3 - Perancangan Terstruktur

Tujuan Perancangan

Memodelkan solusi agar siap diimplementasikan (membuat program).

Pemodelan pada Tahap Perancangan

* Perancangan Arsitektural:

Struktur Modul

* Perancangan Antarmuka:
* Antarmuka antar-modul
* Antarmuka dengan S/W lain atau H/W
* Antarmuka dengan pengguna
* Perancangan Data:

Struktur Data dan Skema Basis Data

* Perancangan Prosedural / Component - Level:

Algoritma

Analysis to Design

Diagram

Description automatically generated

Dibaca dari kanan ke kiri dengan menggunakan kata acuan, contoh untuk membuat data design maka membutuhkan Entity Relationship Diagram dan Data Dictionary sebagai acuan.

Prinsip Design

* Mempertimbangkan pendekatan alternatif.
* Dirujuk dari model analisis.
* Meminimalkan jarak intelektual.
* Menampakkan keseragaman dan integrasi.
* Terstruktur untuk mengakomodasi perubahan.
* Terstruktur untuk mendegradasi lembut.
* Direview untuk meminimalkan konseptual (semantik) error.

Elemen Struktur Modul

* Modul
* Keterkaitan antar modul

Arsitektur

Pengertian Arsitektur

Arsitektur bukan operasi perangkat lunak. Tetapi adalah representasi yang memungkinkan seorang insinyur perangkat lunak untuk:

* Menganalisis efektivitas desain dalam memenuhi persyaratan yang dinyatakannya,
* Mempertimbangkan alternatif arsitektur pada tahap ketika membuat perubahan desain masih relatif mudah.
* Mengurangi risiko yang terkait dengan pembangunan perangkat lunak.

Gaya Arsitektur

* Data-centered architectures

Diagram

Description automatically generated

* Data flow architectures

Diagram

Description automatically generated

* Call and return architectures

Diagram, engineering drawing

Description automatically generated

* Layered architectures

Diagram

Description automatically generated with low confidence

* Object-oriented architectures

Mempartisi Arsitektur

* Partisi horizontal

Mendefinisikan cabang terpisah dari hirarki modul untuk setiap fungsi utama. Menggunakan modul kontrol untuk mengkoordinasikan komunikasi antara fungsi.

Diagram

Description automatically generated

* Partisi vertical

Pengambilan keputusan dan kerja yang bertingkat. Pengambilan keputusan modul harus berada di atas arsitektur.

Diagram

Description automatically generated

Tujuan Mempartisi Arsitektur

* Menghasilkan software yang lebih mudah diuji.
* Menunjuk pada software yang lebih mudah dipelihara.
* Menghasilkan propagasi dari sedikit efek samping.
* Menghasilkan software yang lebih mudah diperluas.

Perancangan Data

Hasil Perancangan Data

* Struktur data
* Skema basis data
* Rancangan detail setiap tabel (nama, deskripsi, volume, field, key , dll)

Tahapan Merancang Data

* Review ERD

A picture containing diagram

Description automatically generated

* Memetakan menjadi skema basis data
* Tabel Mahasiswa

NIM, Nama, Alamat

* Tabel Mata Kuliah

KD\_Kul, Nama\_Kul, SKS

* Tabel Pengambilan\_Kuliah

Sem, Thn, NIM, Kd\_Kul, Nilai

Perancangan Antarmuka

Hasil Perancangan Antarmuka

* Inter-modular interface design
* External interface design
* Human-computer interface design

Arahan Merancang Antarmuka – Aturan Emas Theo Mandel

* Menempatkan pengguna sebagai pengendali
* Mengurangi beban memori pengguna
* Membuat antarmuka yang besifat konsisten

Perancangan Prosedural

Hasil Perancangan Prosedural

Rancangan detil tiap modul dan fungsi.

Notasi yang Digunakan

* Flowcharts
* Box Diagrams
* Decision Table
* Program Design Language

Kriteria Penilaian

Modularity

* Notasi mendukung pengembangan perangkat lunak.

Overall simplicity

* Mudah di pelajari, digunakan, dan ditulis.

Ease of editing

* Mudah untuk dimodifikasi ketika terjadi perubahan.

Machine readability

* Notasi dapat diinput langsung kedalam sistem.

Maintainability

* Mantenance konfigurasi melibatkan pemeliharaan desain procedural.

Structure enforcement

* Menggunakan pemrograman terstruktur.

Automatic processing

* Mengizinkan perancang untuk memverifikasi kebeneran dan kualitas perancangan.

Data representation

* Kemampuan untuk merepresentasikan data lokal dan global secara langsung.

Logic verification

* Verifikasi logika otomatis meningkatkan kecukupan pengujian.

Easily converted to program source cod

* Membuat pembuatan kode lebih cepat.