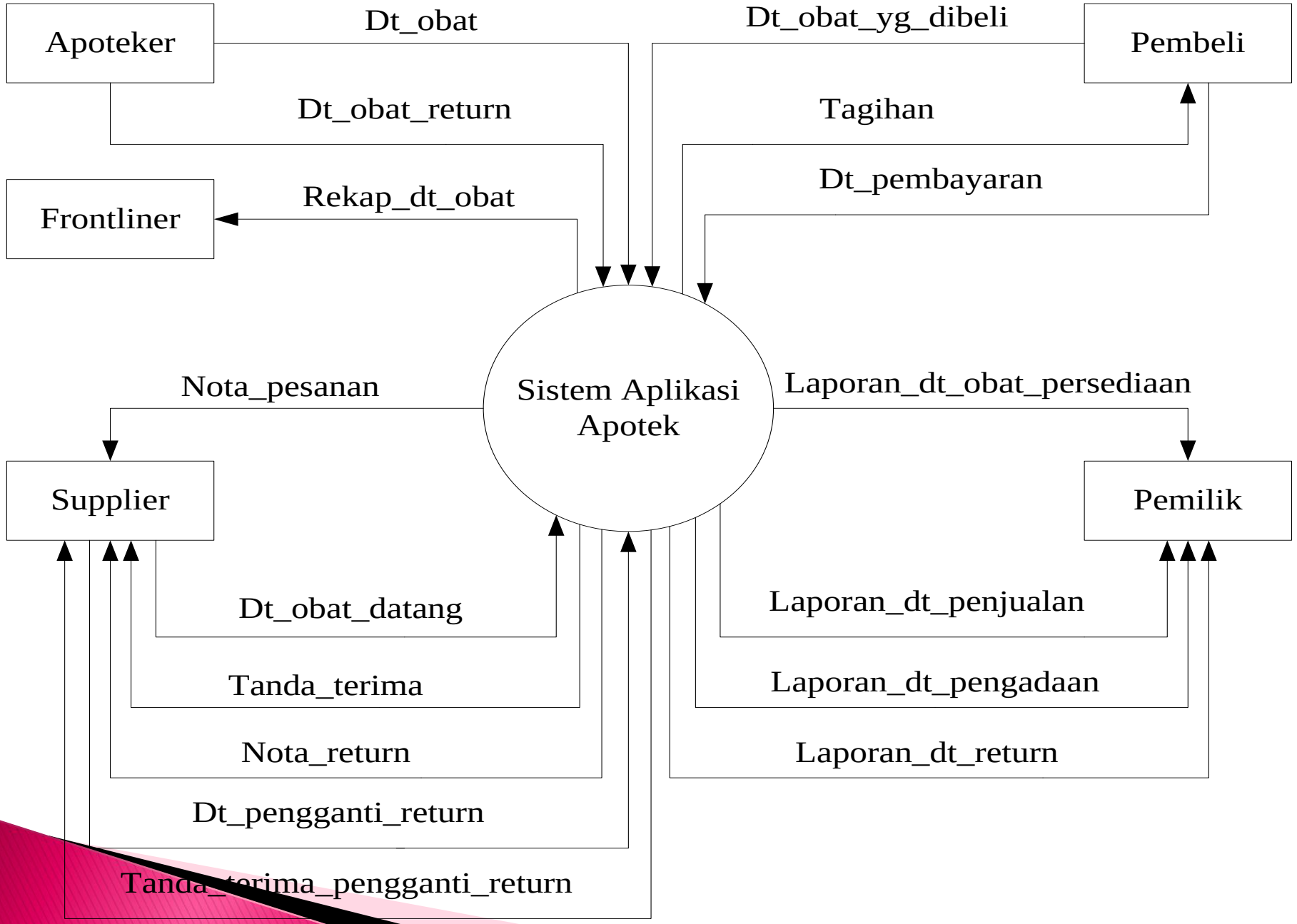
Contoh Bagan Terstruktur

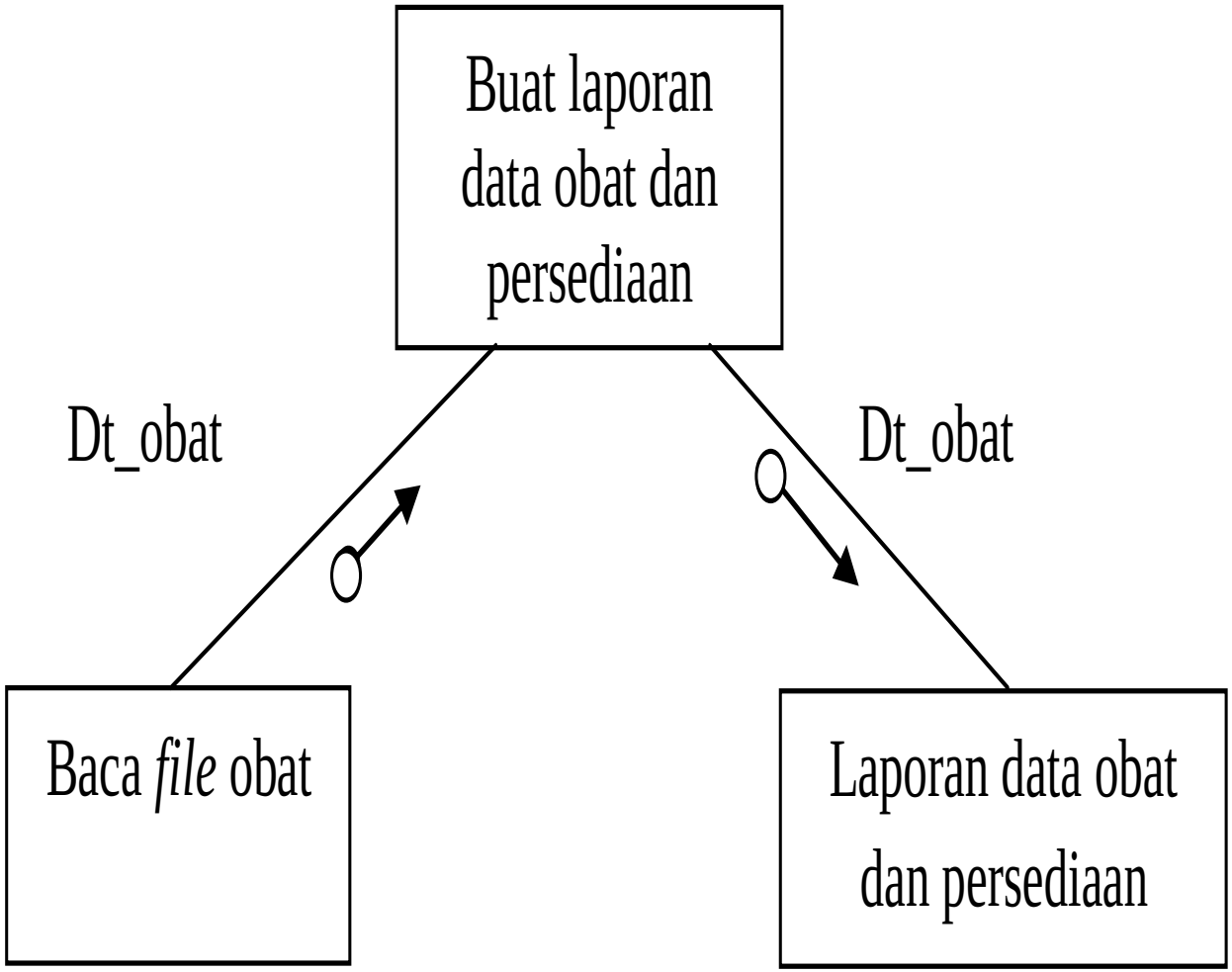


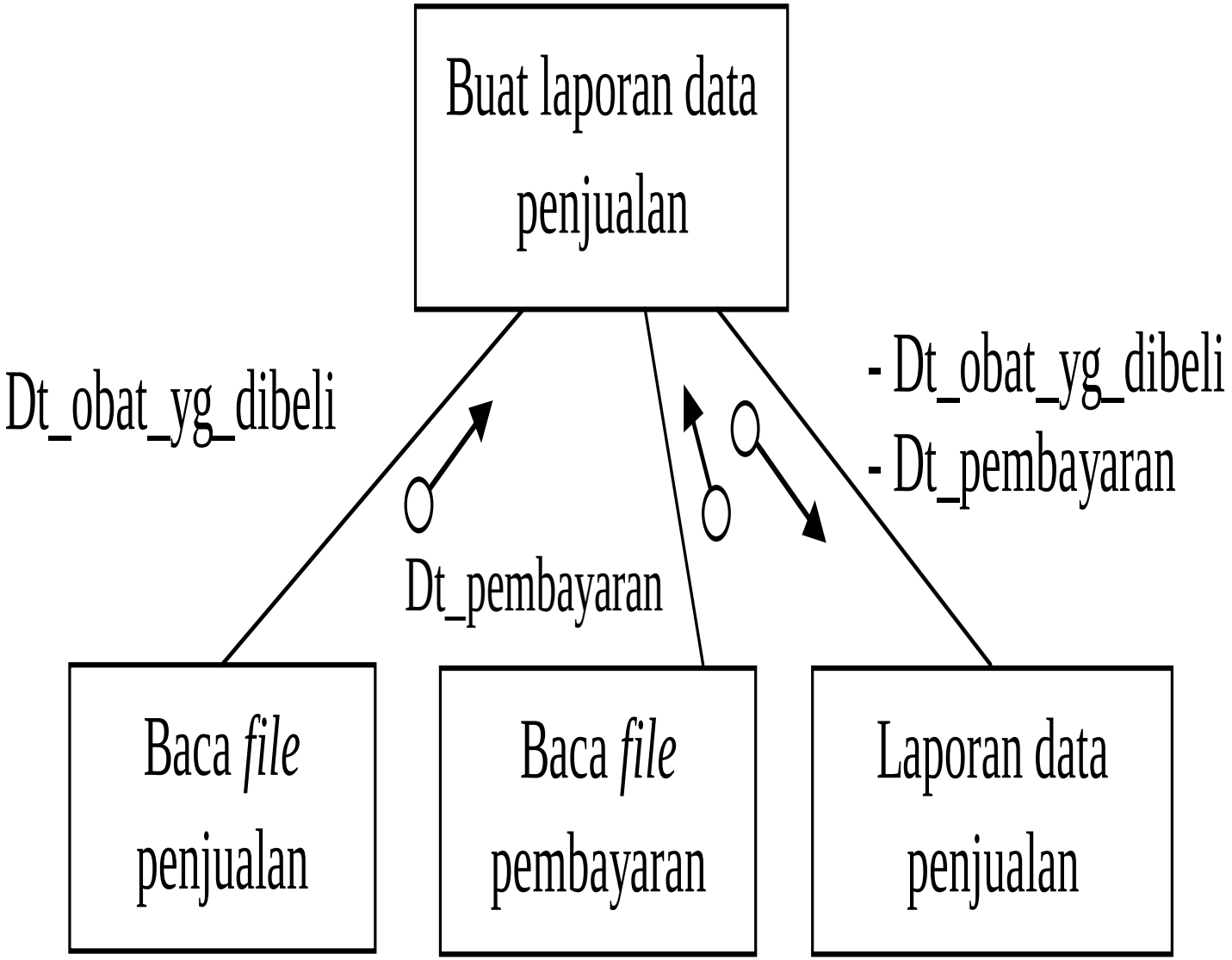
: Modul A memanggil modul B atau C sesuai dengan nilai kondisi yang diseleksi di modul terpenuhi

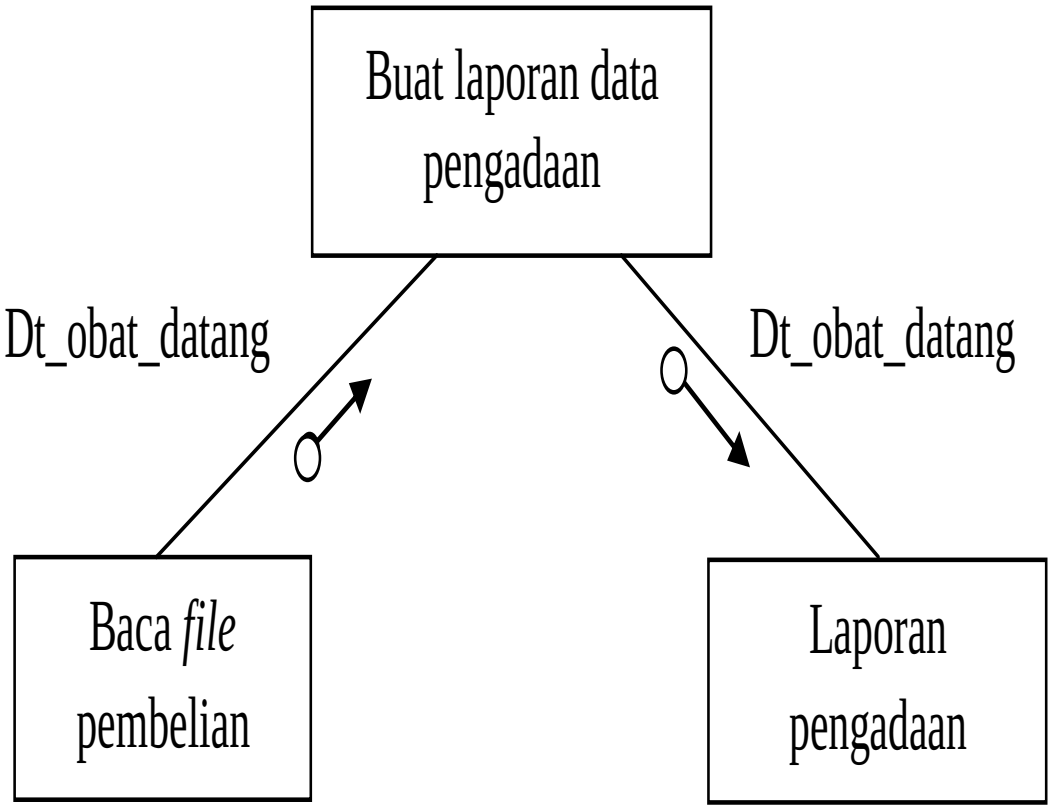


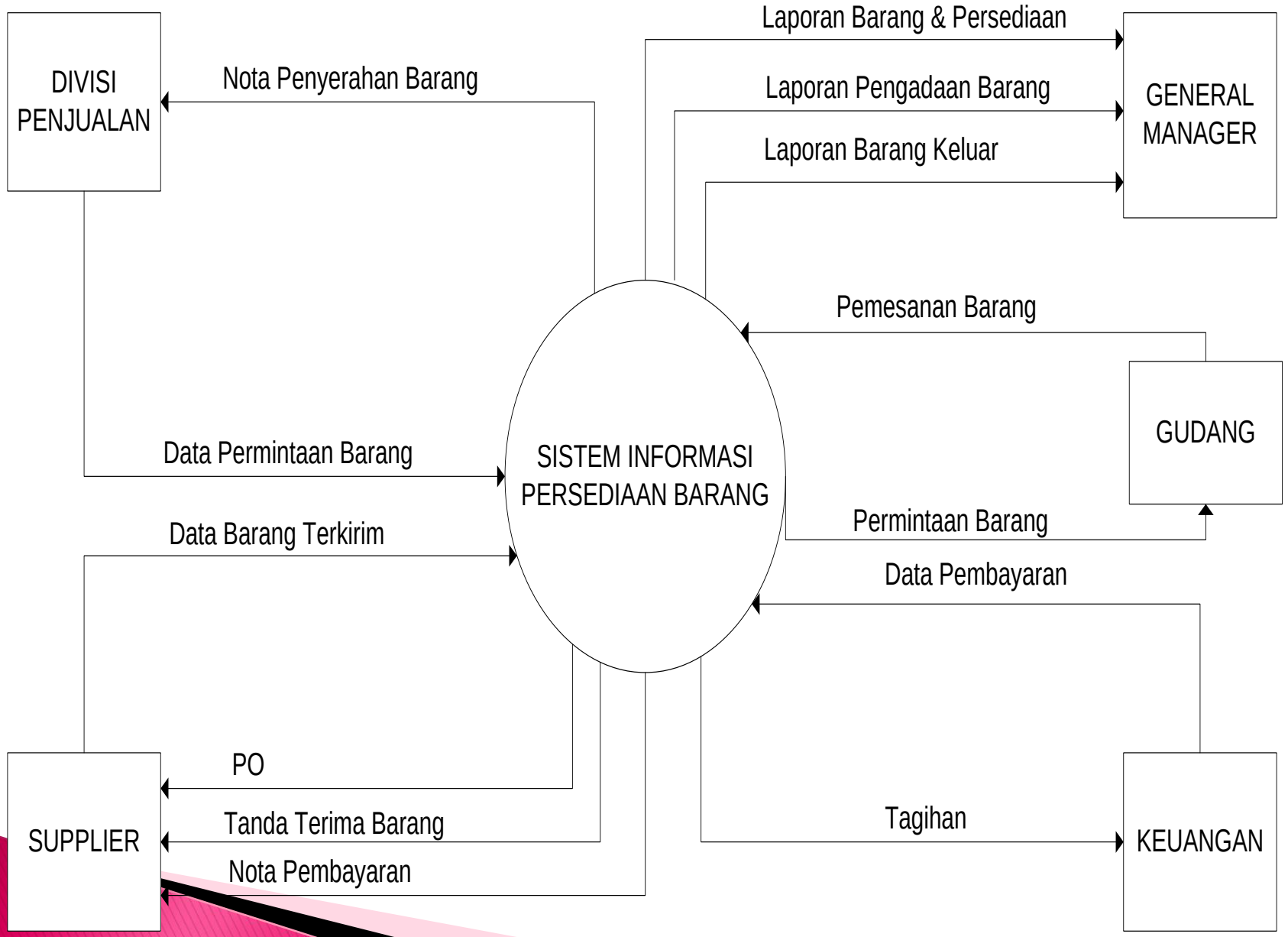
: Modul A memanggil modul B dan C secara berulang


Apoteker
Dt_obat
Dt_obat_yg_dibeli
Pembeli
Dt_obat_return
Tagihan
Frontliner
Rekap_dt_obat
Dt_pembayaran
Sistem Aplikasi Apotek
Nota_pesanan
Laporan_dt_obat_persediaan
Supplier
Pemilik
Dt_obat_datang
Laporan_dt_penjualan
Tanda_terima
Laporan_dt_pengadaan
Nota_return
Laporan_dt_return
Dt_pengganti_return
Tanda_terima_pengganti_return


Buat laporan data obat dan persediaan
Dt_obat
Dt_obat
Baca *file* obat
Laporan data obat dan persediaan


Buat laporan data penjualan
Dt_obat_yg_dibeli
Dt_pembayaran
- Dt_obat_yg_dibeli
- Dt_pembayaran
Baca *file* penjualan
Baca *file* pembayaran
Laporan data penjualan


Buat laporan data pengadaan
Dt_obat_datang
Dt_obat_datang
Baca *file* pembelian
Laporan pengadaan


DIVISI PENJUALAN
Nota Penyerahan Barang
Laporan Barang & Persediaan
Laporan Pengadaan Barang
Laporan Barang Keluar
GENERAL MANAGER
Pemesanan Barang
GUDANG
Data Permintaan Barang
SISTEM INFORMASI PERSEDIAAN BARANG
Permintaan Barang
Data Barang Terkirim
Data Pembayaran
PO
Tanda Terima Barang
Nota Pembayaran
Tagihan
SUPPLIER
KEUANGAN

TUGAS 6

Buatlah Bagan Terstruktur dari :

- Laporan data Barang dan Persediaan
- Laporan Pengadaan Barang
- Laporan pengeluaran barang