BAB II

TINJAUAN PUSTAKA

A. Alpukat

Tanaman alpukat berasal dari daerah tropik Amerika. Nikolai Ivanovich Vavilov seorang ahli botani Soviet, memastikan sumber genetik alpukat berasal dari Meksiko bagian selatan dan Amerika Tengah, kemudian menyebar ke daerah yang beriklim tropik. Tanaman alpukat masuk ke Indonesia pada zaman kerajaan Hindu dan ketika Islam masuk ke Indonesia. Pada mulanya pengembangan tanaman alpukat terkonsentrasi di Pulau Jawa, tetapi saat ini telah menyebar hampir di setiap provinsi di Indonesia. Berdasarkan data Warta penyuluhan pertanian tanaman pangan, daerah sentra produksi alpukat adalah provinsi Jawa Barat dan sekarang mulai menyebar ke segala provinsi (Ir.H. Rahmat Rukmana, 2011:13)

Tanaman alpukat merupakan tanaman buah berupa pohon dengan nama alpuket (Jawa Barat), alpokat (Jawa Timur/Jawa Tengah), boah pokat, jamboo pokat (Batak), advokat, jamboo mentega, jamboo pooan, pookat (Lampung) dan lain-lain.(TTIG Budaya Pertanian). Klasifikasi lengkap tanaman alpukat adalah sebagai berikut:

Divisi : Spermatophyta

Anak divisi : Angiospermae

Kelas : Dicotyledoneae

Bangsa : Ranales

Keluarga : Lauraceae

Marga : Persea

Varietas : Persea americana Mill

Banyak ragam jenis-jenis alpukat yang tersebar di Indonesia, sampai tahun ini telah dilepas 7 varietas alpukat sebagai berikut :

1. Alpukat Ijo Bundar

   Alpukat ini mempunyai permukaan kulit licin, bentuk buah lonjong, kulitnya hijau muda dan berangsur tua saat matang.

2. Alpukat Ijo Panjang

   Varietas ini bentuk buahnya menyerupai buah pir. Kulit buah berwarna hijau dan setelah matang menjadi hijau tua merah.

3. Alpukat Merah Bundar

   Varietas ini berbuah terus menerus. Buah muda kulitnya merah coklat.

4. Alpukat Merah Panjang

   Varietas ini bentuk buahnya menyerupai buah pir. Saat muda kulitnya hijau merah coklat dan setelah matang menjadi merah hitam.

5. Alpukat Mega Gagauan

   Alpukat jenis ini memiliki produksi tinggi, bentuk buah bulat permukaan agak halus dan kulit buah kemerahan.

6. Alpukat Mega Murapi

   Bentuk buah lonjong dan permukaan kulit kasar, warna kulit buah hijau tua.

7. Alpukat Mega Paninggahan

   Bentuk buah bulat lonjong, permukaan kulit halus, warna kulit buah merah maroon.

B. Pengertian Sistem

Sistem adalah suatu kesatuan utuh yang terdiri dari beberapa bagian yang saling berhubungan dan berinteraksi untuk mencapai tujuan tertentu. (Teguh Wahyono, 2004)

Menurut Zulkifli (1997), Sistem adalah himpunan suatu benda nyata atau abstrak yang terdiri dari bagian-bagian atau komponen-komponen yang saling berkaitan, berhubungan, berketergantungan, dan saling mendukung, yang secara keseluruhan bersatu dalam satu kesatuan untuk mencapai tujuan tertentu secara efisien dan efektif.

Menyangkut pengertian tentang sistem menurut Jogiyanto (2009) mengemukakan sebagai kumpulan dari komponen yang saling berhubungan satu dengan yang lainnya membentuk satu kesatuan untuk mencapai tujuan tertentu.

Dari beberapa kutipan diatas penulis dapat menarik kesimpulan bahwa sistem adalah jaringan kerja dengan segala aktifitas yang saling terkait yang dilakukan oleh objek yang saling berhubungan dalam suatu wadah yang sama untuk mencapai suatu tujuan atau sasaran yang telah ditentukan.

C. Kecerdasan Buatan

Kecerdasan buatan (*artificial intelligent*) adalah salah satu bidang ilmu komputer yang mendayagunakan komputer sehingga dapat berprilaku seperti manusia. Ilmu komputer tersebut mengembangkan perangkat lunak dan perangkat keras untuk menirukan tindakan manusia. Aktifitas manusia yang ditirukan seperti penalaran, penglihatan, pembelajaran, pemecahan masalah,

pemahaman bahasa alami dan sebagainya. Rich dan knight mendefenisikan kecerdasan buatan sebagai sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia (Kusrini, 2006).

Kecerdasan bautan merupakan bagian dari ilmu pengetahuan komputer yang khusus ditunjukkan dalam perancangan otomatisasi tingkah laku cerdas dalam sistem kecerdasan komputer. Sistem memperlihatkan sifat-sifat khas yang dihubungkan dengan kecerdasan dalam kelakuan atau tindak tanduk yang sepenuhnya bisa menirukan beberapa fungsi otak manusia, seperti pengertian bahasa, pengetahuan, pemikiran, pemecahan masalah dan lain sebagainya (Kristanto, 2004). Kecerdasan buatan berbeda dengan kecerdasan konvensional. Pemrograman konvensional berbasis pada algoritma yang mendefinisikan setiap langkah dalam penyelesaian masalah. Pemrograman konvensional dapat menggunakan rumus matematika atau prosedur sekuensial untuk menghasilkan solusi. Lain halnya dengan pemrograman dalam kecerdasan buatan yang berbasis pada representasi simbol dan manipulasi. Dalam kecerdasan buatan, sebuah simbol dapat berupa kalimat, kata, atau angka yang digunakan untuk merepresentasikan objek, proses, dan hubungannya. Objek dapat berupa manusia, benda, ide, konsep, kegiatan, atau pernyataan dari suatu fakta (Kusrini, 2006). Proses digunakan untuk memanipulasi simbol untuk menghasilkan saran atau pemecahan masalah. Selain itu kecerdasan buatan dapat melakukan penalaran terhadap data yang tidak lengkap. Hal ini sangat mustahil dilakukan oleh pemrograman konvensional. Kemampuan penalaran dan penjelasan terhadap setiap langkah

dalam pengambilan keputusan menjadi kelebihan dari kecerdasan buatan. Ada tiga tujuan kecerdasan buatan, yaitu: membuat komputer lebih cerdas, mengerti tentang kecerdasan dan membuat mesin lebih berguna.

D. Sistem Pakar

1. Definisi

Sistem pakar (*expert system*) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer yang dirancang untuk menyelesaikan masalah seperti layaknya seorang pakar. Keahlian dipindahkan dari pakar ke suatu komputer. Pengetahuan ini kemudian disimpan di dalam komputer. Pada saat pengguna menjalankan komputer untuk mendapatkan informasi, maka sistem pakar akan menanyakan fakta-fakta dan dapat membuat penalaran (inferensi) sehingga sampai pada suatu kesimpulan. Kemudian, sistem pakar memberikan penjelasan dan kesimpulan atas hasil konsultasi yang telah dilakukan sebelumnya (Turban, 2005). Dengan sistem pakar ini, orang awam pun dapat menyelesaikan masalahnya atau hanya sekedar mencari suatu informasi berkualitas yang sebenarnya hanya dapat diperoleh dengan bantuan para ahli dibidangnya. Sistem pakar ini juga dapat membantu aktivitas para pakar sebagai asisten yang mempunyai pengetahuan yang dibutuhkan.

Sistem pakar adalah sistem berbasis komputer yang menggunakan pengetahuan, fakta, dan teknik penalaran dalam memecahkan masalah yang biasanya hanya dapat dipecahkan oleh seorang pakar dalam bidang tersebut (Kusrini, 2006).

Pemecahan masalah-masalah yang komplek biasanya hanya dapat dilakukan oleh sejumlah orang yang sangat terlatih, yaitu pakar. Dengan penerapan teknik kederdasan buatan, sistem pakar menirukan apa yang dikerjakan oleh seorang pakar ketika mengatasi permasalahan yang rumit, berdasarkan pengetahuan yang dimiliki (Hartati *dan Iswanti*, 2008).

2. Sejarah Sistem Pakar

Sistem pakar mulai dikembangkan pada pertengahan tahun 1960-an oleh komunitas AI. Periode peneliatan kecerdasan buatan ini didominasi olah suatu keyakinan bahwa nalar yang digabung dengan komputer cangih akan menghasilkan prestasi pakar atau manusia pakar. Sistem pakar yang pertama kali muncul adalah *General-purpose Problem Solver* (GPS) yang dikembangkan oleh Newel dan Simon. GPS (dan program-program yang serupa) ini mengalami kegagalan karena cakupannya terlalu luas sehingga terkadang justru meninggalkan pengetahuan-pengetahuan penting yang seharusnya disediakan (Kusrini, 2006).

Pada pertengahan tahun 1960-an, terjadi pergantian dari program serba bisa (*general-purpose*) ke program yang spesial (*special-purpose*) dengan dikembangkannya DENDRAL oleh E. Feigenbaum dari Universitas Stanford dan kemudian diikuti oleh MYCIN.

Pada pertengahan tahun 1970-an, beberapa *expert system* mulai muncul. Sebuah pengetahuan kunci yang dipelajari saat itu adalah kekuatan dari *expert system* berasal dari pengetahuan spesifik yang

dimilikinya, bukan dari formalisme-formalisme khusus dan pola penarikan kesimpulan yang digunakannya.

Awal 1980-an teknologi *expert system* yang mula-mula dibatasi oleh suasana akademis mulai muncul sebagai aplikasi komersial, khususnya XCON, XSEL (dikembangkan dari R-1 pada *Digital Equiptment Corp*.), dan CATS-1 (dikembangkan oleh *General Electric).*

Sistem pakar untuk melakukan diagnosis kesehatan telah dikembangkan sejak pertengahan tahun 1970. Sistem pakar untuk melakukan diagnosis pertama dibuat oleh Bruce Buchanan dan Edward Shortliffe di Stanford University. Sistem ini diberi nama MYCIN.

MYCIN merupakan program interaktif yang melakukan diagnosis penyakit meningitis dan infeksi bacremia serta memberikan rekomendasi terapi antimikroba. MYCIN mampu memberikan penjelasan atas penalarannya secara detail. Dalam uji coba, MYCIN mampu menunjukkan kemampuan seperti seorang spesialis. Meskipun MYCIN tidak pernah digunakan secara rutin oleh dokter, MYCIN merupakan inferensi yang bagus dalam kecerdasan buatan yang lain (Kusrini, 2006).

3. Konsep Dasar Sistem Pakar

Menurut Turban (1995), konsep dasar sistem pakar mengandung keahlian, ahli, pengalihan keahlian, inferensi, aturan, dan kemampuan menjelaskan. Keahlian adalah suatu kelebihan penguasaan pengetahuan di bidang tertentu yang diperoleh dari pelatihan, membaca, atau pengalaman.

Pengalihan keahlian dari para ahli ke komputer untuk kemudian dialihkan lagi ke orang lain yang bukan ahli merupakan tujuan utama dari sistem pakar. Proses ini membutuhkan empat aktivitas, yaitu: tambahan pengetahuan (dari para ahli atau sumber-sumber lainnya), representasi pengetahuan (ke komputer), inferensi pengetahuan, dan pengalihan pengetahuan ke *user*. Pengetahuan yang disimpan di komputer disebut dengan nama basis pengetahuan. Ada dua tipe basis pengetahuan, yaitu: fakta dan prosedur (biasanya berupa aturan).

Salah satu fitur yang harus dimiliki oleh sistem pakar adalah kemampuan untuk menalar. Jika keahlian-keahlian sudah tersimpan sebagai basis pengetahuan dan sudah tersedia program yang mampu mengakses basis data.

Sebagian besar sistem pakar komersial dibuat dalam bentuk *rule-based systems*, yang mana pengetahuan disimpan dalam bentuk aturan-aturan. Aturan tersebut biasanya berbentuk *IF-THEN* (Kusumadewi, 2003).

E. Metode *Forward Chaining*

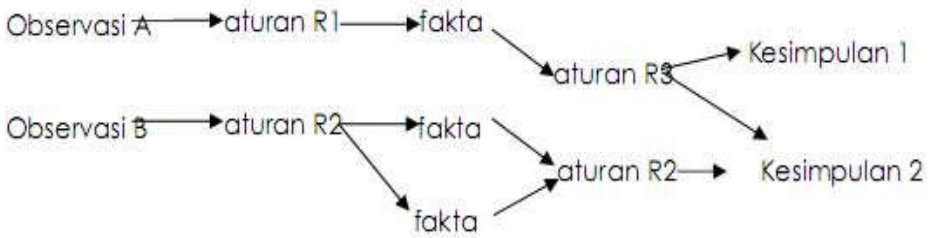
Forward chaining merupakan metode inferensi yang melakukan penalaran dari suatu masalah kepada solusinya. Jika klausa premis sesuai dengan situasi (bernilai *TRUE*), maka proses akan menyatakan konklusi. *Forward chaining* adalah data-driven karena inferensi dimulai dengan informasi yang tersedia dan baru konklusi diperoleh. Jika suatu aplikasi

menghasilkan tree yang lebar dan tidak dalam, maka gunakan forward chaining. (Sumber: Efraim Turban, 2005).*Forward Chaining* digunakan jika:

1. Banyak aturan berbeda yang dapat memberikan kesimpulan yang sama.
2. Banyak cara untuk mendapatkan sedikit konklusi.
3. Benar-benar sudah mendapatkan pelbagai fakta, dan ingin mendapatkan konklusi dari fakta-fakta tersebut.

Adapun tipe sistem yang dapat menggunakan teknik pelacakan forward chaining, yakni :

1. Sistem yang direpresentasikan dengan satu atau beberapa kondisi.
2. Untuk setiap kondisi, sistem mencari rule-rule dalam knowledge base untuk rule-rule yang berkorespondensi dengan kondisi dalam bagian if.
3. Setiap rule dapat menghasilkan kondisi baru dari konklusi yang diminta pada bagian then. Kondisi baru ini dapat ditambahkan ke kondisi lain yang sudah ada.
4. Setiap kondisi yang ditambahkan ke sistem akan diproses. Jika ditemui suatu kondisi, sistem akan kembali ke langkah 2 dan mencari rule-rule dalam knowledge base kembali. Jika tidak ada konklusi baru, sesi ini berakhir .



Gambar 2.1 Diagram *Forward Chaining*

F. *Framework*

Framework berasal dari bahasa inggris yang artinya adalah bingkai kerja. Framework dalam sistem berorientasi objek, merupakan kumpulan class yang melambangkan bentuk abstrak untuk pemecahan sejumlah masalah yang berhubungan. Sehingga bisa mempermudah dan mempercepat pekerjaan seorang programmer, tanpa harus membuat fungsi atau class dari awal. Jadi, dengan adanya framework, pekerjaan akan lebih tertata dan terorganisir. Sehingga dalam pencarian kesalahan dalam pembuatan program akan lebih mudah dideteksi. Intinya, *framework* merupakan pondasi awal sebelum menentukan memakai bahasa pemrograman apa yang akan dipakai.

1. Beberapa kelebihan *framework* adalah Penggunaan skrip yang telah dibuat, dites dan digunakan oleh programmer lain. serta dalam sisi fungsional dan keamanan framework sudah memiliki jaminan. *Framework* juga sudah menggunakan pola perancangan MVC yang digunakan untuk pemecahan masalah modularitas untuk perangkat lunak berbasis web.
2. Kekurangan *framework* adalah adanya kemungkinan batasan-batasan ketika merancang aplikasi menggunakan *framework*. Menambah biaya development apabila *framework* yang digunakan kurang terdokumentasi dan kurang disupport.

G. *Laravel*

Sejak tahun 2012 muncul satu fenomena yang cukup berbeda dan menarik perhatian, dimana ada satu *framework* yang membawa ideologi baru

yang selama ini jarang diperhatikan, yaitu aspek "*clean code*" dan "*expressiveness*". *Framework* ini mengaku "*clean and classy*", kodenya lebih singkat, mudah dimengerti, dan ekspressif, jadi hanya dengan membaca sekilas kode yang ditulis Anda sudah bisa menduga apa maksudnya tanpa perlu membaca dokumentasi. *Framework* ini dinamakan *LARAVEL*. *Framework* ini juga bisa menggunakan *composer*. *Composer* adalah sebuah '*dependency manager'* untuk *PHP*. Anda bisa menginstall suatu *library* melalui *composer* dan *composer* akan secara otomatis menginstall *library* lain yang dibutuhkan, tanpa perlu mendownload satu persatu. Mirip dengan *apt get install* di sistem operasi *linux*. Dan seluruh *library* yang Anda butuhkan akan otomatis didownload dan siap digunakan.

Laravel adalah *framework MVC* pengembangan *web* yang ditulis menggunakan bahasa pemrograman *PHP*. *Laravel* telah dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya, baik biaya pengembangan awal dan biaya pemeliharaan, serta memberikan sintaks ekspresif yang jelas dan set fungsi inti yang akan menghemat waktu pengerjaan.

Laravel adalah salah satu dari beberapa kerangka bahasa pemrograman PHP yang menawarkan *code modular*. Hal ini dicapai melalui kombinasi *driver* dan sistem *bundle*-nya. *Driver* memungkinkan kita untuk dengan mudah mengubah dan memperluas *caching*, *session*, *database*, dan fungsi otentikasi. Penggunaan *bundle* mampu mengemas hingga segala jenis kode untuk digunakan kembali atau untuk memberikan kepada seluruh pengguna

Laravel. *Laravel* sangat menarik, karena apapun yang ditulis dalam *Laravel* dapat dikemas dalam sebuah kemasan (McCool, 2012).

H. *Waterfall*

Pada penulisan tugas akhir ini, penulis menggunakan model pengembangan perangkat lunak *waterfall*. Menurut Rosa dan Shalahuddin (2015:25) "Pada awal pengembangan perangkat lunak, para pembuat program (programmer) langsung melakukan pengkodean perangkat lunak tanpa menggunakan prosedur atau tahapan pengembangan perangkat lunak".

Menurut Rosa dan Shalahuddin (2015:28) "Model *SDLC* air terjun (*waterfall*) sering juga disebut model sekuensial linear (*sequential linear*)atau alur hidup klasik(classic life cycle). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dari analisis desain, pengkodean, pengujian, dan tahap pendukung(support)".

Menurut Rosa dan Shalahuddin (2015:29)Metode yang digunakan pada pengembangan perangkat lunak ini menggunakan model waterfall (Rosa dan Shalahuddin, 2015:29)yang terbagi dalam lima tahapan, yaitu:

1. Analisa kebutuhan perangkat lunak

   Analisa ini bertujuan untuk mengumpulkan data-data yang berkaitan dengan sistem penjualan dan pembelian. Analisa kebutuhan terdiri dari analisa kebutuhan fungsional (fungsi sistem) dan analisa kebutuhan non-fungsional (pengguna sistem dan alat yang diperlukan dalam perancangan sistem).Tujuan dari analisa ini untuk mendapatkan

informasi dasar seputar sistem yang diterapkan dan digunakan sebagai dasar dalam perancangan sistem.

2. Desain

   Setelah mendapatkan data-data dari analisa, maka masuk pada tahap desain. Penulis merancang sistem yang terdiri dari rancangan basis data, rancanganHierarchy Input Process Output (HIPO) dan pemodelan antara muka.

3. Pembuatan kode program

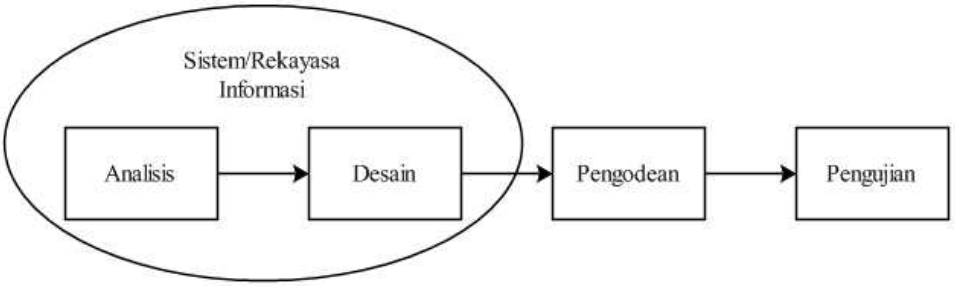
   Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain yang ditranslasikan kedalam program perangkat lunak.

4. Pengujian

   Untuk meminimalisir kesalahan (error) dan keluaran yang dihasilkan sesuai dengan yang diinginkan. Maka pengujian difokuskan pada perangkat lunak secara segi logik dan fungsioanal memastikan bahwa semua bagiansudah diuji.

5. Pendukung(support)atau pemeliharaan(maintenance)

   Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tetapi tidak untuk membuat perangkat lunak yang baru. Agar tidak mengalami perubahan ketika sudah dikirimkan keuser karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkap lunak harus beradaptasi dengan lingkungan baru.



Gambar 2.1.Ilustrasi Model Waterfall

Sumber: Rosadan Shalahuddin (2015:29)

I. *Web*

1. Definisi *Web*

Web adalah salah satu aplikasi yang berisikan dokumen-dokumen multimedia (teks, gambar, suara, animasi, video) di dalamnya yang menggunakan protokol *HTTP* (*hypertext transfer protocol*) dan untuk mengaksesnya mengggunakan perangkat lunak yang disebut *browser*.(Rudianto, 2011).

Browser (perambah) adalah aplikasi yang mampu menjalankan dokumen-dokumen *web* dengan cara diterjemahkan. Prosesnya dilakukan oleh komponen yang terdapat di dalam aplikasi *browser* yang biasa disebut *web engine*. Semua dokumen *web* ditampilkan oleh *browser* dengan cara diterjemahkan.

Situs *Web* adalah dokumen-dokumen *web* yang terkumpul menjadi satu kesatuan yang memiliki *Unified Resource Locator* (URL atau *domain* dan biasanya di-*publish* di *internet* atau *intranet*. (Rudianto. 2011).

2. Definisi Situs *Web*

Situs *web* merupakan kumpulan dari halaman *web* yang sudah dipublikasikan di jaringan *internet* dan memiliki domain atau URL (*Uniform Resource Locator*) yang dapat diakses semua pengguna *internet* dengan cara mengetikkan alamatnya. Berikut adalah contoh alamat situs *web*: www.amiko.ac.id, http://rudyantoarief.com. (Rudianto. 2011).

J. *PHP* (*Hypertext Prepocessor*)

1. Definisi *PHP* (*Hypertext Prepocessor*)

PHP adalah bahasa pemrograman *script* yang paling banyak dipakai saat ini. *PHP* banyak dipakai untuk memrogram situs *web* dinamis, walaupun tidak tertutup kemungkinan digunakan untuk pemakaian lain.Contoh terkenal dari aplikasi *PHP* adalah forum (*phpBB*) dan *Media Wiki* (*software* di belakang *Wikipedia*). *PHP* juga dapat dilihat sebagai pilihan lain dari *ASP.NET/C#/VB.NET Microsoft, ColdFusion Macromedia, JSP/Java Sun Microsystems, dan CGI/Perl.* Contoh aplikasi lain yang lebih kompleks berupa *CMS* yang dibangun menggunakan *PHP* adalah *Mambo, Joomla*, *Postnuke, Xaraya*, dan lain-lain.(Rudianto. 2011).

2. Kelebihan *PHP*(*Hypertext Prepocessor*)

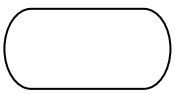
1. Bahasa pemrograman *PHP a*dalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaanya.

2. *Web Server* yang mendukung PHP dapat ditemukan dimana - mana dari mulai *apache*, *IIS*, *Lighttpd*, hingga *Xitami* dengan konfigurasi yang relatif mudah.
3. *PHP* adalah bahasa *open source* yang dapat digunakan di berbagai mesin (*Linux, Unix, Macintosh, Windows*) dan dapat dijalankan secara runtime melalui console serta juga dapat menjalankan perintah-perintah sistem.

K. *Flowchart*

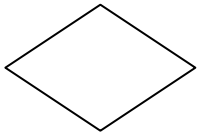
Flowchart dapat menunjukkan secara jelas arus pengendalian suatu *algoritma*, yakni melaksanakan suatu rangkaian kegiatan secara *logis* dan *sistematis*. Suatu diagram alur dapat memberi gambaran dua *dimensi* berupa simbol-simbol grafis. Masing-masing simbol telah ditetapkan lebih dahulu fungsi dan artinya. Simbol-simbol tersebut dipakai untuk menunjukkan berbagai kegiatan operasi dan jalur pengendalian. Arti khusus dari sebuah *flowchart* adalah simbol-simbol yang digunakan untuk menggambarkan urutan proses yang terjadi di dalam suatu program komputer secara *sistematis* dan *logis*. (Sutabri, 2004).

Tabel 2.1. Simbol *Flowchart*

Simbol *Flowchart*	Fungsi
	*TERMINAL* Simbol ini digunakan untuk mengawali atau mengakhiri suatu proses/kegiatan.

PREPARATION

Simbol ini digunakan untuk mempersiapkan harga awal/nilai awal suatu variabel yang akan diproses.



DECISION

Simbol ini digunakan untuk pengujian suatu kondisi yang sedang diproses.

PROSES

Simbol ini digunakan untukmenggambarkan suatu proses yang sedang dieksekusi.

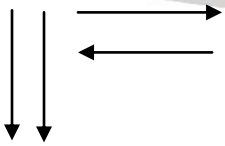
INPUT/OUTPUT

Simbol ini digunakan untuk menggambarkan proses input (*read*) maupun proses *output* (*print*).

SUBROUTINE

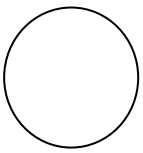
Simbol ini digunakan untuk menggambarkan proses pemanggilan subprogram dari main program.

FLOW LINE



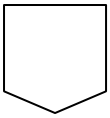
Simbol ini digunakan untuk menggambarakan arus proses dari suatu kegiatan ke kegiatan lain.

CONECTOR



Simbol ini digunakan sebagai penghubung antara suatu proses dengan proses lainnya yang

ada di dalam suatu lembar halaman.



PAGE CONECTOR

Simbol ini digunakan sebagai penghubung antara suatu proses dengan proses lainnya, tetapi berpindah halaman.

MANUAL OPERATION

Simbol ini digunakan untuk menggambarkan suatu kegiatan atau proses yang bersifat manualisasi.

PRINTER

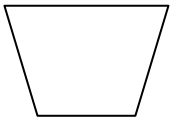
Digunakan untuk menggambarkan suatu kegiatan mencetak suatu informasi dengan mesin printer.

CONSOLE

Simbol ini digunakan untuk menggambarkan suatu kegiatan menampilkan data atau informasi melalui monitor atau *CRT* (*Cathode Ray Tube*).

DISK

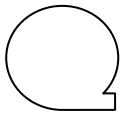
Simbol ini digunakan untuk menggambarkan suatu kegiatan membaca atau menulis data menggunakan media *magnetic disk*.



MANUAL INPUT

Simbol ini digunakan untuk menggambarkan

proses pemasukan data melalui media keyboard.



TAPE

Simbol ini digunakan untuk menggambarkan suatu kegiatan membaca atau menulis data menggunakan media *magnetic tape*.

Sumber : Sutabri. 2004

1. Jenis *flowchart*.

   Bentuk diagram alur (*flowchart*) yang sering digunakan dalam proses pembuatan suatu program komputer adalah sebagai berikut:

   a. Program *flowchart*.

      Simbol-simbol yang menggambarkan proses secara rinci dan detil antara intruksi yang satu dengan intruksi yang lainnya dalam suatu program komputer yang bersifat *logic*.

   b. Sistem *flowchart*.

      Simbol-simbol yang menggambarkan urutan prosedur secara detil dalam suatu sistem komputerisasi. Bersifat fisik.

2. Teknik pembuatan *flowchart*.

   Sebelum kita membuat sebuah program komputer, yang harus kita lakukan sebelumnya adalah membuat *flowchart*. Jenis *flowchart* yang sering digunakan adalah program *flowchart*.

   Teknik pembuatan program *flowchart* ini dibagi menjadi dua bagian, yaitu:

a. *General way.*

Teknik pembuatan *flowchart* dengan cara ini lazim digunakan untuk menyusun logika suatu program. Teknik ini menggunakan pengulangan proses secara tidak langsung (*Non-Direct-Loop*).

b. *Iteration way.*

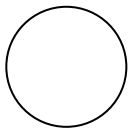
Teknik pembuatan *flowchart* dengan cara ini biasanya dipakai untuk logika program yang cepat dan bentuk permasalahannya kompleks. Pengulangan proses yang terjadi bersifat langsung (*Direct-Loop*). (Sutabri, 2004).

L. *Data Flow Diagram* (*DFD*)

Data Flow Diagram(DFD) adalah representasi grafik dari sebuah sistem. *DFD* menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut. Kita dapat menggunakan *DFD* untuk dua hal utama, yaitu untuk membuat dokumentasi dari sistem informasi yang ada, atau untuk menyusun dokumentasi untuk sistem informasi yang baru. Empat simbol yang digunakan : (Sutabri, 2004)

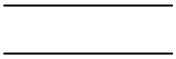
Tabel 2.2. Simbol *Data Flow Diagram* (*DFD)*

Notasi	Fungsi
	Simbol *entitaseksternal* atau *terminator* menggambarkan asal atau tujuan data di luar sistem



Simbollingkaran menggambarkan entitas atau proses dimana aliran data masuk ditransformasikan ke aliran data keluar

Simbol aliran data menggambarkan aliran data



Simbol *file* menggambarkan tempat data disimpan

Jenis-jenis *DFD*:

1. *Diagram contex.*

   Jenis pertama *Context diagram*, adalah data *flow* diagram tingkat atas (*DFD Top Level*), yaitu diagram yang paling tidak detail, dari sebuah sistem informasi yang menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar *entitas-entitas eksternal*. (*CD* menggambarkan sistem dalam satu lingkaran dan hubungan dengan *entitas* luar. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem). Beberapa hal yang harus diperhatikan dalam menggambar *CD* :

   a. Terminologi sistem :

      1) Batas sistem adalah batas antara daerah kepentingan sistem.
      2) Lingkungan sistem adalah segala sesuatu yang berhubungan atau mempengaruhi sistem tersebut.
      3) *Interface* adalah aliran yang menghubungkan sebuah sistem dengan linkungan sistem tersebut.

b. Menggunakan satu simbol proses.

Yang masuk didalam lingkaran konteks (simbol proses) adalah kegiatan pemrosesan informasi (Batas Sistem). Kegiatan informasi adalah mengambil data dari *file*, *mentransformasikan* data, atau melakukan *filing* data, misalnya mempersiapkan dokumen, memasukkan, memeriksa, mengklasifikasi, mengatur, menyortir, menghitung, meringkas data, dan melakukan *filing* data (baik yang melakukan secara manual maupun yang dilakukan secara *terotomasi*).

c. Nama/keterangan di simbol proses tersebut sesuai dengan fungsi sistem tersebut.

d. Antara *Entitas Eksternal/Terminator* tidak diperbolehkan komunikasi langsung.

e. Jika terdapat *termintor* yang mempunyai banyak masukan dan keluaran, diperbolehkan untuk digambarkan lebih dari satu sehingga mencegah penggambaran yang terlalu rumit, dengan memberikan tanda *asterik* (*) atau garis silang (#).

f. Jika Terminator mewakili individu (personil) sebaiknya diwakili oleh peran yang dipermainkan *personil* tersebut.

g. Aliran data ke proses dan keluar sebagai output keterangan aliran data berbeda.

2. *DFD* fisik.

DFD fisik adalah representasi grafik dari sebuah sistem yang menunjukan *entitas-entitasinternal* dan *eksternal* dari sistem tersebut,

dan aliran-aliran data ke dalam dan keluar dari *entitas-entitas* tersebut. *Entitas-entitas internal* adalah *personal*, tempat (sebuah bagian), atau mesin (misalnya, sebuah komputer) dalam sistem tersebut yang *mentransformasikan* data. Maka *DFD* fisik tidak menunjukkan apa yang dilakukan, tetapi menunjukkan dimana, bagaimana, dan oleh siapa proses-proses dalam sebuah sistem dilakukan.

Perlu diperhatikan didalam memberikan keterangan di lingkaran-lingkaran (simbol proses) dan aliran-aliran data (simbol aliran data) dalam *DFD* fisik menggunakan label/keterangan dari kata benda untuk menunjukan bagaimana sistem *mentransmisikan* data antara lingkaran-lingkaran tersebut.

3. *DFD* logis.

DFD logis dalah representasi grafik dari sebuah sistem yang menunjukkan proses-proses dalam sistem tersebut dan aliran-aliran data ke dalam dan ke luar dari proses-proses tersebut. Kita menggunakan *DFD* logis untuk membuat dokumentasi sebuah sistem informasi karena *DFD* logis dapat mewakili logika tersebut, yaitu apa yang dilakukan oleh sistem tersebut, tanpa perlu menspesifikasi dimana, bagaimana, dan oleh siapa proses-proses dalam sistem tersebut dilakukan. Keuntungan dari *DFD* logis dibandingkan dengan *DFD* fisik adalah dapat memusatkan perhatian pada fungsi - fungsi yang dilakukan sistem.

I. *Entity Relationship Diagram (ERD)*

Model *entity-relationship* yang berisi komponen-komponen. Himpunan *entitas* dan himpunan relasi yang masing-masing delengkapi dengan atribut-atribut yang mempresentasikan seluruh fakta dari dunia nyata yang kita tinjau, dapat digambarkan dengan lebih sistematis dengan menggunakan diagram *entity-relationship* (diagram *E-R*). Notasi-notasi simbolik didalam diagram *E-R* yang dapat kita gunakan adalah :

Gambar 2.2 Kardinalitas relasi

Berikut adalah contoh penggambaran relasi antar himpunan *entitas* lengkap dengan kardinalitas relasi dan atribut-atributnya :

1. Relasi satu-ke-satu *(one-to-one)*
2. Relasi satu-kebanyak (*one-to-many*).
3. Relasi banyak-ke-banyak (*many-to-many*).

J. Basis Data

1. Pengertian Basis Data

Basis dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul. Sedangkan data merupakan representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa,

pembeli, pelanggan), barang, hewan peristiwa, konsep, keadaan, dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya. Basis data (*database*) merupakan kumpulan data yang saling berhubungan (punya relasi). (Yakub, 2012). Menurut Janner (2007), Basis Data adalah koleksi data yang bisa mencari secara menyeluruh dan secara sistematis memelihara dan me-*retrieve* informasi.

2. Manfaat Basis Data

a. Kecepatan dan kemudahan (*Speed*), pemanfaatan basis data memungkinkan untuk dapat, menyimpan, merubah, dan menampilkan kembali data tersebut dengan lebih cepat dan mudah.

b. Efisiensi ruang penyimpanan (*space*), dengan basis data efisiensi atau optimalisasi penggunaan ruang penyimpanan dapat dilakukan, karena penekanan jumlah redudansi data, baik dengan sejumlah pengkodean atau dengan membuat label-tabel yang saling berhubungan.

c. Keakuratan (*accuracy*), pembentukan relasi antar data bersama dengan penerapan aturan atau batasan (*constraint*) tipe, domain dan keunikan data dapat diterapkan dalam sebuah basis data.

d. Ketersediaan (*availability*), dapat memilah data utama atau master, transaksi, data histori hingga data kadaluwarsa. Data yang jarang atau tidak digunakan lagi dapat diatur dari sistem basis data yang akif.

e. Keamanan (*security*), untuk menentukan siapa-siapa yang berhak menggunakan basis data beserta objek-objek di dalamnya dan menentukan jenis-jenis operasi apa saja yang boleh dilakukan.

f. Kebersamaan pemakai (*sharebility*), basis data dapat digunakan oleh beberapa pemakai dan beberapa lokasi. Basis Data yang dikelola oleh sistem (aplikasi) yang mendukung *multiuser* dapat memenuhi kebutuhan, akan tetapi harus menghindari inkonsistensi data. (Yakub, 2012).

3. Operasi Basis Data

Pada sebuah *disk* (*harddisk*), basis dapat diciptakan dapat pula ditiadakan. Pada sebuah *disk* juga dapat menempatkan beberapa basis data, misalnya basis data kepegawaian, akademik, penjualan, perpustakaan dan lain-lain. Sementara dalam sebuah basis data dapat ditempatkan pada satu *file* atau tabel barang, faktur, pelanggan dan transaksi barang. Operasi-operasi dasar yang dapat dilakukan basis data adalah :

a. Pembuatan basis data baru (*Create Database*)

b. Penghapusan basis data (*Drop Database*)

c. Pembuatan file atau tabel baru ke suatu basis data (*Create Table*)

d. Penghapusan file atau tabel dari suatu basis data (*Drop Table*)

e. Penambahan atau pengisian data baru di sebuah basis data (*Insert)*

f. Pengambilan data dari sebuah file atau tabel (*Retrieve* atau *Search*)

g. Pengubahan data dalam sebuah file atau tabel (*Update*)

h. Penghapusan data dari sebuah file atau tabel (*Delete*)

Operasi pembuatan basis data dan tabel merupakan operasi awal yang hanya dilakukan sekali dan berlaku seterusnya. Sedangkan untuk operasi pengisian data merupakan operasi rutin yang dilakukan berulang ulang. (Yakub, 2012)

4. Persyaratan Basis Data

Ketentuan yang harus diperhatikan pada pembuatan *file* basis data agar dapat memenuhi kriteria sebagai basis data, yaitu: *redudansi* data, inkonsistensi data, pengaksesan data, data terisolasi untuk standarisasi, masalah keamanan, masalah integritas data, data *multiuser*.

a. *Redudansi* dan Inkonsistensi Data

Penyimpanan data yang sama dibeberapa tempat disebut *redudansi*, hal ini akan menyebabkan pemborosan dan menimbulkan inkonsistensi data (data tidak konsisten) karena bila terjadi maka data harus dirubah pada beberapa tempat, hal ini tentunya tidak efisien.

b. Pengaksesan Data

Data di dalam basis data harus siap diakses oleh siapa saja yang membutuhkan dan mempunyai hak untuk mengaksesnya. Oleh karena itu perlu dibuat suatu program pengelolaan atau suatu aplikasi untuk mengakses data yang dikenal sebagai *Database Management System* (*DBMS*).

c. Data Terisolasi untuk Standarisasi

Jika data tersebar dalam beberapa *file* dalam bentuk *format* yang tidak sama, makan akan menyulitkan dalam menulis program

aplikasi, baik untuk mengambil dan menyimpan data. Oleh karena itu ada dalam satu *database* harus dibuat satu *format* yang sama, sehingga mudah dibuat program aplikasinya.

d. Masalah Keamanan atau *Security*

Setiap pemakai sistem basis data tidak semua bagian diperbolehkan untuk mengakses semua data, misalnya data mengenai gaji pegawai hanya boleh dibuka oleh bagian keuangan, sedang bagian gudang dan bagian lain tidak diperkenankan untuk membukanya. Keamanan dapat diatur dan disesuaikan baik ditingkat basis data atau aplikasinya.

e. *Multiple User*

Salah satu alasan basis data dibangun karena nantinya data tersebut akan digunakan oleh banyak orang, baik dalam waktu berbeda maupun bersamaan. Oleh karena itu diperlukan basis data yang handal dan dapat mendukung banyak pemakai atau multiuser. (Yakub, 2012).

K. *Behavioral (Black-Box) Tests*

Menurut Black (2009 :3), Tester menggunakan behavioral test (disebut juga Black-Box Tests), sering digunakan untuk menemukan bug dalam high level operations, pada tingkatan fitur, profil operasional dan skenario customer. Tester dapat membuat pengujian fungsional black box berdasarkan pada apa yang harus sistem lakukan. Behavioral testing melibatkan pemahaman rinci mengenai domain aplikasi, masalah bisnis yang dipecahkan

oleh sistem dan misi yang dilakukan sistem. Behavioral test paling baik dilakukan oleh penguji yang memahami desain sistem, setidaknya pada tingkat yang tinggi sehingga mereka dapat secara efektif menemukan bug umum untuk jenis desain.

Menurut Nidhra dan Dondeti (2012:1), black box testing juga disebut functional testing, sebuah teknik pengujian fungsional yang merancang test case berdasarkan informasi dari spesifikasi.