BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem Informasi

Sistem Informasi adalah sekumpulan elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan untuk mengintegrasikan data, memproses dan menyimpan serta mendistribusikan informasi. *Sutedjo* (2006:10)

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan. *Leitch* (2005:8)

Dari definisi diatas, sistem informasi dapat disimpulkan sebagai suatu pengaturan, proses dan teknologi, informasi yang berinteraksi dalam suatu informasi dimana didapat komponen yang saling berhubungan dan menghasilkan laporan-laporan yang diperlukan.

II.2. Volume Penjualan

Pada dasarnya volume penjualan cukup luas beberapa menyebutnya sebagai seni. Istilah menjual adalah ilmu dan seni mempengaruhi pribadi yang dilakukan oleh penjual untuk mengajak orang lain agar bersedia membeli barang dan jasa yang ditawarkan adalah barang yang terjual dalam bentuk uang untuk

jangka waktu tertentu dan didalamnya terdapat strategi pelayanan yang baik. (*Geraldy Tambajong ; 2013 : 1293*)

II.3. *Eksekutif*

Dari sudut pandang organisasi, eksekutif ialah orang atau kelompok orang yang memiliki kewenangan administratif atau pengawasan dalam suatu organisasi. Umumnya merupakan manajer senior yang membuat perencanaan dan kebijakan perusahaan. (*Joko Christian ; 2010 : 104*)

Eksekutif memiliki 5 fungsi utama, yaitu

1. Merencanakan (*planning*).
2. Mengorganisasikan (*organizing*).
3. Menyusun staf (*staffing*) .
4. Mengarahkan (*directing*).
5. Mengendalikan (*controlling*).

II.4. *Executive Information System*

Disebut juga sebagai *Executive Support System (ESS)* merupakan salah satu bentuk sistem informasi yang disusun dari banyak sumber data dalam bentuk summary yang dipergunakan oleh pihak manajemen senior untuk melakukan monitor *performance*, *assessment* dan pengembangan strategi bisnis. (*Joko Christian ; 2010 : 105*)

Untuk memenuhi fungsinya, EIS memiliki karakteristik sebagai berikut :

1. Dibuat secara spesifik untuk seorang eksekutif.
2. Akan digunakan langsung oleh eksekutif tanpa perantara.

3. Mampu melakukan proses ekstrak, menyaring (*filter*), menyingkat dan melacak "*critical* data".
4. Mengakses dan mengintegrasikan data internal dan eksternal.
5. Bersifat *user friendly*.

Format data yang disediakan oleh EIS harusmemenuhi kebutuhan pihak eksekutif. Berikut adalah karakteristik data yang harus didukung oleh EIS:

1. *Highly summarized* data.
2. *Drill down* dan *drill up*.
3. Integrasi data dari basis data yang berbeda - beda.
4. *Trend* jangka panjang.
5. Dapat mengakses data eksternal.
6. Informasi yang disampaikan dalam bentuk faktor penentu.

II.5. Mesin Pertanian

Penggunaan alat mesin Petanian bertujuan untuk efesiensi tenaga kerja, meningkatkan kenyamanan kerja, prestise pekerjaan, dan merubah citra usaha pertanian. Sehingga penggunaan alat pertanian akan menentukan daya saing produk, mutu, tingkat harga, jumlah produk dan kontinyuitas suplai. (*Mitra Bestari ; 2011 : 107*)

Komponen yang pelu dipersiapkan dalam produksi pertanian antara lain:

1. Traktor pengolah tanah.
2. Alat penanam.
3. Mesin penebar pupuk.
4. Mesin penyiang.

5. Mesin pemanen dan perontok.

II.6. UML (*Unified Modelling Language*)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT,*

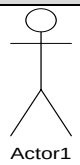
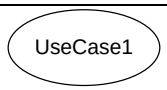
metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

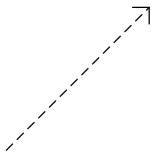
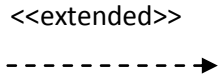
II.6.1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem,

mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. *(Yuni Sugiarti ; S.T, M.Kom ; 2013 : 41)*

Tabel II.1 Simbol-simbol Use Case Diagram

Simbol	Notasi	Keterangan
	*Actor*	Pengguna sistem atau yang berinteraksi langsung dengan sistem, bisa manusia, aplikasi, ataupun objek lain.
	*Use Case*	Digambarkan dengan lingkaran *elips* dengan nama *use case*-nya tertulis ditengah lingkaran.
	Assotiation	Digambarkan dengan sebuah garis yang berfungsi menghubungkan *actor* dengan *use case*.

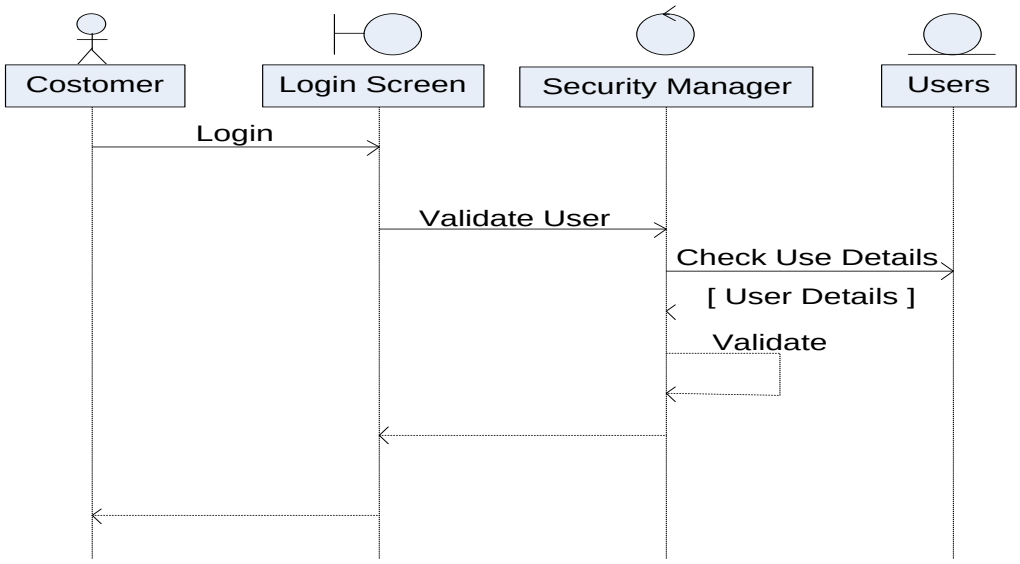
	*Defendency*	*Depenciency* merupakan relasi yang menunjukkan bahwa perubahan pada salah satu *elemen* memberi pengaruh pada *elemen* lain.
	*Extended*	*Extend*ed menunjukan bahwa suatu bagian dari *elemen* digaris tanpa panah bisa disisipkan kedalam *elemen* yang ada digaris dengan panah.

Sumber : *Yuni Sugiarti, S.T, M.Kom (2013 : 74)*

II.6.2. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.1 Contoh *Sequence* Diagram
Sumber : Yuni Sugiarti, S.T, M.Kom (2013 : 63)

II.6. 3. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

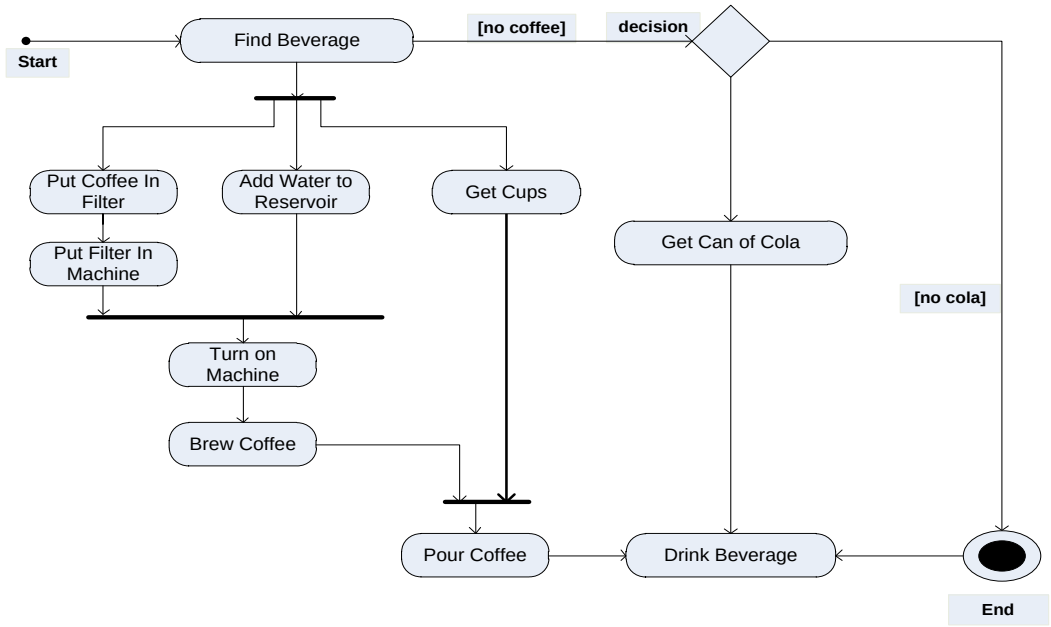
Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case*

menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar II.2 Activity Diagram
Sumber : Yuni Sugiarti, S.T, M.Kom (2013 : 76)

II.7. *Visual Studio*

Visual basic adalah bahasa pemrograman yang digunakan untuk membuat aplikasi *windows* yang berbasis grafis *(GUI = Grafic User Interface)*. Visual Basic merupakan *Event Driven Programming* (Program terkendali kejadian)

artinya program menunggu sampai adanya respon dari pemakai berupa event atau kejadian tertentu, misalnya : memilih menu utama, file, dsb. Ketika event terdeteksi, kode yang berhubungan dengan event akan dijalankan (Suryo A, 2002). Microsoft Visual Basic 6.0 adalah bahasa pemrograman yang bekerja dalam lingkup windows. Ms.Visual Basic 6.0 dapat memanfaatkan kemampuan ms.windows secara optimal.

Kemampuannya dapat dipakai untuk merancang aplikasi yang berpenampilan seperti program aplikasi lainnya berbasis ms.windows. (Agus, 2001). Struktur aplikasi visual basic terdiri dari *form*, kontrol, *properties, method, procedure, dan module. Form* adalah windows atau jendela dimana merupakan tempat untuk membuat *user interface* atau tampilan, kontrol adalah tampilan berbasis grafis yang dimasukkan pada form untuk membuat interaksi dengan pemakai *(textbox, label, scroll bar), properties* adalah nilai atau karakteristik yang dimiliki oleh sebuah obyek Visual Basic, contoh : *name, caption, size, dll*, *method* adalah serangkaian perintah yang sudah tersedia pada suatu obyek yang dapat diminta untuk mengerjakan tugas khusus, *procedure* kejadian adalah kode yang berhubungan dengan suatu obyek, kode ini akan dieksekusi ketika ada respon dari pemakai berupa event tertentu, dan *module* adalah kumpulan dari prosedur umum, deklarasi variabel dan definisi konstanta yang digunakan aplikasi. (Tominanto ; 2010 : 9).

II.8. *Database MySQL*

Sunarfrihantono (2006:56), MySQL adalah sebuah perangkat lunak sistem manajemen basis data *SQL* (bahasa Inggris: *database management system*) atau

DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti *Apache*, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL *(General Public License).* Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat *closed source* atau komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu SQL *(Structured Query Language).* SQL adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Keandalan suatu sistem *database* (DBMS) dapat diketahui dari cara kerja optimizer-nya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh

user maupun program-program aplikasinya. MySQL dapat dikatakan lebih unggul dibandingkan *database server* lainnya dalam *query* data. Hal ini terbukti untuk query yang dilakukan oleh *single user*, kecepatan *query* MySQL bisa sepuluh kali lebih cepat. Selain itu MySQL juga memiliki beberapa keistimewaan, antara lain :

1. Portability

   MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.

2. Open Source

   MySQL didistribusikan secara *open source* (gratis), dibawah lisensi GPL.

3. Multiuser

   MySQL dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

4. *Performance tuning*

   MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.

5. Column types

   MySQL memiliki tipe kolom yang sangat kompleks, seperti *signed/unsigned integer, float, double, char, text, date, timestamp,* dan lain-lain.

6. Command dan functions

   MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam *query*.

II.8.1. Tipe-Tipe Data Pada *MySQL*

Arbie (2004:26), Ada beberapa tipe data pada *database MySQL* terdiri dari beberapa tipe, diantaranya adalah :

1. Tipe Data *Numerik,* pada tipe data ini, data yang disimpan hanya data angka (*numerik*) saja. Data ini dapat disimpan dalam bentuk angka positif maupun negatif.
2. Tipe Data *String,* pada tipe data ini berisi nilai *string* (*alpha* numerik/karakter) dan numerik, nilai numerik di sini tidak untuk operasi perhitungan sebelum konversi.
3. Tipe Data dan Waktu, tipe data ini menyimpan informasi waktu, baik tanggal maupun jam. Data yang disimpan di sini *numerik,* tetapi pembacaannya terhadap data adalah *string,* maka perlu dilakukan konversi bila ingin melakukan perhitungan.
4. Tipe Data lainnya, Terdapat tiga macam tipe data lainnya, selain yang telah disebutkan di atas, yaitu *ENUM* dan *SET*. Tipe data *ENUM* merupakan tipe data yang menyimpan beberapa pilihan data yang akan disimpan, tetapi hanya satu pilihan yang boleh disimpan. Sedangkan *SET* mirip dengan *ENUM*, tetapi bisa memilih lebih dari satu pilihan.