BAB 2

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1. Tinjauan Pustaka

Penelitian tentang sistem pendukung keputusan telah dilakukan Heni Ayu Septilia dan Styawati dengan judul "Sistem Pendukung Keputusan Pemberian Dana Bantuan Menggunakan Metode AHP".Sistem pendukung keputusan berfungsi mendukung dalam penentuan bantuan untuk masyarakat. Kriteria yang digunakan dalam pemberian dana PKH yaitu pendidikan, pekerjaan, penghasilan, *status*,umur, tempat tinggal, kesehatan, dan jumlah anak[4].

Penelitian lainnnya dilakukan oleh Karuna Sindu Dasa, I Putu Satwika dan Ketut Queena Fredlina dengan judul "Sistem Pendukung Keputusan Pemberian Beasiswa PPA di STMIK Primakara Menggunakan Metode *Weighted Product*".Pemberian beasiswa adalah pemberian berupa bantuan keuangan yang diberikan kepada perorangan yang bertujuan untuk digunakan demi keberlangsungan pendidikan yang ditempuh[5].

Mustika Intan Suri dan Ajeng Savitri Puspaningrum juga melakukan peneilitan dengan judul "Sistem In*form*asi Manajemen Berita Berbasis WEB".Sistem ini berbasis *web* karena agar dapat dijalankan dari jarak jauh menggunakan internet dan sebagai solusi manajamen berita yang efektif, yang dapat digunakan untuk mengirim berita oleh reporter, memeriksa berita oleh koordinator liputan/redaktur, kemudian digunakan untuk menyimpan serta mengelola *data* berita oleh *staff* liputan berita dan dokumen (libradok) yang nantinya akan dijadikan laporan bulanan serta memastikan bahwa aplikasi tersebut dapat diimplementasikan dengan melakukan pengujian fungsionalitas terhadap sistem di kantor LPP RRI Bandar Lampung[6].

Penelitian lainnya dilakukan oleh Petricia Oktavia dengan judul "Sistem Pendukung Keputusan Seleksi Penerima Beasiswa Dengan Metode Weighted Poduct Pada SMP Negeri 1 Parung Berbasis WEB" sistem ini menggunakan metode *weighted product* karena merupakan

penjumlahan terbobot untuk mencari penjumlahan terbobot dari *rating* kinerja pada setiap alternatif pada semua *atribut*[7].

Pada penelitian ini, penulis bermaksud mengembangun sistem pendukung keputusan pemberian beasiswa IOM berbasis *web* dengan studi kasus Politeknik Negeri Cilacap dengan menggunakan metode *Weighted Product* mampu membantu dalam pemberian beasiswa dengan memberikan nilai yang tingkat ketepatannya baik dan dengan perhitungan yang sederhana.sehingga penulis menggunakan metode *weighted product* dimana untuk menentukan calon mahasiswa yang akan menerima beasiswa berdasarkan kriteria yang ada. Dan Perbedaan penelitian ini dengan penelitian terdahulu adalah objek yang digunakan.

2.2. Landasan Teori

2.2.1. Sistem Pendukung Keputusan

Sistem Pendukung Keputusan (SPK) merupakan suatu pendekatan atau metodologi untuk mendukung keputusan[8]. SPK menggunakan CBIS (*Computer Based Information System*) yang fleksibel, interaktif dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi untuk masalah manajemen spesifik yang tidak terstruktur[8]. SPK menggunakan *data*, memberikan antarmuka pengguna yang mudah dan dapat menggabungkan pemikiran pengambil keputusan[8].

Sebagai tambahan, SPK biasanya menggunakan berbagai *model* dan dibangun oleh suatu proses interaktif dan *iterative*[8]. Ia mendukung semua fase pengambilan keputusan dan dapat memasukkan suatu komponen pengetahuan[8].

SPK dapat digunakan oleh pengguna tunggal pada satu PC atau biasa menjadi berbasis Web untuk digunakan oleh banyak orang pada beberapa lokasi[8].

1. *Weighted Product*

Weighted Product (WP) adalah salah satu metode yang digunakan untuk penyelesaian sistem pengambilan keputusan dengan mempertimbangkan kriteria dan bobot[9].

Langkah-langkah penyelesaiaan dari metode ini adalah :.

1. Menghitung nilai bobot w[10].
2. Menghitung nilai bobot s[10].
3. Menghitung nilai bobot v[10].

Dengan *formula* sebagai berikut :

Menghitung nilai bobot w :

$$W_j = \frac{W_j}{\sum Wj} \quad \text{.....(1)}$$

Persamaan (1) merupakan rumus menghitung nilai bobot W.Hasil perhitungan dari W_j apabila atributnya benefit maka dikali 1 sedangkan *cost* dikali -1[10].

Keterangan :

Wj : Bobot kriteria
W^j : w *indeks* ke-j
ΣWj : Penjumlahan bobot kriteria

Menghitung nilai bobot s :

$$S_i = \prod_{j=1}^{n} X_{ij}^{W_j} \quad \text{.....(2)}$$

Persamaan (2) merupakan rumus menghitung nilai bobot S.

Keterangan :

S^i : nilai dari setiap altenatif
X^{ij} : menyatakan nilai/skor kriteria
W^j : bobot setiap kriteria
i : menyatakan alternatif
j : menyatakan kriteria
n : menyatakan banyaknya kriteria

Menghitung nilai bobot v:

$$V_i = \frac{\prod_{j=1}^{n} X_{ij}^{W_j}}{\prod_{j=1}^{n} X_{ij} * W_j} \quad \text{.....(3)}$$

Persamaan (3) merupakan rumus menghitung nilai bobot v.

Keterangan :

V : menyatakan alternatif yang dianalogikan sebagai vektor V
X^{ij} : menyatakan nilai/skor kriteria
W^{j} : bobot setiap kriteria
i : menyatakan alternatif
j : menyatakan kriteria
n : menyatakan banyaknya kriteria.

2.2.2. Basis *Data*

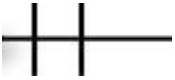
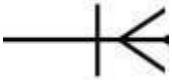
Basis *Data* adalah sekumpulan file yang berhubungan dengan *minimum redundancy* dan melayani banyak pemakai secara *optima*l[9].

1. ERD(*Entitiy Relationship Diagram*)

Pengertian ERD adalah sekumpulan cara atau peralatan untuk mendeskripsikan *data-data* atau objek-objek yang dibuat berdasarkan dan berasal dari dunia nyata yang disebut entitas (*entity*) serta hubungan (*relationship*) antar entitas-entitas tersebut dengan menggunakan beberapa notasi[11].Simbol-simbol yang digunakan dalam ERD dijelaskan pada Tabel 2.1 Simbol ERD (*Entity Relationship Diagram*) sebagai berikut :

Tabel 2.1 Simbol Erd

No	Keterangan
1. Relasi/Hubungan 	Hubungan yang terjadi antara 1 entitas atau lebih yang tidak mempunyai fisik tetapi hanya sebagai konseptual. Dan untuk mengetahui jenis hubungan yang ada antara 2 *file*[12].

2. Atribut 	Atribut ialah karakteristik dari entitas atau relasi yang menyediakan penjelasan detil tentang entitas atau relasi tersebut. Berfungsi untuk memperjelas atribut yang dimiliki oleh sebuah entitas. Atribut memiliki bentuk lingkaran lebih tepatnya elips[12].
3. Alur	Alur memiliki fungsi untuk menghubungkan atribut dengan entitas dan entitas dengan relasi. Dan berbentuk garis[12].
4. *One to One* 	*One to One*, digunakan untuk menghubungkan antar entitas dengan hubungan satu ke satu[12].
5. One to Many 	*One to Many* atau *Many to One*, digunakan untuk menghubungkan antar entitas dengan hubungan satu ke banyak atau sebaliknya[12].
6. Entitas (*Entity*) 	Entitas ialah sebuah objek berwujud nyata yang dapat dibedakan dengan objek lainnya. Objeknya dapat bersifat konkret maupun abstrak. *Data* konkret adalah sesuatu yang benar-benar ada atau dapat dirasakan oleh alat indra, sedangkan abstrak tidak berwujud[12].

2.2.3. Rekayasa Perangkat Lunak

Rekayasa Perangkat Lunak sendiri adalah aplikasi dari sebuah pendekatan kuantifiabel, disiplin dan sistematis kepada pengembangan, operasi dan pemeliharaan perangkat lunak[13].

1 Metode Pengembangan Sistem *Waterfall*

Model *Waterfall* adalah salah satu jenis *model* pengembangan aplikasi dan termasuk ke dalam *classic life cycle* (siklus hidup klasik), yang mana menekankan pada fase yang berurutan dan sistematis[3]. Untuk *model* pengembangannya, dapat dianalogikan seperti air terjun, dimana setiap tahap dikerjakan secara berurutan mulai dari atas hingga ke bawah[3].

Tahap-Tahap Metode *Waterfall*

1. *Requirement*

Tahapan metode *waterfall* yang pertama adalah *Requirement* dari *software* yang akan dikerjakan. In*form*asi dan *insight* yang diperoleh dapat berupa dari hasil wawancara, survei, studi literatur, observasi, hingga diskusi[3].

2. *Design*

Tahap yang selanjutnya adalah *Design* yaitu pembuatan desain aplikasi sebelum masuk pada proses *implementation.* Tujuan dari tahap ini, supaya mempunyai gambaran jelas mengenai tampilan dan antarmuka *software* yang kemudian akan dieksekusi oleh tim *programmer*[3].

3. *Implementation*

Tahapan metode *waterfall* yang berikutnya adalah *implementation* yaitu implementasi kode *program* dengan menggunakan berbagai *tools* dan bahasa pemrograman sesuai dengan kebutuhan tim dan perusahaan. Jadi, pada tahap implementasi ini lebih berfokus pada hal teknis, dimana hasil dari desain perangkat lunak akan diterjemahkan ke dalam bahasa pemrograman melalui tim *programmer* atau *developer*[3].

4. *Integration and Testing*

Tahap yang keempat *Integration and testing* yaitu kegiatan integrasi dan pengujian sistem. Pada tahap ini, akan dilakukan penggabungan modul yang sudah dibuat pada tahap sebelumnya. Setelah proses integrasi sistem telah selesai, berikutnya masuk pada pengujian modul[3].

5. *Operation and Maintenance*

Tahapan metode *waterfall* yang terakhir adalah *Operation and maintenance* yaitu pengoperasian dan perbaikan dari aplikasi. Setelah dilakukan pengujian sistem, maka akan masuk pada tahap produk dan pemakaian perangkat lunak oleh pengguna (*user*). Untuk proses pemeliharaan, memungkinkan pengembang untuk melakukan perbaikan terhadap kesalahan yang ditemukan pada aplikasi setelah digunakan oleh *user*[3].

2.2.4. Metode Pengujian Sistem

Metode pengujian sistem yang digunakan adalah dengan metode *black box testing*. Metode *black box testing* merupakan pengujian kualitas perangkat lunak yang berfokus pada fungsionalitas perangkat lunak[13]. Pengujian *black box* bertujuan untuk menemukan fungsi yang tidak benar, kesalahan antarmuka, kesalahan pada struktur *data*, kesalahan per*form*ansi, kesalahan inisialisasi dan terminasi. penguji (*tester*) dapat mendefinisikan kumpulan kondisi dan melakukan pengetesan pada spesifikasi fungsional *program*.

Ciri-ciri *black box testing*:

1. *Black box testing* berfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software*[14].
2. *Black box testing* bukan teknik alternatif daripada *white box testing*. Lebih dari pada itu, ia merupakan pendekatan pelengkap dalam mencakup *error* dengan kelas yang berbeda dari metode *white box testing*[14].
3. *Black box testing* melakukan pengujian tanpa pengetahuan detil struktur *internal* dari sistem atau komponen yang dites. juga disebut sebagai *behavioral testing*[14], *specification-based testing*, *input*/*output testing* atau *functional testing*[14].

Kategori *error* yang akan diketahui melalui *black box testing*:

1) Fungsi yang hilang atau tak benar/salah[14]
2) *Error* dari antar-muka/*interface*[14]

3) *Error* dari struktur *data* atau akses *eksternal database*[14]
4) Kesalahan/Error dari kinerja atau tingkah laku/*perform*[14]
5) *Error* dari *inisialisasi* dan *terminal*[14]

2.2.5. Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek (*Object Oriented Programming*/OOP) adalah suatu konsep pemrograman yang menggunakan objek sebagai suatu konsep pemrograman yang menggunakan objek sebagai *building block* dalam pengembangan aplikasinya[15].
Keuntungan menggunakan pemrograman berorientasi objek adalah sebagai berikut :

1) Pemrograman Berorientasi Objek Mudah Dirawat
   Menggunakan struktur desain berorientasi objek keterbacaan tinggi. Karena adanya warisan, bahkan jika persyaratan berubah. Jadi perawatan hanya dalam modul lokal, sehingga sangat mudah dirawat dan lebih murah[16].
2) Pemrograman Berorientasi Objek Memiliki Ekspansi Yang Mudah

Melalui warisan, kita dapat sangat mengurangi kode berlebih dan menyebarkan penggunaan kode yang ada[16].

kita bisa menggunakan modul *standar*, di sini disebut "*standar*" mengacu pada kesepakatan yang dicapai antara *programmer* satu sama lain. Pembangunan *program* bersama, Tanpa harus untuk memulai dari awal. Sehingga dapat mengurangi waktu pengembangan perangkat lunak dan meningkatkan efisiensi produksi[16].

Paket *modular* dapat mengakses definisi tingkat properti dan metode objek. Antarmuka *eksternal* diekspos melalui pengubah akses keamanan yang berbeda untuk mencegah *data* internal diubah secara tidak aman[16].

Hal ini memungkinkan *program* memiliki tingkat *modularitas* yang lebih tinggi yang nyaman untuk pemeliharaan dan modifikasi nanti. Bahasa berorientasi objek memungkinkan beberapa contoh objek ada pada saat yang sama tanpa mengganggu satu sama lain[16].

3) Kemudahan *Pemodelan*

Modifikasi kecil untuk *dimodelkan* setelah konsep objek abstrak. Meskipun bahasa berorientasi objek dengan objek kehidupan nyata bukanlah konsep yang sama[16].

Tetapi berkali-kali, sering menggunakan objek kehidupan nyata yang sangat memudahkan konstruksi Proses pencetakan. Meskipun secara langsung menggunakan objek nyata untuk *dimodelkan* terkadang bisa menjadi *kontraproduktif*[16].

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek:

1. Objek

Definisi yang *form*al dari objek adalah sebuah konsep, abstraksi atau sesuatu yang diberi batasan jelas dan dimaksudkan untuk sebuah aplikasi[17]. Objek dapat berupa benda atau suatu konsep dimana dapat disimpan *data* mengenai objek tersebut sehubungan dengan sifat dari benda yang bersangkutan.

2. Kelas

Setiap objek mempunyai identitas yang unik. identitas yang unik ini membuat kita dapat membedakan dua objek yang berbeda, walaupun kedua objek tersebut mempunyai keadaan dan nilai yang sama pada atributnya. Objek dikelompokkan menjadi kelas-kelas objek dan kelas-kelas objek terdiri dari objek-objek yang memiliki atribut dan proses yang sama[17].

3. Atribut

Atribut dari suatu kelas merepresentasikan properti-properti yang dimiliki oleh kelas tersebut[17]. Atribut mempunyai tipe yang menjelaskan tipe instansiasinya.

4 Operasi

Operasi menentukan bagaimana sebuah objek beraksi dan bereaksi terhadap permintaan dari objek lainnya[17], direpresentasikan dengan kelompok pesan yang direspon oleh objek.

5 Pesan

Pesan merupakan suatu unit komunikasi untuk menghubungkan antara satu objek dengan objek lain[17]. Suatu objek dapat meminta objek lain untuk melaksanakan suatu proses atau menyediakan *data*.

6 Enkapsulasi

Sebuah sistem yang dibangun berdasarkan metode berorientasi objek akan menghasilkan Sebuah sistem yang komponennya dibungkus (dienkapsulasi) menjadi kelompok *data* dan fungsi[17]. Dan Setiap komponen dalam sistem tersebut dapat mewarisi atribut dan sifat dari komponen lainnya, dan dapat berinteraksi satu sama lainnya.

7 Pewarisan (*inheritance*)

Merupakan mekanisme yang memungkinkan suatu objek (kelas) mewarisi sebagian atau seluruh definisi dari objek lain sebagai bagian dari dirinya[17]

2.2.6. UML

Pada pemrograman berorientasi objek, UML adalah langkah awal dalam menganalisis dan mendesain atau *memodelkan* suatu sistem, untuk sebuah *software* dengan aturan-aturan dan notasi-notasi yang ada dalam bentuk grafis yang cukup spesifik yang diperuntukan untuk di dalam *Objek Oriented Models*[18].

a) *Use Case Diagram*

Use case Diagram Merupakan *diagram* yang bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan suatu sistem tersendiri melalui sebuah cerita bagaimana sebuah sistem dipakai[13] tabel 2.2 simbol use case *diagram*.

Tabel 2.2 Simbol *Use Case Diagram*

No	Simbol	Nama dan Fungsi

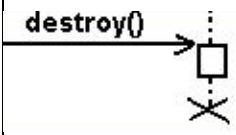
1.	 *Actor*	*Actor* befungsi untuk menjelaskan orang atau apa saja yang berinteraksi dengan sistem[19].
2.	 *Include*	*Include* berfungsi *menun*jukan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari usecase lainnya[19].
3.	*Association*	*Association* berfungsi penghubung antara *actor* dengan *use case*[19].
4.	Generalisasi	*Generalisasi* berfungsi *menun*jukan spesialisasi *actor* untuk dapat berpartisipasi dengan *use case*[19].
5.	 *System*	*System* yaitu batasan sebuah sistem[19].
6.	 *Use Case* *e*	*Use case* menjelaskan urutan kegiatan yang dilakukan *aktor* dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, *use case* hanya menjelaskan apa yang dilakukan oleh *aktor* dan sistem, bukan bagaimana *aktor* dan sistem melakukan kegiatan[19].

b) *Sequence Diagram*

Sequence diagram merupakan *diagram* yang menjelaskan alur proses dari setiap *use case* yang sudah dibuat[13]. Oleh karena itu untuk menggambar *diagram sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat *diagram sequence* juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Berikut simbol-simbol yang ada pada *diagram sequence tabel 2.3 simbol sequence diagram*.

Tabel 2.3 Simbol Sequence *Diagram*

No	Simbol	Fungsi
1.	*Actor* 	*Actor* berfungsi Orang, proses, atau sistem lain yang berinteraksi dengan sistem in*form*asi dan mendapat manfaat dari sistem[20].
2.	*Lifeline*	*Lifeline* berfungsi Menyatakan kehidupan suatu objek[20].
3.	*Object* 	*Object* berfungsi berpartisipasi secara berurutan dengan mengirimkan dan menerima pesan[20].
4.	Waktu aktif 	Waktu aktif berfungsi Menyatakan objek dalam keadaan aktif dan berinteraksi pesan[20].

5.	Pesan tipe *create* <<create>>	Pesan tipe *Create* berfungsi menyatakan Suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat[20].
6.	Pesan tipe *call* 1 : nama_method()	Pesan tipe *call* berfungsi menyatakan Suatu objek memanggil operasi atau metode yang ada pada objek lain atau dirinya sendiri[20].
7.	Pesan tipe *send* 1 : masukan	Pesan tipe *send* berfungsi menyatakan objek mengirimkan *data* atau masukan atau in*form*asi ke objek lainnya, arah panah mengarah pada objek yang dikirimi[20].
8.	Pesan tipe *return* 1 : keluaran	Pesan tipe *return* berfungsi menyatakan objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke ojek tertentu[20].
9.		Pesan tipe *destroy* berfungsi menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhir, sebaiknya jika ada create maka ada destroy[20].

2.2.7. *Flowchart*

Flowchart merupakan suatu jenis *diagram* yang mempresentasikan *algoritma* atau langkah-langkah instruksi yang berurutan dalam sistem[21].Adapun fungsinya yaitu Merancang Proyek Baru, Mengelola Alur Kerja, Me*model*kan Proses Bisnis, Mendokumentasikan Setiap Proses, Mempresentasikan Algoritma,

dan Mengaudit Proses[21].untuk melakukannya harus mengetahui simbol simblol *Flowcahart*. Simbol yang di pakai dalam *flowchart* dibagi menjadi 3 kelompok :

1) Simbol Arus
   Digunakan untuk menghubungkan simbol satu dengan yang lain (*connecting line*)[21].

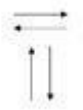
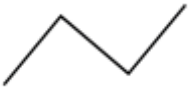
2) Simbol Proses
   Menunjukan jenis operasi pengolahan dalam suatu proses atau prosedur[21].

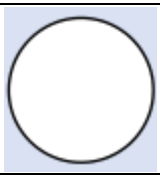
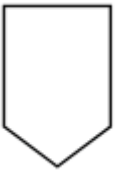
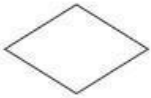
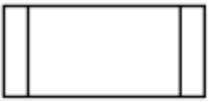
3) Simbol *Input / Ouput*
   Menunjukkan jenis peralatan yang digunakan sebagai *media input atau output*[21].

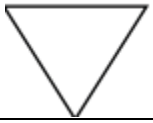
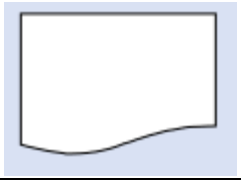
Simbol-simbol *Flowchart*

Flowchart disusun dengan simbol-simbol. Simbol ini dipakai sebagai alat bantu menggambarkan arus,proses dan *input* maupun *output* di dalam *program*[21]. Simbol-simbol yang dipakai antara lain.tabel 2.4 simbol *flowchart*

Tabel 2.4 Simbol *Flowchart*

No	]Simbol	Nama dan Fungsi
1.		*Flow Direction Symbol* berfungsi untuk menghubungkan antar simbol, menyatakan arus suatu proses[21].
2.		*Communication Link* berfungsi untuk transmisi *data* dari satu lokasi ke lokasi lain[21].

3.		*Connector* berfungsi untuk menyatakan sambungan dari proses yang satu ke proses berikutnya dihalaman yang sama[21].
4.		*Offline Connector* berfungsi untuk menyatakan sambungan dari proses yang satu ke proses berikutnya dihalaman yang berbeda[21].
5.		*Process Symbol* berfungsi untuk *menu*njukkan pengolahan yang dilakukan oleh komputer[21].
6.		*Manual Operation Symbol* berfungsi untuk *menu*njukkan pengolahan yang tidak dilakukan oleh komputer[21].
7.		*Decision Symbol* berfungsi untuk *menu*njukan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban ya atau tidak[21].
8.		*Pedefined Process* berfungsi untuk mempersiapkan penyimpanan yang sedang / akan digunakan dengan memberikan harga awal[21].
9.		*Terminal Symbol* berfungsi untuk menyatakan permulaan atau akhir suatu *program*[21].

10.		*Offline Storage* berfungsi untuk *menu*njukan bahwa *data* akan disimpan ke *media* tertentu[21].
11.		*Manual input symbol* berfungsi untuk meng*input*kan *data* secara *manual* dengan *keyboard*[21].
12.		*Input atau Output Symbol* berfungsi untuk menyatakan proses *input* atau *output* tanpa tergantung jenis peralatannya[21].
13.		*Punched Card* berfungsi untuk menyatakan masukan dan keluaran yang berasal dari *card*(kartu)[21].
14.		*Disk Strorage* berfungsi untuk menyatakan masukan dan keluaran yang berasal dari *disk*[21].
15.		*Magnetic tipe data* berfungsi untuk menyatakan masukan dan keluaran yang berasal dari pita *magnetis*[21].
16.		*Document* berfungsi untuk menyatakan masukan dan keluaran yang berasal dari dokumen[21].

17.		*Display* berfungsi untuk menyatakan keluaran melalui layar *monitor*[21].

{Halaman Sengaja Dikosongkan}