BAB II

LADASAN TEORI

2.1. Konsep Dasar Program

Berisi tentang teori-toeri yang berhubungan dengan definisi program, bahasa pemrograman Visual Basic 6.0 serta peralatan pendukung yang digunakan sebagai landasan untuk perancangan program penggajian.

A. Pengertian Program

Menurut sutedjo (2002:3) “Program adalah kata, ekspresi, pernyataan, atau kombinasinya yang disusun dan dirangkai menjadi satu kesatuan prosedur yang berupa urutan langkah untuk menyelesaikan masalah yang diimplementasikan dengan menggunakan bahasa pemrogaraman sehingga dapat dieksekusi oleh computer”. Program merupakan sederetan instruksi atau statement dalam bahasa yang dimengerti oleh computer yang bersangkutan instruksi tersebut berfungsi untuk mengatur pekerjaan saja yang dilakukan oleh computer agar mendapatkan atau menghasilkan suatu hasil atau keseluruhan atau output yang diharapkan menurut yulikuspartono (2004:29).

B. Bahasa Pemrograman

Menurut sugiyono (2009:6), “Bahasa pemrograman atau programing *language* yaitu bahasa yang digunakan untuk menulis suatau program”.

Tanpa adanya bahasa pemrograman kita tidak bias memanfaatkan computer sebagai alat mengolah atau menganalisa data. Bahasa pemrograman dapat dibedakan berdasarkan generasi diantaranya adalah sebagai berikut :

1. Bahasa Mesin (*Machine Language*)

   Merupakan bahasa yang dimengerti oleh mesin (komputer) yang didalamnya terdapat CPU yang hanya mengenal 2 (dua) keadaan yang berlawanan yaitu bila terjadi kontak (ada arus) bernilai 1, dan bila kontak terputus (tidak harus) bernilai 0.

2. Bahasa Tingkat Rendah (*Low Level Language*)

   Biasanya dalam bahasa *assembly* (*Symbolic Language*) karena ditulis dalm bentuk symbol yang agak menyerupai bahsa inggris dan mudah diingat yang disebut dengan *mnemonics*. Misalnya : A untuk kata *add* (menambahkan), 8 untuk kata (mengulangi), Mov untuk kata *move* (memindahkan).

3. Bahasa tingkat mengengah (*Middle Level Language*)

   Merupakan bahasa pemrograman yang menggunakan aturan-aturan gramatikal dalam penulisan peryataan, mudah untuk dipahami dan memiliki instruksi-instruksi tertentu yang dapat langsung diakses oleh computer, contoh bahasa C.

4. Bahasa Tingkat Tinggi (*High Level Language*)

   Merupakan bahasa pemrograman yang dalam penulisan penyalaannya mudah dipahami secara langsung.

5. Bahasa Berorientasi Obyek (*Object Oriented Language*)

Bahasa pemrograman ini mengandung fungsi-fungsi untuk menyelrsaikan suatu permasalahan dan program tidak harus ditulis secara detail semua pernyataan, tapi cukup memasukkan kriteria-kriteria yang dikehendaki saja dan penulisan instruksinya menggunakan system blok dan modular. Sehingga mudah ditelusuri programnya, apabila akan mencari dan memperbaiki kesalahan instruksi. Contoh *visual basic, visual foxpro, java*.

C. Basis Data

Menurut Kristanto (2004 : 10) Pengertian Basis data (Database) adalah kumpulan *file-file* yang mempunyai kaitan antara satu *file* dengan *file* lain sehingga membentuk suatu bangunan data untuk menginformasikan suatu perusahaan atau instansi dalam batasan tertentu.

Beberapa hal yang termasuk unsur-unsur dari basis data adalah sebagai berikut:

1) Entitas

Entitas adalah orang, tempat, kejadian atau konsep yang informasinya direkam.

2) *Field*

Setiap *entity* mempunyai atribut atau sebutan untuk mewakili suatu *entity*.

3) *Record*

Record adalah kumpulan isi elemen data *(atribut)* yang saling berhubungan menginformasikan tentang suatu *entity* secara lengkap.

4) *Data value*

Merupakan data actual atau informasi yang disimpan ditiap data elemen isi attribute disebut nilai data.

5) Kunci Elemen Data *(Key Data Element)*

Tanda pengenal yang secara unik mengidentifikasikan entitas dari suatu entitas.

D. Model Pengembangan Perangkat Lunak

Menurut Rosa dan Shalahuddin (2011:27-28) adalah Permodelan dalam suatu perangkat lunak merupakan suatu hal yang dilakukan di tahapan awal. Di dalam suatu rekayasa perangkat lunak, sebenarnya masih memungkinkan tanpa melakukan permodelan. Hal ini tidak dapat lagi dilakukan suatu industri perangkat lunak.

Permodelan adalah perangkat lunak merupakan suatu yang harus dikerjakan di bagian awal rekayasa, dan permodelan ini akan mempengaruhi pekerjaan-pekerjaan dalam rekayasa perangkat lunak tersebut.

Model perangkat lunak masih menjadi obyek penelitian, tapi sekarang ada banyak model umum atau paradigm yang berbeda dari pengembangan perangkat lunak, antara lain :

1) Pengembang *waterfall*
2) Pengembang secara *evolusioner*

3) Transformasi informal
4) Penggabungan system dengan menggunakan komponen-komponen yang dapat digunakan kembali.

Waterfall model pertama kali doperkenalkan oleh *Winston Royce* tahun 1970. *Waterfall* model merupakan model klasik yang sederhana dengan aliran sistem yang linier. Output dari setiap tahap merupakan input dari tahap berikutnya.

Model ini telah diperoleh dan diproses dari proses rekayasa lainnya dan menawarkan cara pembuatan rekayasa perangkat lunak secara lebih nyata. Model ini melibatkan tim SQA *(Software Quaintity Assurance)* dengan 5 tahapan, dimana setiap tahapan selalu dilakukan verifikasi atau testing. Tahapan model *waterfall* meliputi :

1. Analisa dan definisi persyaratan

   Dalam tahapan ini jasa, kendala dan tujuan dari konsultasi dengan pengguna sistem. Kemudian semuanya dibuat dalam bentuk yang dapat dimengerti oleh user dan staff pengembang. Dengan kata lain, dalam tahapan ini dilakukan analisa kebutuhan, kemudian diverifikasi klien dan tim SQA.
2. Perancangan system perangkat lunak

   Proses desain sistem membagi kebutuhan-kebituhan menjadi system perangkat lunak atau perangkat keras. Proses tersebut menghasilkan sebuah arsitektur keseluruhan. Desain perangkat lunak termasuk menghasilkan fungsi system perangkat lunak dalam bentuk yang

mungkin ditransformasi kedalam satu atau lebih program yang dapat dijalankan. Tahapan ini telah menentukan alur software hingga pada tahap algoritma detail. Diakhir tahapan ini, kembali diperiksa tim SQA.

3. Implementasi dan pengujian unit

   Selama tahap ini, desain perangkat lunak disadari sebagai sebuah program lengkap atau unit program. Desain yang telah disetujui, diubah dalam bentuk kode-kode program. Tahap ini, kode-kode program yang dihasilkan masih pada tahap modul-modul. Diakhiri tahap ini, tiap modul di *testing* tanpa diintegrasikan.

4. Integrasi dan pengujian sistem

   Unit program diintegrasikan dan diuji menjadi sistem yang lengkap untuk meyakinkan bahwa persyaratan perangkat lunak telah dipenuhi. Setelah uji coba, system disampaikan ke konsumen.

5. Operasi dan pemeliharaan

   Pemeliharaan termasuk pembetulan kesalahan yang tidak ditentukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru ditemukan. Setiap tahap dari model ini menggunakan *Document Drivent,* yaitu tahap selanjutnya selalu bekerja berdasarkan dokumen yang telah diberikan sebelumnya. Tahapan pada *waterfall* model tidak akan selesai jika tidak disetujui SQA.

2.2. Tools Program

A. *Enterprise Relationship Diagram* (ERD)

Menurut Ladjamudin (2013:142) ERD adalah "suatu model jaringan yang menggunakan susunan data yang disimpan dalam system secara abstrak, ERD juga menggambarkan hubungan antara satu entitas yang memiliki sejumlah atribut dengan entitas yang lain dalam suatu system yang terintegritas." ERD juga digunakan oleh professional system untuk berkomunikasi dengan pemakai eksekutif tingkat tingi dalam suatu organisasi.

Aturan-aturan dalam melakukan transformasi B-R Diagram ke *Logical Record Strucktur* (Ladjamudin, 2013:159) adalah sebagai berikut :

a. Setiap entity akan diubah ke bentuk sebuah kotak dengan nama antity berada diluar kotak dan atribut berada didalam kotak.
b. Sebuah relasi kadang disatukan dalam sebuah kotak bersama entity, kadang dipisah dalam sebuah kotak tersendiri.

Aturan pokok diatas sangat mempengaruhi elemen yang menjadi titik perhatian utama pada langkah transformasi yaitu cardinality atau kardinalitas.

1. 1:1 *(one to one)*

   Pada kardinalitas one to one, sebaiknya panah diarahkan ke entity dengan jumlah atribut yang leih sedikit.

2. 1:M *(one to many)*

   Pada kardinalitas relasi one to many, maka relasi harus digabungkan dengan entity pada pihak yang many, dantidak perlu melihat banyak sedikitnya atribut pada entity tersebut.

3. M:M *(Many to Many)*

   Pada cardinal *many to many*, maka relationship berubah status file menjadi konektor (yang akan merubah kardinalitas many to many seolah-olah menjadi one to many), sehingga baik entity maupun relasi akan menjadi structur record tersendiri. Dengan demikian, maka panah dari entity a dan b mengarah ke relationship tersebut.

B. *Logical Record Structure* (LRS)

Menurut Hasugian Dan Shidiq (2012:608) *Logical Record Structure* (LRS) adalah representasi dari struktur *record-record* pada table-tabel yang terbentuk dari hasil himpunan entitas. Menentukan kardinalitas, jumlah t*able* dan *foreign Key (FK).*

1. Konveksi *Entity Rational Diagram* (ERD) ke *Logical Record Structure* (LRS) dan Relasi.

   Diagram –ER (ERD) harus dikonversikan kebentuk *structure (structure record* secata logick). Setelah itu baru dikonveksikan kebentuk relasi (table).

2. Konveksi diagram –ER ke *Logical Record Structure* (LRS)

   Sebuah model system yang digambarkan dengan sebuah diagram –ER akan mengikuti pola/aturan permodelan tertentu. Dalam kaitannya dengan konveksi ke LRS maka perbahan yang terjadi adalah mengikuti aturan-aturan berikut ini:

   a. Setiap entitas akan diubah kebentuk kotak.

b. Sebuah relasi *relationship* kadang disatukan dalam sebuah kotak bersama entitasjika hubungan yang terjadi pada Diagramm –ER *1-M (*relasi bersatu dengan *cardinality M)* atau tingkat hubungan 1-1 (relasi bersatu dengan *cardinality* yang paling membutuhkan referensi*),* terkadang sebuah relasi dipisahkan dalam sebuah kotak tersendiri jika tingkat hubungannya *M:N (mny to many).*

c. Konveksi *Logical Record Structure (LRS) ke* Relasi (Tabel)

Relasi adalah sebuah bentuk pernyataan data secara grafis dan dimensi, yang terdiri dari kolom dan baris. Relasi adalah bentuk visual dalam sebuah file yang terdiri dari *field* dan *record.* Pada *field* mewakili sebuah atribut dan *record* gabungan dari beberapa *field.*

C. Pengkodean

Menurut Jogiyanto (2005:384)adalah "suatu susunan digit (angka), huruf dan karakter khusus yang dapat dirancang dalam bentuk kode". Kode digunakan untuk mengklasifikasi data, memasukkan data ke dalam computer dan untuk mengambil bermacam-macam informasi yang berhuubungan dengannya. Kode dapat berupa angka, huruf dan karakter khusus. Manfaat pengkodean antara lain:

1. Mempercepat atau mempersingkat proses penulisan baik dari elemen data, proses penyajian maupun pen-*entry*-an data pada computer.
2. Menghemat media penyimpanan data seperti *harddish* dan lain-lain.

3. Untuk mempermudah dan mempercepat proses pemasukan, pencarian serta pengolahan data guna memperoleh informasi yang akurat.

   Ada bebrapa macam Tipe Kode antara lain :

   a. Kode Mnemonic *(Mnemonic Code)*

      Kode Mnemonic digunakan untuk tujuan agar mudah diingat. Kode mnemonic dibuat dengan dasar singkatan atau mengambil sebagian karakter dari item yang akan diwakili dengan kode ini. Kelebihan : Mudah diigat, dan kelemahan : kode dapat menjadi panjang.

      Contoh :

      Kode pelanggan : Kode "MYS" untuk pelanggan dengan nama Muhammad Yusuf Subarkah.

   b. Kode urut (*Sequential Code)*

      Kode urut, disebut juga kode seri merupakan kode yang nilainya urut antara satu kode dengan kode berikutnya. Kelebihan : mudah diingat, kode dapat pendek, tetapi harus unik. Kelemahan : penambahan kode hanya dapat ditambhan pada akhir urutan, tidak fleksibel bila terjadi perubahan kode.

      Contoh :

      001 Kas

      002 Piutang Dagang

   c. Kode Blok *(Block Kode)*

      Kode Blok mengklasifikasikan item kedalam kelompok blok tertentu yang mencerminkan sati klasifikasi tertentu atas dasar maksimum yang diharapkan. Kelebihan : nilai dari kode mempunyai arti, kode

dapat ditambah atau dikurangi. Kelembahan : pangjang kode tergantung dari jumlah bloknya.

Contoh :

Rekening-rekening dalam Buku Besar

Blok	**Kelompok**
1000-1999	Aktiva Lancar

d. Kode Grup *(Group Code)*

Kode grup merupakan kode yang berdasarkan *field- field* dan tiap-tiap *field* kode mempunyai arti. Kelebihan : nilai kode mempunyai arti, mudah diperluas, dapat menunjukkan jenjang dari data. Kelemahan : kode dapat menjadi panjang.

e. Kode Desimal *(Decimal Code)*

Mengklasifikasikan kode atas dasar 10 unit angka decimal dimulai dari angka 0 sampai dengan angka 9 atau dari 00 sampai dengan 99 tergantung banyaknya kelompok.

Syrat-syarat yang haru s diperhatikan dalam pembuatan kode adalah sebagai berikut :

1. Harus mudah dingat
2. Harus fleksibel, jika ada perubahan tidak akan merubah semuannya.
3. Harus efisien (singkat), karen =a kode yang pendek akan mudah diingat.
4. Harus unik (beda), berarti tidk ada kode yang sama.

5. Harus konsisten, tidak boleh berubah-ubah dalam jangka pendek.
6. Harus standarisasi karena kode yang tidak standard akan mengakibatkan kebingungan, salah pengertian dan cenderung terjadi kesalahan pemakaian bagi yang menggunakan kode tersebut.
7. Hindari penggunakaan spasii
8. Panjang kode harus sama.

D. Diagram HIPO *(Hierarchy plus Input-Output)*

Menurut Jogiyanto (2005:787) "HIPO *(Hierarchy plus Input-Process-Output)* merupakan metodologi yang dikembangkan dan didukung oleh IBM. Tetapi saat ini HIPO juga banyak digunakan sebagai alat *design* dan teknik dokumentasi dalam siklus pengembangan system". HIPO dapat digunakan sebagai alat pengembangan system dan teknik dokumentasi program. Penggunaan HIPO ini mempunyai sasaran utama yaitu sebagai berikut:

1. Untuk menyediankan suatu struktur guna memahami fungsi-fungsi dari program.
2. Untuk lebih menekankan fungsi-fungsi yang harus diselesaikan oleh program, bukannya menunjukkan statemen-statemen program yang dgunkan untuk melaksanaan fungsi tersebut.

3. Untuk menyediakan penjelasan yang jelas dari *input* yang harus digunakan dan *output* yang harus dihasilkan oleh masing-masing fungsi pada tiap-tiap tingkatan dari diagram HIPO.
4. Untuk menyediakan *output* yang tepat dan sesuai dengan kebutuhan-kebutuhan pemakai.

Diagram HIPO menggunakan tiga mcam diagram untuk masing-masing tingkatannya, yaitu sebagai berikut :

a. *Visual table of contents*

Diagram ini menggambarkan hubungan dari modul-modul dalam suatu system secara berjenjang.

b. *Overview diagrams*

Overview diagram digunakn untuk menunjukkan secara hgaris besar hubungan dari i*nput* dan *output*, dimana bagian *input* menunjukkan item-item data yang akan digunakan oleh bagian proses yang berisikan langkah-langkah untuk menggambarkan kerja dari fungsi atau modul dan bagian *output* berisi hasil pemrosesan data.

c. *Detail diagram*

Detail diagram berisikan elemen-elemen dasar dari paket yang menggambarkan secara rinci kerja dari fungsi atau modul.

E. Bagan Alir *(Flowchart)*

Menurut Jogiyanto (2005:795) " bagan alir *(flowchart)* adalah bagian *(chart)* yang menunjukkan alir (*flow)* di dalam program atau prosedur system secara logika". Menurut Jogiyanto (2005:802) "Bagian alir program (program

flowchart) merupakan bagian alir yang serupa dengan bagian alir system, yaitu untuk menggambarkan prosedur didalam system".

Adapun dalam penulisan *Flowchart* dikenal dua model, yaitu sebagai berikut :

1. System *Flowchart*

   Dalam system flowchart ini bagan yang memperlihatkan urutan prosedur dan proses dari beberapa file didalam media tertentu yang dipakai dalam pengolahan data antara lain:

   a. Untuk menggambarkan file yang dipakai sebagai input dan output
   b. Tidak digunakan untuk menggambarkan urutan langkah untuk memecahkan masalah
   c. Hanya untuk menggambarkan prosedur dalam system yang dibentuk

2. *Program Flowchart*

   Pada program *flowchart* ini bagan yang memperlihatkan urutan dan hubungan proses dalam suati program. Dan ada dua jenis metode penggambaran program *flowchart* antara lain :

   a. *Conceptual flowchart,* menggambarkan alur pemecahan masalah secara global.
   b. *Detail flowchart,* menggambarkan alur pemecahan masalah secara mendetail.