BAB II

LANDASAN TEORI

2.1. Konsep Dasar Program

Konsep dasar program berisi teori-teori tentang definisi program, pemrograman, bahasa pemrograman (*programming language*) beserta jenis-jenis bahasa pemrograman. Di dalam konsep dasar program ini juga berisi teori-teori tentang *software* yang penulis pakai dalam membuat program pelayanan jasa laundry.

2.1.1. Pengertian Program

Menurut Yuswanto dalam Dewi, Kurniati, & Irmayani (Dewi, Kurniati, & Irmayani, 2017) mengemukakan bahwa, "Program merupakan kata, ekspresi pernyataan atau kombinasi yang disusun dan dirangkai menjadi satu kesatuan prosedur, berupa urutan langkah untuk menyelesaikan masalah yang diimplementasikan dengan menggunakan bahasa pemograman sehingga dapat dieksekusikan oleh komputer".

Sedangkan menurut Jogiyanto dalam Dewi et al (Dewi et al., 2017) berpendapat bahwa, "Program merupakan kegiatan menulis kode program yang akan di eksekusi oleh komputer".

Jadi dapat disimpulkan bahwa program merupakan kata, ekspresi pernyataan atau kode yang disusun sedemikian rupa menjadi kesatuan prosedur untuk menyelesaikan masalah dengan penggunaan komputer sebagai eksekutornya.

2.1.2. Bahasa Pemrograman

Menurut Kurniadi dalam Sunarti (Sunarti, 2014) memberikan batasan bahwa, “Bahasa pemrograman adalah perintah-perintah atau instruksi yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu”. Untuk membuat program tidak lepas dari karakteristik dari seorang *programer*, seperti memiliki pola pikir yang logis, memiliki ketekunan dan ketelitian, memiliki penugasan bahasa dan teknik pemrograman yang baik.

Bahasa pemrograman yang digunakan dalam penulisan tugas akhir ini, adalah: ***Visual Studio 2010***.

A. *Microsoft Visual Studio*

Menurut Ruli (Ruli, 2017) berpendapat bahwa, “*Microsoft Visual Studio* merupakan sebuah perangkat lunak lengkap yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasi lainnya dalam bentuk aplikasi *console*, aplikasi *Windows*, ataupun aplikasi *Web*”. Kompiler yang dimasukkan ke dalam paket *Visual Studio* antara lain *Visual C++*, *Visual C#*, *Visual Basic*, *Visual Basic .NET*, *Visual InterDev*, *Visual J++*, *Visual J#*, *Visual FoxPro*, dan *Visual SourceSafe*.

B. *Microsoft Visual Basic .NET*

Menurut Yuswanto dan Subari dalam Sunarti (Sunarti, 2014) mengemukakan bahwa, “Bahasa pemrograman Visual Basic. NET merupakan bahasa yang berorientasi obyek dengan mendukung empat pilar utama dari OOP, yaitu *Abtraction*, *Inheritance*, *Polymorphism* dan *Encapsulation*”. Pemrograman berorientasi obyek merupakan metode pemrograman untuk pengembangannya harus mendefinisikan tipe data dari struktur data dan juga tipe dari operasi yang

diaplikasikan kestruktur data. Dengan demikian struktur data menjadi obyek yang memiliki data dan fungsi.

Menurut Ruli (Ruli, 2017) mengemukakan bahwa, "*Microsoft Visual Basic .NET* adalah sebuah alat untuk mengembangkan dan membangun aplikasi yang bergerak di atas sistem *.NET Framework*, dengan menggunakan bahasa *BASIC*". Dengan menggunakan alat ini, para *programmer* dapat membangun aplikasi *Windows Forms*, Aplikasi *web* berbasis *ASP.NET*, dan juga aplikasi *command line*. Alat ini dapat diperoleh secara terpisah dari beberapa produk lainnya (seperti *Microsoft Visual C++*, *Visual C#*, atau *Visual J#*), atau juga dapat diperoleh secara terpadu dalam *Microsoft Visual Studio .NET*. Bahasa *Visual Basic .NET* sendiri menganut paradigma bahasa pemrograman berorientasi objek yang dapat dilihat sebagai evolusi dari *Microsoft Visual Basic* versi sebelumnya yang diimplementasikan di atas *.NET Framework*.

C. *Crystal Report*

Menurut Atmoko dalam Misriati (Misriati, 2015) memberikan pengertian bahwa, "*Crystal Report* adalah komponen yang akan kita gunakan untuk membuat *report* atau laporan dari program yang akan kita buat, agar dapat dipahami oleh pengguna, yang di mana *report* tersebut di ambil dari kumpulan data dari tabel yang tersimpan di dalam *database MySQL*".

2.1.3. Basis Data

Menurut Priyadi (2014:2) mengemukakan bahwa, "Basis data adalah sekumpulan fakta berupa representasi tabel yang saling berhubungan dan disimpan dalam media penyimpanan secara digital".

Menurut Sunarti (Sunarti, 2014) memberikan pengertian bahwa,“Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lain dan tersimpan diluar komputer serta digunakan perangkat lunak (*software*) tertentu untuk memanipulasinya”.

A. *Database Management System (DBMS)*

Menurut Sukamto dan Shalahuddin (2015:44) mengemukakan bahwa, “DBMS (*Database Manajemen System*) atau dalam bahasa Indonesia sering disebut sebagai Sistem Manajemen Basis Data adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola dan menampilkan data”. Berikut ini adalah 4 macam DBMS (*Database Manajemen System*) versi komersial yang paling banyak digunakan di dunia saat ini, yaitu:

1. *Oracle*;
2. *Microsoft SQL Server*;
3. *IBM DB2*;
4. *Microsoft Access*.

Sedangkan DBMS (*Database Manajemen System*) versi *open source* yang cukup berkembang dan paling banyak digunakan saat ini adalah sebagai berikut:

1. *MySQL;*
2. *PostgreSQL;*
3. *Firebird;*
4. *SQLite.*

Pada program pelayanan jasa laundry di penulisan tugas akhir ini penulis menggunakan DBMS (*Database Manajemen System*) *MySQL* untuk melengkapi *database* di dalam program.

B. *Structured Query Language (SQL)*

Menurut Sukamto dan Shalahuddin (2015:46) memberikan pengertian bahwa, "SQL (*Structured Query Language*) adalah bahasa yang digunakan untuk mengelola data pada RDBMS". SQL awalnya dikembangkan berdasarkan teori aljabar relasional dan kalkulus.

C. *MySQL*

Menurut Damayanti & Wardati (Damayanti & Wardati, 2016) memberikan pengertian bahwa, "*MySQL* adalah program *database* yang mampu mengirim dan menerima data dengan sangat cepat dan multiuser". *MySQL* memiliki dua bentuk lisensi, yaitu *free software* dan *shareware*. *MySQL* yang *free software* bebas digunakan untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensi, yang berada di bawah lisensi GNU/GPL (*general public license*).

D. *Web Server*

Menurut Pratama (2014:439) mengemukakan bahwa, "*Web server* merupakan perangkat lunak yang dijalankan di sistem operasi pada komputer server maupun desktop, yang berfungsi untuk menerima permintaan (*request*) dalam bentuk protokol, misalkan HTTP (*Hyper Text Transfer Protocol*) dan HTTPS (*Hyper Text Transfer Protocol Secure*)".

Menurut Fathansyah (2015:466) memberikan pengertian bahwa, "*Server Web (Web Server)* merujuk pada perangkat keras (*server*) dan perangkat lunak yang menyediakan layanan akses kepada pengguna melalui protokol komunikasi HTTP ataupun variannya (seperti FTP dan HTTPS) atas berkas-berkas yang terdapat pada suatu URL ke pemakai". Lebih jauh lagi, *server Web* juga dapat

berinteraksi dengan basis data, sehingga untuk mengelolanya juga diperlukan DBMS dan aplikasi basis data. Interaksi *server Web* dengan basis data (yang juga dapat ditempatkan di *server* terpisah) akan membuat penayangan data bersifat dinamis/interaktif sehingga dapat dimanfaatkan pula untuk aplikasi bisnis.

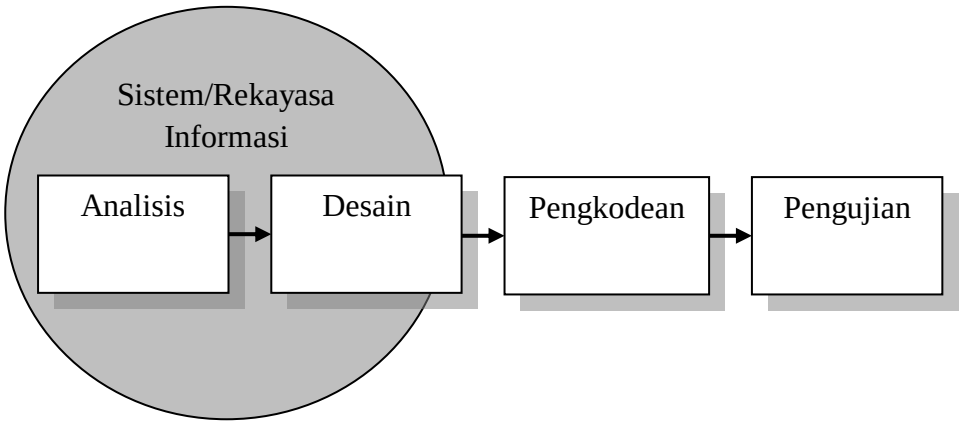
Pada program pelayanan jasa laundry di penulisan tugas akhir ini penulis menggunakan *Web Server* XAMPP untuk berinteraksi dengan basis data di dalam program.

E. *Xampp*

Menurut Pratama (2014:440) memberikan pengertian bahwa, "*XAMPP* adalah aplikasi *web server* bersifat instan (siap saji) yang dapat digunakan baik di sistem operasi Linux maupun di sistem operasi Windows".

2.1.4. Model Pengembangan Perangkat Lunak

Menurut Sukamto dan Shalahuddin (2015:28) mengemukakan bahwa, "Model SDLC air terjun (*Waterfall*) sering juga disebut model sekuensi linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*)". Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian dan tahap pendukung (*support*). Berikut adalah gambar model air terjun:



Sumber: Sukamto dan Shalahuddin (2015:28)

Gambar II.1. Ilustrasi Model *Waterfall*

5. Analisis kebutuhan perangkat lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

6. Desain

   Desain perangkat lunak adalah proses multilangkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

7. Pembuatan kode program

   Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

8. Pengujian

   Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

9. Pendukung (*support*) atau pemeliharaan (*maintenance*)

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.2. *Tools* Program

Dalam penulisan tugas akhir ini *Tools* Program yang digunakan oleh penulis meliputi *Entity Relationship Diagram* (*ERD*), *Logical Record Structure (LRS)*, pengkodean, HIPO (*Hierarky Input Process Output*), *Flowchart*, dan *Black-box Testing*.

2.2.1. *Entity Relational Diagram (ERD)*

Menurut Sukamto dan Shalahuddin (2015:53) mengemukakan bahwa, "ERD adalah bentuk paling awal dalam melakukan perancangan basis data relasional". Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan *Entity Relationship Diagram* (ERD). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional. Sehingga jika penyimpanan basis data menggunakan OODBMS maka perancangan basis data tidak perlu menggunakan ERD. ERD memiliki beberapa aliran notasi seperti notasi Chen (dikembangkan oleh Peter Chen), Barker (dikembangkan oleh Richard Barker, Ian Palmer, Harry Ellis), notasi Crow's Foot, dan beberapa notasi lain. Namun yang banyak digunakan adalah notasi dari Chen.

Menurut Pratama (2014:49) memberikan pengertian bahwa, "ERD (*Entity Relationship Diagram*) adalah diagram yang menggambarkan keterkaitan antartabel beserta dengan *field-field* di dalamnya pada suatu database sistem". Sebuah database memuat minimal sebuah tabel dengan sebuah atau beberapa buah field (kolom) di dalamnya. Namun pada kenyataannya, database lebih sering memiliki lebih dari satu buah tabel (dengan beberapa field di dalamnya).

Menurut Ladjamudin dalam Misriati (Misriati, 2015) menyimpulkan bahwa, "ERD adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak". ERD lebih menekankan pada struktur-struktur dan *relationship* data. Elemen-elemen ERD adalah:

1. Entitas (*Entity*)

   Digambarkan dengan sebuah bentuk persegi panjang dan digunakan untuk menunjukan sekumpulan orang, tempat, objek atau konsep dan sebagainya yang menunjukkan dimana data dicatat atau disimpan.

2. Hubungan atau Relasi

   Digambarkan dengan kotak berbentuk diamond atau belah ketupat dengan garis yang menghubungkan ke *entity* yang terkait. Maka *relationship* diberi nama dengan kata kerja. Hubungan atau relasi menunjukkan antara *entity* yang berbeda.

3. Atribut

   Menunjukkan karakteristik dari tiap *entity* atau sesuatu yang menjelaskan entitas atau hubungan, sehingga atribut dikatakan elemen data dari entitas dan *relationship*. Dari setiap atribut-atribut entitas terdapat satu atribut yang dijadikan sebagai kunci (*key*).

4. *Cardinality* (tingkat hubungan)

   Kardinalitas menunjukkan tingkat hubungan yang terjadi, dilihat dari segi kejadian atau banyak tidaknya hubungan antar entitas tersebut. Ada tiga kemungkinan hubungan yang ada, yaitu:

   a. Satu ke satu (*one to one* atau 1:1)

      Tingkat hubungan dinyatakan satu ke satu jika suatu kejadian pada entitas pertama hanya mempunyai satu hubungan dengan satu kejadian pada entitas kedua. Demikian juga sebaliknya, satu kejadian pada entitas yang kedua hanya bisa mempunyai satu kejadian pada entitas yang pertama.

b. Satu ke banyak (*one to many* atau 1:M)

Tingkat hubungan satu ke banyak (1:M) adalah sama dengan banyak ke satu (M:1), tergantung dari arah mana hubungan-hubungan tersebut dilihat. Untuk satu kejadian pada entitas yang pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas yang kedua. Sebaliknya satu kejadian pada entitas yang kedua hanya bisa mempunyai satu hubungan dengan satu kejadian pada entitas yang pertama.

c. Banyak ke banyak (*many to many* (M:N)

Tingkat hubungan banyak ke banyak terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan denfan kejadian pada entitas lainnya. Baik dilihat dari sisi entitas yang pertama maupun dilihat dari sisi entitas yang kedua.

2.2.2. *Logical Record Structure (LRS)*

Menurut Friyadie dalam Taufik & Ermawati (Taufik & Ermawati, 2017) mengemukakan bahwa, "Sebelum tabel dibentuk dari field atau atribut entitas secara fisik atau level internal, maka harus dibuatkan suatu bentuk *relational model* yang dibuat secara *logic* atau level *external* dan konsep, dari pernyataan tersebut dibutuhkan yang disebut dengan *Logical Record Structure* (LRS)".

Dalam pembuatan *Logical Record Structure* (LRS) terdapat tiga hal yang dapat mempengaruhi, yaitu:

1. Jika tingkat hubungan (*cardinality*) satu pada satu (*one-to-one*), maka digabungkan dengan entitas yang lebih kuat (*strong entity*), atau digabungkan dengan entitas yang memiliki atribut yang lebih sedikit.

2. Jika tingkat hubungan (*cardinality*) satu pada banyak (*one-to-many*), maka hubungan relasi atau digabungkan dengan entitas yang tingkat hubungannya banyak.
3. Jika tingkat hubungan (*cardinality*) banyak pada banyak (*many-to-many*), maka hubungan relasi tidak akan digabungkan dengan entitas manapun, melainkan menjadi sebuah LRS.

2.2.3. Pengkodean

Menurut Mustakini (2014:384) mengemukakan bahwa, "Kode digunakan untuk tujuan mengklasifikasikan data, memasukan data ke dalam komputer dan untuk mengambil bermacam-macam informasi yang berhubungan dengannya". Kode dapat dibentuk dari kumpulan angka, huruf dan karakter-karakter khusus (misalnya % , /, -, $, #, &, :, dan lain sebagainya). Angka merupakan simbol yang banyak digunakan pada sistem kode. Akan tetapi kode yang berbentuk angka lebih dari 6 digit akan sulit untuk diingat. Kode numerik (*numeric code*) menggunakan 10 macam kombinasi angka di dalam kode. Kode alphabetik (*alphabetic code*) menggunakan 26 kombinasi huruf untuk kodenya. Kode alphanumerik (*alphanumeric code*) merupakan kode yang menggunakangabungan angka., huruf dan karakter-karakter khusus. Meskipun kode numerik, alphabetik, dan alphanumerik merupakan kode yang paling banyak digunakan di dalam sistem informasi, tetapi kode yang lain juga mulai banyak digunakan, seperti misalnya kode batang (*bar code*).

Di dalam merancang suatu kode harus diperhatikan beberapa hal Mustakini (2014:384) yaitu:

1. Harus mudah diingat

   Supaya kode mudah diingat, maka dapat dilakukan dengan cara menghubungkan kode tersebut dengan obyek yang diwakili dengan kodenya.

2. Harus unik

   Kode harus unik untuk masing-masing item yang diwakilinya. Unik berarti tidak ada kode yang kembar.

3. Harus Fleksibel

   Kode harus fleksibel sehingga memungkinkan perubahan-perubahan atau penambahan item baru dapat tetap diwakili oleh kode.

4. Harus Efisien

   Kode harus sependek mungkin, selain mudah diingat juga akan efisien bila direkam di simpanan luar komputer.

5. Harus Konsisten

   Bilamana mungkin, kode harus konsisten dengan kode yang telah dipergunakan.

6. Harus distandarisasi

   Kode harus di standarisasi untuk seluruh tingkatan dan departemen dalam organisasi.

7. Spasi dihindari

   Spasi didalam kode sebaiknya dihindari, karena dapat menyebabkan kesalahan di dalam menggunakannya.

8. Hindari karakter yang mirip

   Karakter-karakter yang hampir serupa bentuk dan bunyi pengucapannya sebaiknya tidak digunakan dalam kode.

9. Panjang kode harus sama

   Masing-masing kode yang sejenis harus mempunyai panjang yang sama.

Menurut Mustakini (2014:386) berpendapat bahwa, “Ada beberapa macam tipe dari kode yang dapat digunakan di dalam sistem informasi, diantaranya adalah kode mnemonik (*mnemonic code*), kode urut (*sequential code*), kode blok (*block code*), kode group (*group code*), dan kode desimal (*decimal code*)”. Masing-masing tipe dari kode tersebut mempunyai kebaikan dan kelemahan tersendiri. Dalam praktek, tipe-tipe kode yang ada dapat di kombinasikan.

Pengertian macam-macam kode menurut Mustakini (2014:386) yaitu:

1. Kode Mnemonik

   Kode mnemonik (*mnemonik code*) digunakan untuk tujuan supaya mudah diingat. Kode mnemonik dibuat dengan dasar singkatan atau mengambil sebagian karakter dari item yang akan diwakili dengan kode ini. Umumnya kode mnemonik menggunakan huruf akan tetapi dapat juga menggunakan gabungan huruf dan angka. Misalnya barang dagangan komputer IBM PC dengan ukuran memori 640 KB, *colour monitor*, dapat dikodekan menjadi KIBM-PC-640-CO supaya lebih mudah diingat dan kelemahannya adalah kode dapat menjadi terlalu panjang.

2. Kode Urut

   Kode urut (*sequential code*) disebut juga dengan kode seri (*serial code*) merupakan kode yang nilainya urut antara satu kode dengan kode berikutnya.

3. Kode Blok

   Kode blok (*block code*) mengklasifikasikan item ke dalam kelompok blok tertentu yang mencerminkan satu kalasifikasi tertentu atas dasar pemakaian maksimum yang diharapkan.

4. Kode Group

   Kode Group (*group code*) merupakan kode yang berdasarkan *field-field* dan tiap *field* kode mempunyai arti. Dalam buku-buku teks, maka akan terlihat suatu kode yang disebut dengan ISBN (*International Standart Book Number*) yang terdiri dari 10 digit terbagi dalam 4 *field*. ISBN merupakan kode group yang masing-masing field mempunyai arti tertentu.

5. Kode Desimal

   Kode Desimal (*decimal code*) mengklasifikasikan kode atas dasar 10 unit angka desimal mulai dari angka 0 sampai dengan angka 9 atau dari angka 00 sampai 99 tergantung dari banyaknya kelompok.

2.2.4. HIPO (*Hierarchy Input Proses Output*)

A. Pengertian

> Mustakini (2014:787) berpendapat bahwa:
> HIPO (*Hierarchy plus Input-Process-Output*) merupakan metodologi yang dikembangkan dan didukung oleh IBM. HIPO sebenarnya adalah alat dokumentasi program. Akan tetapi sekarang HIPO juga banyak digunakan sebagai alat disain dan teknik dokumentasi dalam siklus pengembangan sistem. HIPO berbasis pada fungsi, yaitu tiap-tiap modul di dalam sistem digambarkan oleh fungsi utamanya.

HIPO dapat digunakan sebagai alat pengembangan sistem dan teknik dokumentasi program dan penggunaan HIPO ini mempunyai sasaran utama sebagai berikut ini.

1. Untuk menyediakan suatu struktur guna memahami fungsi-fungsi dari sistem.
2. Untuk lebih menekankan fungsi-fungsi yang harus diselesaikan oleh program.
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-diagram HIPO.
4. Untuk menyediakan output yang tepat dan sesuai dengan kebutuhan kebutuhan pemakai.

B. Tingkatan Diagram HIPO

Fungsi-fungsi dari sistem digambarkan oleh HIPO dalam tiga tingkatan yaitu:

a. *Visual Table of Content* (VTOC)

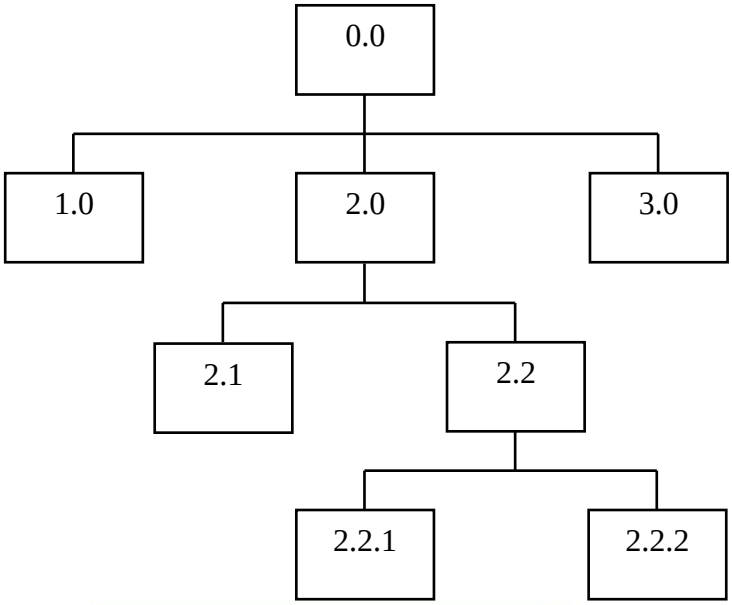
Diagram ini menggambarkan hubungan dari fungsi – fungsi di sitem secara berjenjang.

b. *Overview* Diagram

Overview Diagram menunjukkan secara garis besar hubungan dari *input*, proses, *output*. Bagian input menunjukkan item-item data yang akan digunakan oleh bagisan proses. Bagian proses berisi jumlah langkah-langkah yang menggambarkan kerja dan fungsi. Bagianoutput berisi item-item data yang dihasilkan atau dimodifikasi oleh langkah-langkah proses.

c. Detail *Diagrams*

Detail *Diagrams* merupakan diagram tingkatan yang paling rendah di diagram HIPO. Diagram ini berisi dengan elemen-elemen dasar dari paket yang menggambarkan secara rinci kerja dan fungsi.



Sumber: Mustakini (2014:788)

Gambar II.2. *Visual table of contents*

2.2.5. Diagram Alir Program (*Flowchart*)

A. Pengertian

Menurut Mustakini (2014:795) mengemukakan bahwa, “Bagan alir (*fowchart*) adalah bagan (*chart*) yang menunjukkan alir (*flow*) di dalam program atau prosedur sistem secara logika”. Bagan alir digunakan terutama untuk alat bantu komunikasi dan untuk dokumentasi. Pada waktu akan menggambat suatu bagan alir, analis sistem atau pemrogram dapat mengikuti pedoman-pedoman sebagai berikut ini:

1. Bagan alir sebaiknya digambar dari atas ke bawah dan mulai dari bagian kiri dari suatu halaman.
2. Kegiatan di dalam bagan alir harus ditunjukkan dengan jelas.

3. Harus ditunjukkan dari mana kegiatan akan dimulai dan dimana akan berakhirnya.
4. Masing-masing kegiatan di dalam bagan alir sebaiknya digunakan suatu kata yang mewakili suatu pekerjaan, misalnya:
   a. "Persiapkan" dokumen
   b. "Hitung" gaji
5. Masing-masing kegiatan di dalam bagan alir harus di dalam urutan yang semestinya.
6. Kegiatan yang terpotong dan akan disambung ditempat lain harus ditunjukkan dengan jelas menggunakan simbol penghubung.
7. Gunakanlah simbol-simbol bagan alir yang standar.

B. Bentuk *Flowchart*

Terdapat dua bentuk *Flowchart* yang dapat digunakan yaitu:

a. Program *Flowchart*

Yaitu simbol-simbol *Flowchart* yang digunakan untuk menggambarkan logik dari pemrosesan terhadap data.

b. Sistem *Flowchart*

Yaitu simbol-simbol peralatan sistem komputer yang digunakan untuk menyatakan proses pengolahan data.

C. Tehnik Pembuatan

Terdapat dua tehnik pembuatan *Flowchart* dalam menyusun logika suatu program yaitu:

a. *General Way*

Teknik pembuatan dengan cara ini lazim digunakan dalam menyusun logika suatu program, yang menggunakan proses secara tidak langsung.

b. *Iteration Way*

Teknik pembuatan *flowchart* dengan cara ini biasanya dipakai untuk logika program yang tepat dan juga bentuk permasalahan yang kompleks.

2.2.6. *Black-Box Testing*

Menurut Sukamto dan Shalahuddin (2015:275) memberikan pengertian bahwa, "*Black box testing* adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program". Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah.

Menurut Rusadi dalam Sintawati & Sari (Sintawati & Sari, 2017) "*Black-box testing* adalah metode pengujian perangkat lunak yang tes fungsionalitas dari aplikasi yang bertandatangan dengan struktur internal atau kerja. Menggunakan deskripsi eksternal perangkat lunak, termaksud spesifikasi persyaratan, dan desain untuk menurunkan uji kasus".