BAB V

KONSEP DAN PRINSIP DESAIN

Tujuan :

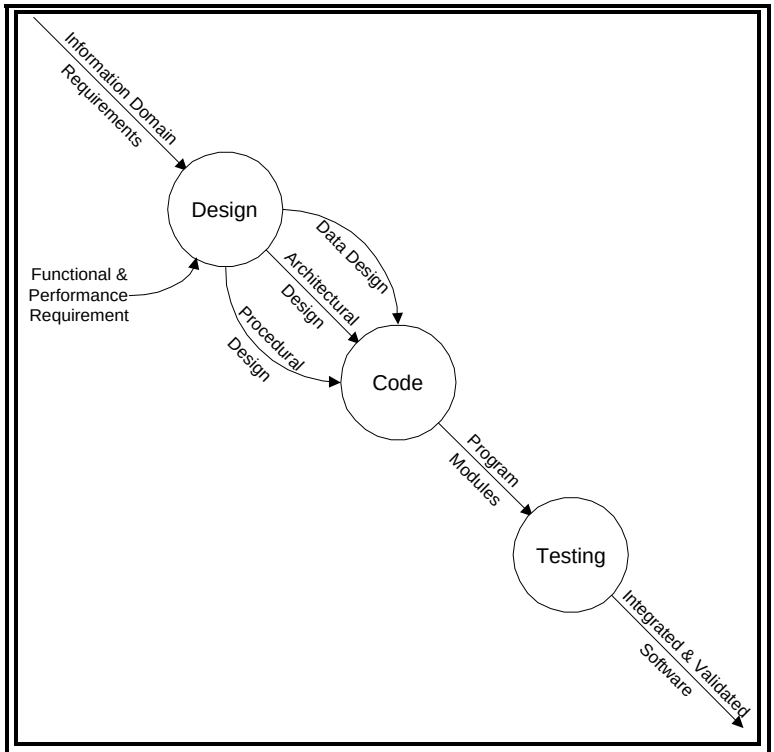
Menghasilkan suatu model atau representasi dari entitas yang kemudian akan dibangun.

FASE PENGEMBANGAN DAN DESAIN PERANGKAT LUNAK

Fase pengembangan terdiri dari 3 langkah :

1. Design
2. Code Generation (manual or automatic)
3. Testing

Setiap langkah melakukan transformasi informasi dalam suatu cara yang akhirnya menghasilkan software komputer yang valid



Development Phase

Software requirements

Dijelaskan dengan “Information Domain”, “Functional and performance requirments”, “Feed the design step”

Menggunakan metodelogi :

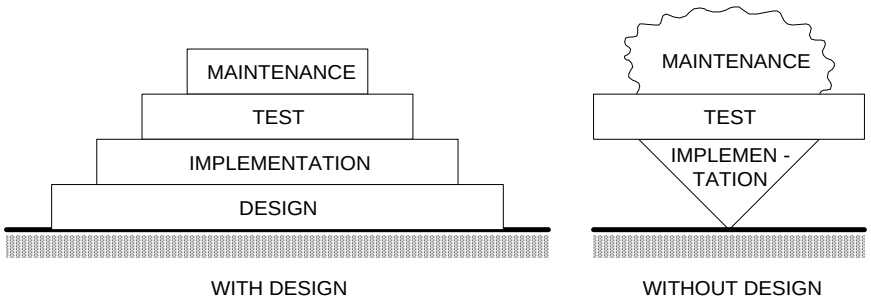
1. Data Design

2. Architectural Design
3. Procedural Design

Data Design ⇨ difokuskan pada definisi dari struktur data

Architectural Design ⇨ mendefinisikan hubungan antara elemen struktur utama dari program

Procedural Design ⇨ mengubah struktur elemen ke dalam prosedur software



Pentingnya Design

PROSES DESAIN

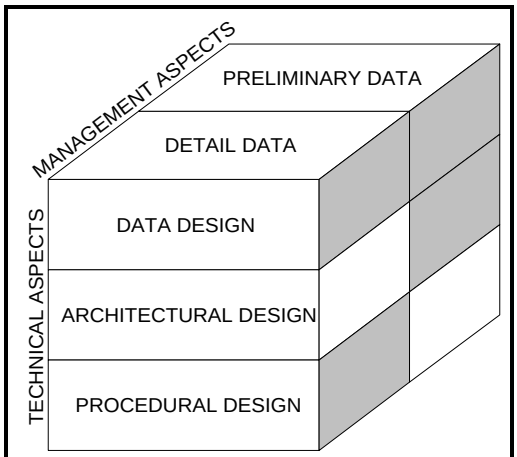
- Software design ⇨ Suatu proses yang melawati serangkaian kebutuhan yang membentuk sebuah perangkat lunak
- Software design dibagi dalam 2 tahap :

1. Preliminary Design

Pada tahap ini difokuskan dengan transformasi dari keperluan / kebutuhan ke dalam data dan arsitektur software

2. Detail Design

Difokuskan pada penghalusan representasi arsitektur yang berisi struktur data detail dan algoritma untuk software



Hubungan antara aspek teknik dan management pada desain

KUALITAS DESAIN DAN SOFTWARE

Beberapa tuntunan dalam melakukan agar dihasilkan desain dengan kriteria yang baik, yaitu suatu desain haruslah :

1. Memperlihatkan organisasi hirarki yang mengontrol elemen-elemen software
2. Berkenaan dengan modul. Software secara logika terbagi dalam elemen-elemen yang membentuk fungsi dan sub fungsi
3. Berisi representasi yang berbeda dan terpisah dari data dan prosedur
4. Membentuk modul (contoh subroutine dan procedure) yang memperlihatkan karakteristik fungsi yang tidak saling bergantung
5. Diturunkan dengan menggunakan metode perulangan yang didukung oleh informasi yang ada selama analisa kebutuhan software

EVOLUSI DESAIN SOFTWARE

- Evolusi dari desain software merupakan proses yang berkelanjutan terus selama 3 dekade
- Beberapa metodologi telah tumbuh, dan secara umum memiliki karakteristik sebagai berikut :
  1. Sebuah mekanisme untuk menterjemahkan representasi domain informasi ke dalam representasi desain
  2. Notasi untuk merepresentasikan fungsi komponen-komponen dan interfaces-nya
  3. Heuristics bagi penyaringan dan partisi
  4. Petunjuk untuk penaksiran kualitas

DASAR-DASAR DESAIN

Membantu software engineer untuk menjawab pertanyaan-pertanyaan berikut :

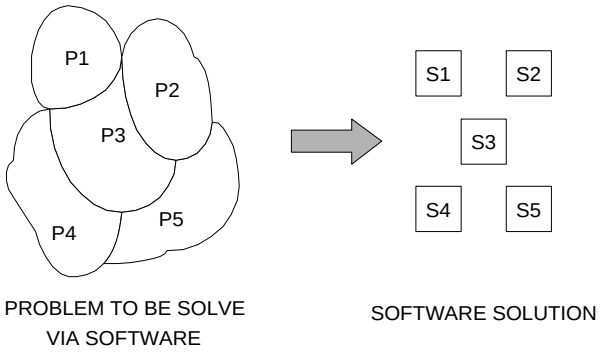
- Apakah kriteria yang dapat dipakai untuk mempartisi software menjadi sejumlah komponen ?

- Bagaimana fungsi atau struktur data dipisahkan dari suatu representasi konseptual software ?
- Apakah ada kriteria yang seragam yang menetapkan kualitas tehnik dari suatu software desain ?

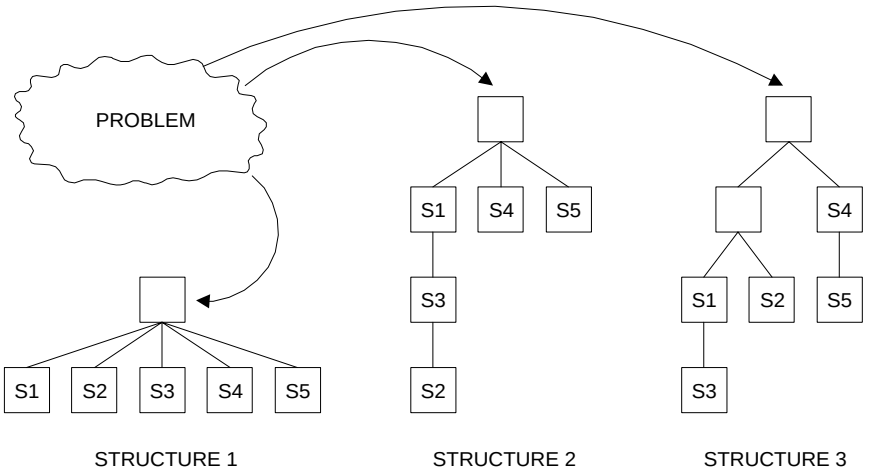
ARSITEKTUR SOFTWARE

Arsitektur perangkat lunak menyinggung 2 karakteristik penting dari sebuah program komputer :

1. Hirarki struktur dari komponen-komponen prosedural (modul)
2. Struktur data



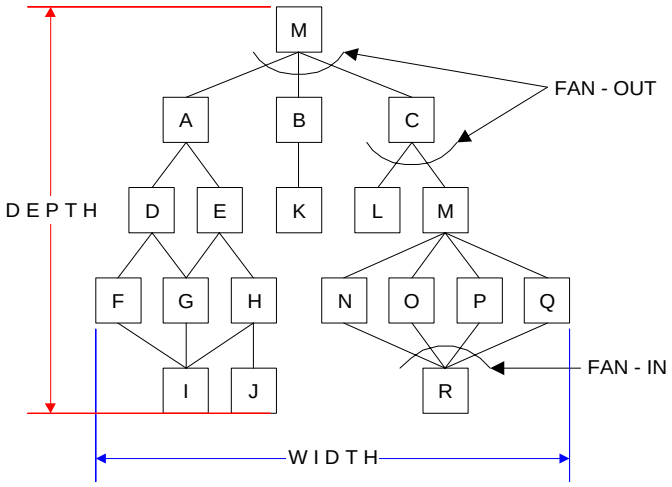
Evolution of structure



Different Structure

PROGRAM STRUCTURE

- Program structure menampilkan / menyajikan organisasi (seringkali organisasi hirarki) dari komponen-komponen program (modul-modul) dan mengandung arti hirarki dari kontrol program
- Notasi yang digunakan adalah diagram tree. Biasanya dinamakan *structure chart*



Terminologi Structure

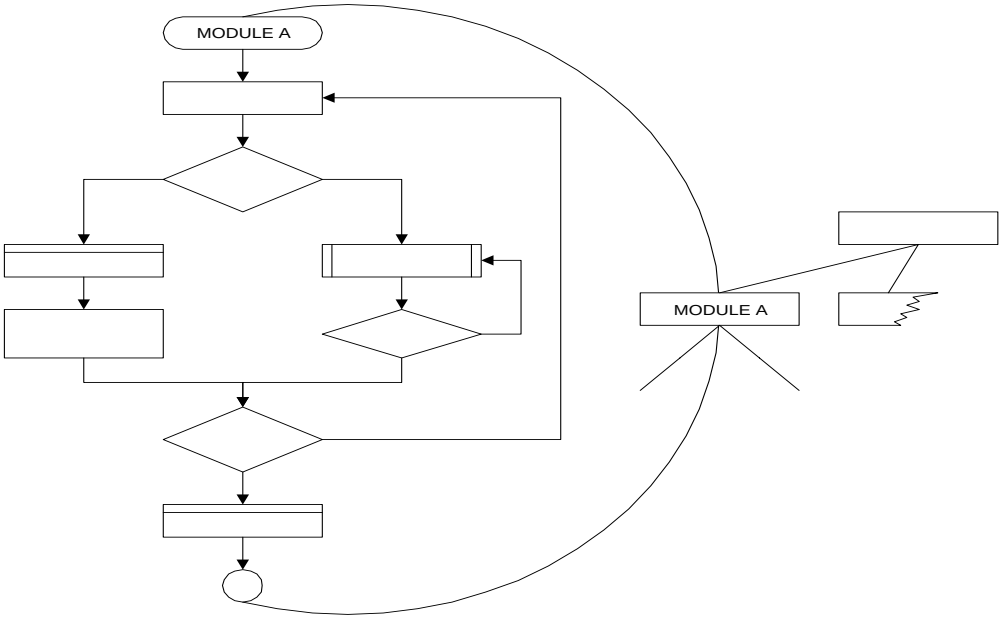
DATA STRUCTURE

Menggambarkan relasi logikal antara sejumlah elemen dari . Contoh :

```
type G = array [1..100] of integer;
...
...
Procedure S ( var T : G ; n : integer ; sum : integer );
Var
   I : integer;
begin
   sum := 0;
   for I:=1 to n do
      sum := sum + t[i];
end;
```

SOFTWARE PROCEDURE

Difokuskan pada detail pemrosesan dari setiap modul secara individu. Prosedur harus mengandung spesifikasi yang benar / tepat dari pemrosesan, termasuk : *sequence of events, decision points, repetitive operations,* dan struktur data.



Procedure dengan sebuah modul

MODULARITAS

Software dibagi kedalam nama-nama yang terpisah dan elemen-elemen yang dapat dipanggil, yang disebut dengan modul, yang termasuk kedalam memenuhi syarat-syarat permasalahan

Misalkan :

C(x) = fungsi kompleksitas dari suatu masalah

E(x) = fungsi usaha/waktu yang diperlukan untuk memecahkan suatu masalah

P_1 ,P_2 = masalah 1, masalah 2

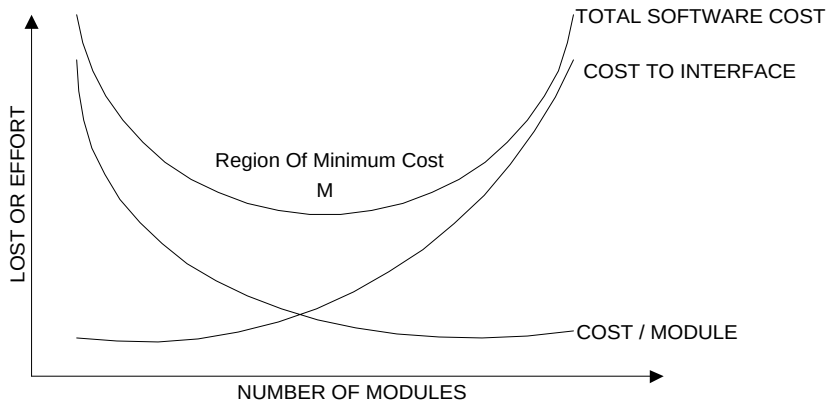
Jika : $C(P_1) > C(P_2)$ maka : $E(P_1) > E(P_2)$

Berdasarkan penelitian :

1. $C (P_1 + P_2) > C (P_1) + C (P_2)$
2. $E (P_1 + P_2) > E (P_1) + E (P_2)$

Konklusi :

1. Kompleksitas suatu masalah gabungan P_1 dan P_2 akan berkurang jika masalah tersebut dipisahkan
2. Akan lebih mudah menyelesaikan suatu masalah jika dipecah / dipartisi



Modularity & Software Cost

ABSTRAKSI

Jika kita menggunakan suatu solusi modular untuk beberapa masalah, maka beberapa level / tingkat abstrasi dapat ditampilakn / diperlihatkan.

- Pada level tertinggi, suatu solusi berada pada term yang umum dengan menggunakan bahasa natural
- Level yang lebih rendah lebih berorientasi pada prosedur-prosedur

Contoh :

Abstraksi 1

```
The software will incorporate a computer graphics interface
that will enable visual communication with the drafts person
and a digitizer interface that replace the drafting board and
square. All line and curve drawing, all geometric computation,
and all sectioning and auxiliary views will be performed by
the CAD Comp.
```

Abstraksi 2

```
CAD Software tasks :
   user interaction task ;
   2-D drawing creation task ;
   graphics display task ;
   drawing file management task ;
end.
```

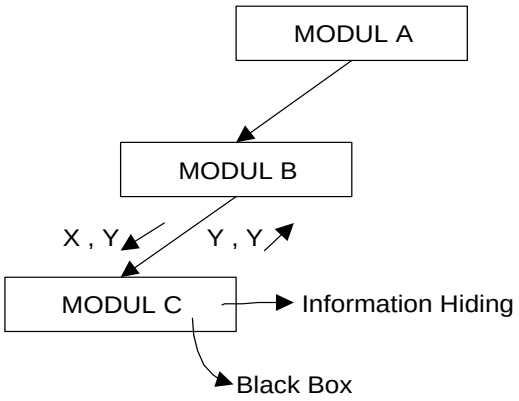
Abstraksi 3

```
procedure : 2-D drawing creation ;
   repeat until (drawing creation task terminates)
      do while (digitizer interaction occurs)
         digitizer interface task ;
         determine drawing request case ;
```

```
          line    : line drawing task ;
          circle  : circle drawing task ;
          ...
          ...
     end ;
     do while (keyboard interaction occurs)
        keyboard interaction task ;
        process analysis / computation case ;
          view    : auxiliary view task ;
          section : cross sectioning task ;
          ...
          ...
     end
     ...
     ...
  end repetition ;
end procedure.
```

PENYEMBUNYIAN INFORMASI

Contoh :



Black Box ⇨ input, output, & proses dike-tahui tetapi proses detail tidak diketahui.

Bagi Modul B, Modul C adalah Black Box

Keuntungan :

- Jika diperlukan modifikasi selama testing dan maintenance ⇨ data & prosedur disembunyikan dari bagian lain, dari program / software secara keseluruhan.
- Kesalahan-kesalahan yang terjadi selama modifikasi tidak merambat pada bagian lain.

DESAIN MODULAR EFEKTIF

Modular design ⇨ mereduksi komplesitas masalah, menyediakan fasilitas untuk melakukan perubahan (dalam hal pemeliharaan), dan memudahkan implementasi dengan pengembangan paralel dari bagian-bagian yang berbeda dalam suatu sistem

MODULE TYPES

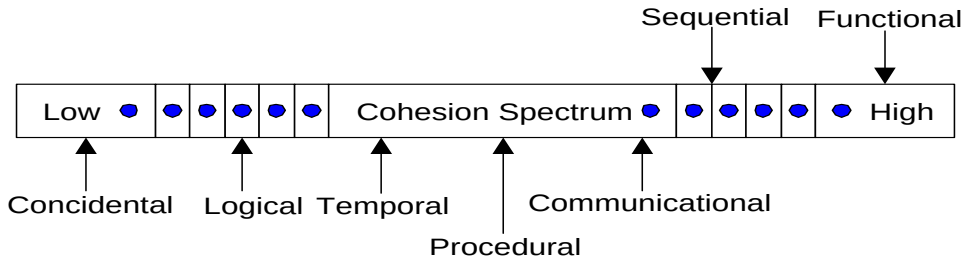
- Abstraksi dan penyembunyian informasi dipakai untuk mendefinisikan modul-modul di dalam lingkungan *software architecture*
- Di dalam structure software, suatu modul mungkin dikategorikan sebagai berikut :
  1. *Sequential module* ⇨ dieksekusi tanpa interupsi yang dilakukan software aplikasi
  2. *Incremental module* ⇨ dapat diinterupsi oleh program aplikasi dan kemudian kembali ke titik semula setelah interupsi selesai
  3. *Parallel module* ⇨ dieksekusi secara simultan dengan modul lain dalam lingkungan *Concurrent multiprocessor*

INDEPENDENSI FUNGSIONAL

- Konsep *functional independence* berkembang dari modularitas dan konsep abstraksi serta *information hiding*
- Independence diukur dengan menggunakan 2 kriteria kualitatif, yaitu
  1. Cohesion
  2. Coupling

Cohesion (keterpautan)

- Perluasan / kelanjutan dari *information hiding*
- Suatu modul kohesif membentuk sebuah tugas tunggal di dalam suatu software prosedur dan memerlukan sedikit interaksi dengan prosedur yang dibuat dalam bagian lain dari suatu program



Coincidental cohesion :

sebuah modul yang membentuk sejumlah tugas yang berhubungan satu sama lain dengan longgar

Logically cohesion :

sebuah modul yang membentuk tugas-tugas yang dihubungkan secara logical

Temporal cohesion :

jika sebuah modul berisi sejumlah tugas yang dihubungkan dengan segala yang harus dieksekusi di dalam waktu yang bersamaan.

Procedural cohesion :

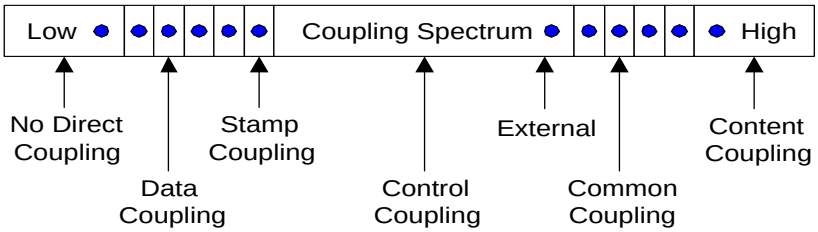
jika pemrosesan elemen-elemen dari suatu modul dihubungkan dan harus dieksekusi dalam urutan spesifik

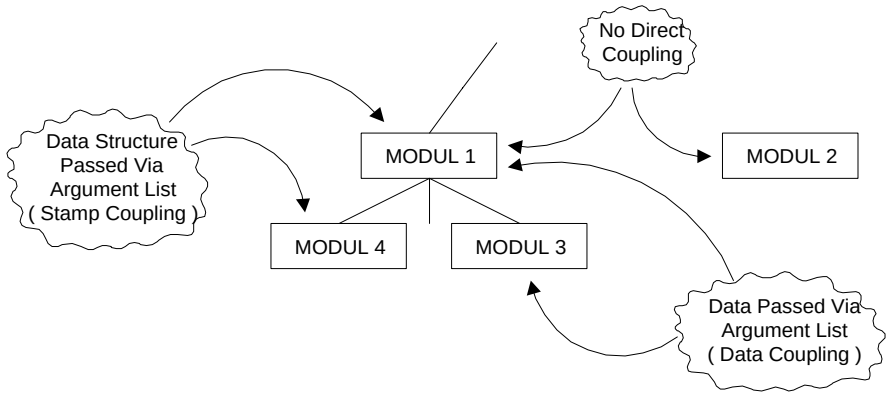
Communication cohesion :

jika pemrosesan elemen-elemen dikonsentrasikan pada satu area dari suatu struktur data

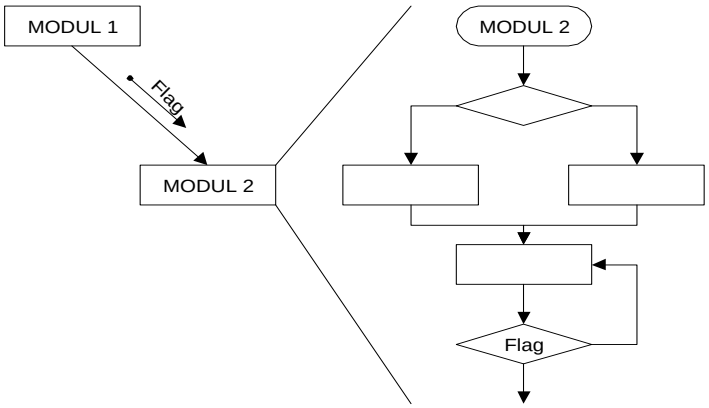
Coupling (bergandengan)

- Merupakan suatu pengukuran dari keterkaitan / keterhubungan antara sejumlah modul dalam struktur program
- Coupling tergantung pada kompleksitas interface antar modul

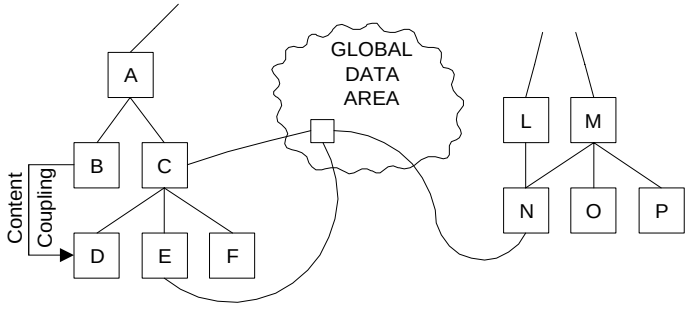




Low Coupling



Control coupling terjadi ketika modul1 mengirim-kan kontrol data ke modul2



Modul C, E, dan N menun-jukkan *common-coupling*

High Coupling

HEURISTIK DESAIN BAGI MODULARITAS YANG EFEKTIF

- Evaluasi "iterasi pertama" dari struktur program untuk mengurangi perangkaian dan meningkatkan kohesi.
- Usahakan meminimalkan struktur dengan fan-out yang tinggi; usahakan untuk melakukan fan-in pada saat kedalaman (depth) bertambah.

- Jagalah supaya lingkup efek dari suatu model ada dalam lingkup control.
- Evaluasi interface modul untuk mengurangi kompleksitas dan redundansi.
- Tetapkan modul-modul yang fungsinya dapat diprediksi, tetapi hindari modul yang terlalu restriktif.
- Usahakan modul-modul “entri kontrol" dengan menghindari “hubungan patalogis”
- Kemaslah software berdasarkan batasan desain dan persyaratan probabilitas.

MODEL DESAIN

Prinsip dan konsep desain yang dibicarakan pada bab ini membangun sebuah fondasi untuk pembuatan model desain yang mencakup representasi data, arsitektur, interface dan prosedur.

DOKUMENTASI DESAIN

Outline spesifikasi desain :

I. Ruang Lingkup
   A. Sasaran Sistem
   B. Persyaratan utama software
   C. Batasan-batasan dan pembatasan desain
II. Desain Data
   A. Obyek data dan struktur data resultan
   B. Struktur file dan database
      1. Struktur file eksternal
         a. struktur logis
         b. deskripsi record logis
         c. metode akses
      2. data global
      3. file dan referensi lintas data

III. Desain Arsitektural
    A. Kajian data dan aliran control
    B. Struktur program yang diperoleh
IV. Desain Interface
    A. Spesifikasi interface manusia-mesin
    B. Aturan desain interface manusia-mesin
    C. Desain interface eksternal
        1. Interface untuk data eksternal
        2. Interface untuk sistem atau peralatan eksternal
V. Desain Prosedural

Untuk masing-masing modul :

    A. Naratif pemrosesan
    B. Deskripsi Interface
    C. Deskripsi bahasa (atau lainnya) desain
    D. Modul-modul yang digunakan
    E. Struktur data internal
    F. Keterangan/larangan/pembatasan
VI. Persyaratan Lintas-Referensi
VII. Ketetentuan pengujian
        1. Panduan pengujian
        2. Strategi integrasi
        3. Pertimbangan Khusus
VIII. Catatan Khusus
IX. Lampiran