BAB II

LANDASAN TEORI

II.1. Konsep Dasar

II.1.1. Sistem

Gordon B. Davis dalam bukunya menyatakan bahwa sistem bisa berupa abstrak atau fisik. Sistem yang abstrak adalah susunan gagasan-gagasan atau konsepsi yang teratur yang saling bergantung. Sedangkan sistem yang bersifat fisik adalah serangkaian unsur yang bekerja sama untuk mencapai suatu tujuan.

Norman L. Enger menyatakan bahwa suatu sistem dapat terdiri atas kegiatan-kegiatan yang berhubungan guna mencapai tujuan-tujuan perusahaan seperti pengendalian inventaris atau penjadwalan produksi. Sedangkan Prof. Dr. Mr. S. Prajudi Atmosudirdjo menyatakan bahwa suatu sistem terdiri atas objek-objek atau unsur-unsur atau komponen-komponen yang berkaitan dan berhubungan satu sama lainnya sedemikian rupa sehingga unsur-unsur tersebut merupakan suatu kesatuan pemrosesan atau pengolahan yang tertentu.

Secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisasi, saling berinteraksi, saling tergantung satu sama lain dan terpadu (Tata Sutabri, 2012: 3-7).

II.1.2. Karakteristik Sistem

Model umum sebuah sistem terdiri dari *input*, proses, dan *output*. Hal ini merupakan konsep sebuah sistem yang sangat sederhana mengingat sebuah sistem dapat mempunyai beberapa masukan dan keluaran sekaligus. Selain itu sebuah sistem juga memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem (Tata Sutabri, 2012: 13-14).

Adapun karakteristik yang dimaksud adalah sebagai berikut:

1. Komponen Sistem (*Components*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem.

2. Batasan Sistem (*Boundary*)

   Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem lainnya atau sistem dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkungan Luar Sistem (*Environtment*)

   Bentuk apapun yang ada diluar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut dengan lingkungan luar sistem. Lingkungan luar sistem ini dapat menguntungkan dan dapat juga merugikan sistem tersebut.

4. Penghubung Sistem (*Interface*)

   Media yang menghubungkan sistem dengan subsistem yang lain disebut dangan penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lain. Keluaran suatu subsistem akan menjadi masukan untuk subsistem yang lain dengan melewati penghubung.

5. Masukan Sistem (*Input*)

   Energi yang dimasukkan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal *(signal input).*

6. Keluaran Sistem (*Output*)

   Hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain.

7. Pengolah Sistem (*Procces*)

   Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem (*Objective*)

   Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministik. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah dirancanakan.

II.1.3. Data

Data merupakan representasi dari fakta atau gambaran mengenai suatu objek atau kejadian, ambil contoh fakta mengenai biodata mahasiswa yang

meliputi nama, alamat, jenis kelamin, agama yang dianut, dan lain-lain. Contoh lain dari fakta mengenai kejadian/transaksi dalam sebuah perusahaan dagangan adalah seperti transaksi penjualan yang meliputi waktu transaksi, pelaku transaksinya (pelanggan, kasir), barang yang ditransaksikan, serta jumlah dan harganya. Data dinyatakan dengan nilai yang berbentuk angka, deretan karakter, atau symbol (Kusrini, 2007: 3).

II.1.4. Informasi

Informasi merupakan proses lebih lanjut dari data yang sudah memiliki nilai tambah, informasi dapat dikelompokkan menjadi 3 bagian, yaitu:

1. Informasi Strategis. Informasi ini digunakan untuk mengambil keputusan jangka panjang, yang mencakup informasi eksternal, rencana perluasan perusahaan, dan sebagainya.
2. Informasi Taktis. Informasi ini dibutuhkan untuk mengambil keputusan jangka menengah, seperti informasi tren penjualan yang dapat dimanfaatkan untuk menyusun rencana penjualan.
3. Informasi Teknis. Informasi ini dibutuhkan untuk keperluan operasional sehari-hari, seperti informasi persediaan stock, retur penjualan, dan laporan kas harian.

Informasi adalah data yang telah diklasifikasikan atau diolah atau diinterpretasikan untuk digunakan dalam proses pengambilan keputusan. Sistem pengolahan informasi akan mengolah data menjadi informasi atau mengolah data dari bentuk tak berguna menjadi berguna bagi yang menerimanya. Nilai informasi

berhubungan dengan keputusan. Bila tidak ada pilihan atau keputusan maka informasi tidak diperlukan. Keputusan dapat berkisar dari keputusan berulang sederhana sampai keputusan strategis jangka panjang. Nilai informasi dilukiskan paling berarti dalam konteks pengambilan keputusan (Tata Sutabri, 2012: 21-22).

II.1.5. Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan laporan-laporan yang diperlukan oleh pihak luar tertentu.

Tipe sistem informasi adalah sebagai berikut:

1. Sistem informasi Akuntansi
2. Sistem informasi Pemasaran
3. Sistem informasi Manajemen Persediaan
4. Sistem informasi Personalia
5. Sistem informasi Distribusi
6. Sistem informasi Pembelian
7. Sistem informasi Kekayaan
8. Sistem informasi Analisis Kredit
9. Sistem informasi Penelitian dan Pengembangan
10. Sistem informasi Teknik

Semua sistem informasi tersebut dimaksudkan untuk memberikan informasi kepada semua tingkat manajemen, mulai manajemen tingkat bawah, manajemen tingkat menengah, hingga manajemen tingkat atas (Tata Sutabri, 2012: 38-41).

II.2. Sistem Pendukung Keputusan / *Decision Support System* (DSS)

Decision support system merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem itu digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi yang tidak terstruktur, dimana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat.

Decision support system biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk mengevaluasi suatu peluang. *Decision support system* yang seperti itu disebut dengan aplikasi *decision support system*. Aplikasi *decision support system* digunakan dalam pengambilan keputusan. Aplikasi *decision support system* menggunakan CBIS (*Computer Based Information Systems*) yang fleksibel, interaktif, dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi atas masalah manajemen spesifik yang tidak terstruktur.

Aplikasi *decision support system* menggunakan data, memberikan antarmuka pengguna yang mudah mudah, dan dapat menggabungkan pemikiran pengambil keputusan.

Decision support system lebih ditujukan untuk mendukung manajemen dalam melakukan pekerjaan yang bersifat analitis dalam situasi yang kurang terstruktur dan dengan kriteria yang kurang jelas.

Decision support system tidak dimaksudkan untuk mengotomatiskan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambil keputusan untuk melakukan berbagai analisis menggunakan model-model yang tersedia (Kusrini, 2007: 15-16).

II.2.1. Arsitektur Sistem Pendukung Keputusan

Aplikasi sistem pendukung keputusan bisa terdiri dari beberapa subsistem, yaitu (Kusrini, 2007: 25-26):

1. Subsistem manajemen data

   Subsistem manajemen data memasukkan satu database yang berisi data yang relavan untuk suatu situasi dan dikelola oleh perangkat lunak yang disebut sistem manajemen database (DBMS/ *Data Base Management System*). Subsistem manajemen data bisa diinterkoneksikan dengan data *werehouse* perusahaan, suatu repositori untuk data perusahaan yang relavan dengan pengambilan keputusan.

2. Subsistem manajemen model

   Merupakan paket perangkat lunak yang memasukkan model keuangan, statistik, ilmu manajemen, atau model kuantitatif lain yang memberikan kapabilitas analitik dan manajemen perangkat lunak yang tepat. Bahasa-bahasa pemodelan untuk membangun model-model kustom juga dimasukkan.

Perangkat lunak itu sering disebut sistem manajemen basis model (MBMS). Komponen tersebut bisa dikoneksikan ke penyimpanan korporat atau eksternal yang ada pada model.

3. Subsistem antarmuka pengguna

   Pengguna berkomunikasi dengan dan memerintahkan sistem pendukung keputusan melalui subsistem tersebut. Pengguna adalah bagian yang dipertimbangkan dari sistem. Para peneliti menegaskan bahwa beberapa kontribusi unik dari sistem pendukung keputusan berasal dari interaksi yang intensif antara komputer dan pembuat keputusan.

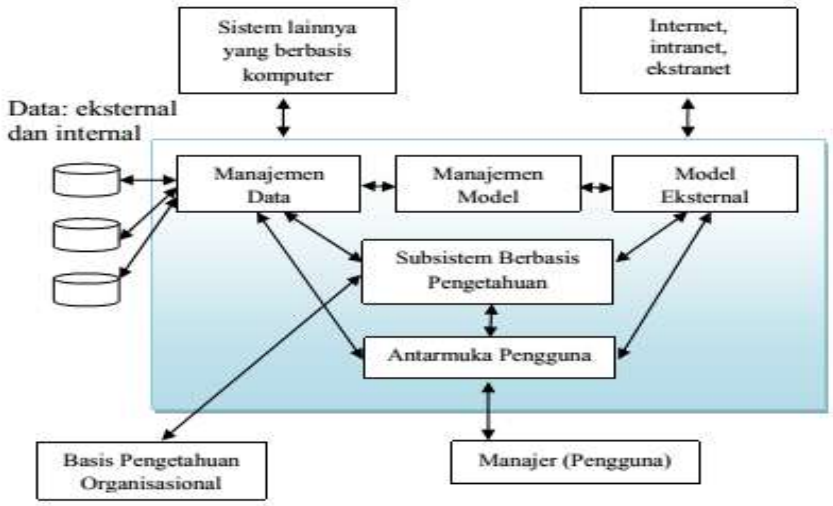
4. Subsistem manajemen berbasis-pengatahuan

   Subsistem tersebut mendukung semua subsistem lain atau bertindak langsung sebagai suatu komponen independen dan bersifat opsional.

   Selain memberikan intelegensi untuk memperbesar pengetahuan si pengambil keputusan, subsistem tersebut bisa diinterkoneksikan dengan repositori pengetahuan perusahaan (bagian dari sistem manajemen pengetahuan), yang kadang-kadang disebut basis pengetahuan organisasional.

Berdasarkan defenisi, sistem pendukung keputusan harus mencakup tiga komponen utama dari DBMS, MBMS, dan antarmuka pengguna. Subsistem manajemen berbasis pengetahuan adalah opsional, tetapi bisa memberikan banyak manfaat karena memberikan intelegensi bagi ketiga komponen utama tersebut. Seperti pada semua sistem informasi manajemen, pengguna bisa dianggap sebagai komponen sistem pendukung keputusan. Komponen-komponen tersebut membentuk sistem aplikasi sistem pendukung keputusan yang bisa dikoneksikan

ke intranet perusahaan, ekstranet, atau internet. Arsitektur dari sistem pendukung keputusan ditunjukkan dalam Gambar II.1.



Gambar II.1. Arsitektur DSS
(Sumber : Kusrini, 2007 : 26)

II.2.2. Tujuan *Decision Support System* (DSS)

Tujuan dari DSS adalah (Kusrini, 2007: 16-17):

1. Membantu manajer dalam pengambilan keputusan atas masalah semiterstruktur.
2. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.
3. Meningkatkan efektivitas keputusan yang diambil manajer lebih daripada perbaikan efisiensinya.
4. Kecepatan komputasi. Komputer memungkinkan para pengambil keputusan untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.

5. Peningkatan produktivitas. Produktivitas juga bisa ditingkatkan menggunakan peralatan optimalisasi yang menentukan cara terbaik untuk menjalankan sebuah bisnis.
6. Dukungan kualitas. Komputer bisa meningkatkan kualitas keputusan yang dibuat. Sebagai contoh, semakin banyak data yang diakses, makin banyak juga alternatif yang bisa dievaluasi.
7. Berdaya saing. Teknologi pengambilan keputusan bisa menciptakan pemberdayaan yang signifikan dengan cara memperbolehkan seseorang untuk membuat keputusan yang baik secara cepat, bahkan jika mereka memiliki pengetahuan yang kurang.
8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan.

II.3. Metode *Simple Multi-Attribute Rating Technique* (SMART)

SMART merupakan metode dalam pengambilan keputusan multi kriteria yang dikembangkan oleh Edward pada tahun 1977. Teknik pengambilan keputusan multi kriteria ini didasarkan pada teori bahwa setiap alternatif terdiri dari sejumlah kriteria yang memiliki nilai-nilai dan setiap kriteria memiliki bobot yang menggambarkan seberapa penting ia dibandingkan dengan kriteria lain. Pembobotan ini digunakan untuk menilai setiap alternatif agar diperoleh alternatif terbaik (Jadno, 2013).

SMART mengunakan *linier adaptif model* untuk meramal nilai setiap alternatif. SMART lebih banyak digunakan karena kesederhanaannya dalam merespon kebutuhan pembuat keputusan dan caranya menganalisa respon.

Analisis yang terbaik adalah transparan sehingga metode ini memberikan pemahaman masalah yang tinggi dan dapat diterima oleh pembuat keputusan. Pembobotan pada SMART mengunakan skala 0 sampai 1, sehingga mempermudah perhitungan dan perbandingan nilai pada masing-masing alternatif (Jurnal Ilmiah Mikrotek, Vol. 1, No. 1, 2013).

Model yang digunakan dalam SMART yaitu:

$$Maximize \sum_{j}^{k} = 1\, w_j . u_{ij}, \forall i = 1, \dots, n \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \text{(II. 1)}$$

Di mana :

- w_j adalah nilai pembobotan kriteria ke-*j* dari *k* kriteria.
- u_{ij} adalah nilai *utility* alternatif *i* pada kriteria *j*.
- Pemilihan keputusan adalah mengidentifikasi mana dari *n* alternatif yang mempunyai nilai fungsi terbesar.
- Nilai fungsi ini juga dapat digunakan untuk meranking *n* alternatif.

II.3.1. Teknik SMART

Adapun teknik atau langkah-langkah dalam proses SMART, yaitu (Kustiyahningsih, dkk, 2012):

1. Langkah 1 : menentukan jumlah kriteria.
2. Langkah 2 : sistem secara *default* memberikan skala 0-100 berdasarkan prioritas yang telah diinputkan kemudian dilakukan normalisasi.

   Normalisasi $= \frac{w_j}{\sum w_j}$ ... ... (II. 2)

Keterangan :

- w_j : bobot suatu kriteria
- $\sum w_j$: total bobot semua kriteria

3. Langkah 3 : memberikan nilai kriteria untuk setiap alternatif.
4. Langkah 4 : hitung nilai *utility* untuk setiap kriteria masing-masing.

$$u_i(a_i) = 100\ \frac{(\ C_{out\ i} - C_{min})}{(C_{max} - C_{min})}\% \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (\text{II}.3)$$

Keterangan :

- $u_i(a_i)$: nilai *utility* kriteria ke-1 untuk kriteria ke-i
- C_{max} : nilai kriteria maksimal
- C_{min} : nilai kriteria minimal
- $C_{out\ i}$: nilai kriteria ke-i

5. Langkah 5 : hitung nilai akhir masing-masing.

$$u\ (a_i) = \sum_{J=1}^{m} w_j u_i(a_i)\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (\text{II}.4)$$

II.3.2. Proses Pemodelan SMART

Edwards mendefinisikan ada sepuluh langkah dalam penyelesaian metode SMART yaitu (Jadno, 2013):

1. Mengidentifikasi masalah keputusan pendefenisian. Masalah harus dilakukan untuk mencari akar masalah dan batasan-batasan yang ada. Keputusan seperti apa yang akan diambil harus didefenisikan terlebih dahulu, sehingga proses pengambilan keputusan dapat terarah dan tidak menyimpang dari tujuan yang akan dicapai. Pendefenisian pembuat keputusan (*decision maker*) dilakukan

agar pemberian nilai terhadap kriteria dapat sesuai dengan kepentingan kriteria tersebut terhadap alternatif.

2. Mengidentifikasi kriteria-kriteria yang digunakan dalam membuat keputusan.
3. Mengidentifikasi alternatif-alternatif yang akan di evaluasi. Pada tahap ini akan dilakukan proses pengumpulan data.
4. Mengidentifikasi batasan kriteria yang relevan untuk penilaian alternatif. Perlu untuk membatasi nilai. Ini dapat dicapai dengan menghilangkan tujuan yang kurang penting. Edwards berpendapat bahwa tidak perlu memiliki daftar lengkap suatu tujuan. Lima belas dianggap terlalu banyak dan delapan dianggap cukup besar.
5. Melakukan peringkat terhadap kedudukan kepentingan kriteria. Dalam hal ini dinilai cukup mudah dibandingkan dengan pengembangan bobot. Hal ini perlu dilakukan untuk dapat memberikan bobot pada setiap kriteria. Karena bobot yang diberikan pada kriteria akan bergantung pada perangkingan kriteria.
6. Memberi bobot pada setiap kriteria. Pemberian bobot diberikan dengan nilai yang dapat ditentukan oleh *user* sendiri. Dalam hal ini akan dilakukan dua kali pembobotan yaitu berdasarkan kriteria yang dianggap paling penting dan berdasarkan kriteria yang dianggap paling tidak penting. Kriteria yang dianggap paling penting diberikan nilai 100. Kriteria yang penting berikutnya diberikan sebuah nilai yang menggambarkan perbandingan kepentingan relatif ke dimensi paling tidak penting. Proses ini akan diteruskan sampai pemberian bobot ke kriteria yang dianggap paling tidak penting diperoleh.

Langkah yang sama juga akan dilakukan dengan membandingkan kriteria yang paling tidak penting yang diberikan nilai 10. Kriteria yang paling penting berikutnya diberikan sebuah nilai yang menggambarkan perbandingan kepentingan relatif ke dimensi paling penting. Proses ini akan diteruskan sampai pemberian bobot ke kriteria yang dianggap paling penting diperoleh.

7. Menghitung normalisasi bobot kriteria. Bobot yang diperoleh akan dinormalkan dimana bobot setiap kriteria yang diperoleh akan dibagikan dengan hasil jumlah setiap bobot kriteria. Normalisasi juga akan dilakukan berdasarkan kriteria yang paling penting dan kriteria yang paling tidak penting. Nilai dari dua normalisasi yang diperoleh akan dicari nilai rata-ratanya.
8. Mengembangkan *single-attribute utilities* yang mencerminkan seberapa baik setiap alternatif dilihat dari setiap kriteria. Tahap ini adalah memberikan suatu nilai pada semua kriteria untuk setiap alternatif . Dalam bidang ini seorang ahli memperkirakan nilai alternatif dalam skala 0–100. Dimana 0 sebagai nilai minimum dan 100 sebagai nilai maksimum.
9. Menghitung utilitas terhadap setiap alternatif. Perhitungan dilakukan menggunakan fungsi yang telah ada yaitu $Maximize \sum_{j}^{k} = 1 \, w_j . u_{ij}$. Di mana w_j adalah nilai pembobotan kriteria ke-j dari k kriteria dan u_{ij} adalah nilai *utility* alternatif i pada kriteria j. Nilai w_j diperoleh dari langkah 6 dan nilai u_{ij} diperoleh dari langkah 8.

10. Memutuskan nilai utilitas dari setiap alternatif akan diperoleh dari langkah 9. Jika suatu alternatif tunggal yang akan dipilih, maka pilih alternatif dengan nilai utilitas terbesar.

II.3.3. Kelebihan Metode SMART

SMART memiliki beberapa kelebihan dibandingkan dengan metode pengambilan keputusan lainnya yaitu (Jadno, 2013):

1. Mungkin melakukan penambahan / pengurangan alternatif

   Pada metode SMART penambahan atau pengurangan alternatif tidak akan mempengaruhi perhitungan pembobotan karena setiap penilaian alternatif tidak saling bergantung.

2. Sederhana

   Perhitungan pada metode SMART sangat sederhana sehingga tidak memerlukan perhitungan matematis yang rumit yang memerlukan pemahaman matematika yang kuat. Penggunaan metode yang kompleks akan membuat user sulit memahami bagaimana metode bekerja.

3. Transparan

   Proses menganalisa alternatif dan kriteria dalam SMART dapat dilihat oleh *user*, sehingga *user* dapat memahami bagaimana alternatif itu dipilih. Menurut situs infoharvest.com *Answers to Frequently Asked Questions about decision analysis*, Alasan-alasan bagaimana alternatif itu dipilih dapat dilihat dari prosedur-prosedur yang dilakukan dalam SMART mulai dari penentuan kriteria, pembobotan, dan pemberian nilai pada setiap alternatif.

4. Multikriteria

   Metode SMART mendukung pengambilan keputusan dengan kriteria yang banyak. Pengambilan keputusan dengan kriteria yang banyak akan menyulitkan *user* dalam menentukan keputusan yang tepat.

5. Fleksibel pembobotan

   Pembobotan yang dipakai di dalam metode SMART ada 3 jenis, yaitu pembobotan secara langsung (*direct weighting*), pembobotan swing (*swing weighting*) dan pembobotan centroid (*centroid weighting*). Pembobotan secara langsung lebih fleksibel karena *user* dapat mengubah-ubah bobot kriteria sesuai dengan tingkat kepentingan kriteria yang diinginkan.

II.4. PHP (*Hypertext Preprocessor)*

PHP adalah bahasa *server-side scripting* yang menyatu dengan HTML untuk membuat halaman web yang dinamis. Karena PHP merupakan *server-side scripting* maka sintaks dan perintah-perintah PHP akan dieksekusi di *server* kemudian hasilnya dikirimkan ke *web browser* dalam format HTML. Dengan demikian kode program yang ditulis dalam PHP tidak akan terlihat oleh *user* sehingga keamanan halaman web lebih terjamin. PHP dirancang untuk membentuk halaman web dinamis, yaitu halaman web yang dapat membentuk suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data ke halaman web.

PHP termasuk dalam *Open Source Product,* sehingga *source code* PHP dapat diubah dan didistribusikan secara bebas. PHP juga dapat berjalan pada

berbagai *web server* seperti ISS (*Internet Information Server*), PWS (*Personal Web Server*), Apache, Xitami. PHP juga mampu lintas *platform.* Artinya PHP dapat berjalan di banyak sistem operasi yang beredar saat ini, di antaranya: Sistem Operasi Microsoft Windows (semua versi), Linux, Mac OS, Solaris. PHP dapat dibangun sebagai modul pada *web server* Apache dan sebagai *binary* yang dapat berjalan sebagai CGI (*Common Gateway Interface).* PHP dapat mengirim HTTP *header*, dapat mengatur *cookies,* mengatur *outhentication* dan *redirect users*.

Salah satu keunggulan yang dimiliki oleh PHP adalah kemampuannya untuk melakukan koneksi ke berbagai macam *software* sistem manajemen basis data/*Database Management System* (DBMS), sehingga dapat menciptakan suatu halaman web yang dinamis. PHP mempunyai koneksitas yang baik dengan beberapa DBMS antara lain Oracle, Sybase, mSql, MySQL, Microsoft SQL Server, Solid, PostgreSQL, Adabas, FilePro, Velocis, dBase, Unix dbm, dan tak terkecuali semua database ber-*interface* ODBC (M. Rudiyanto Arief, 2011: 43).

II.5. MySQL

MySQL pertama kali dirintis oleh seorang programmer *database* bernama Michael Widenius. MySQL *database server* adalah RDBMS (*Relasional Database Management System*) yang dapat menangani data yang bervolume besar. Meskipun begitu tidak menuntut *resource* yang besar (Wahana Komputer, 2010: 5).

MySQL adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan *database*

sebagai sumber dan pengolahan datanya. Kepopuleran MySQL antara lain karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses *database*-nya sehingga mudah untuk digunakan, kinerja *query* cepat, dan mencukupi untuk kebutuhan *database* perusahaan-perusahaan skala menengah-kecil. MySQL juga bersifat *open source* dan *free* pada berbagai *platform* (kecuali pada windows yang bersifat *shareware*). MySQL didistribusikan dengan lisensi *open source* GPL (*General Public License*) mulai versi 3.23, pada bulan Juni 2000.

MySQL merupakan *database* yang pertama kali didukung oleh bahasa pemrograman *script* untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan *software* pengembangan aplikasi web yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman *script* PHP (M. Rudiyanto Arief, 2011: 151).

II.5.1. Kelebihan dan Keuntungan Memakai MySQL

MySQL memiliki beberapa kelebihan dan keuntungan dibanding *database* lain, di antaranya adalah (Wahana Komputer, 2010: 7):

1. Banyak ahli berpendapat MySQL merupakan *server* tercepat.
2. MySQL merupakan sistem manajemen *database* yang *Open Source* (kode sumbernya terbuka), yaitu *software* ini bersifat *free* atau bebas digunakan oleh perseorangan atau instansi tanpa harus membeli atau membayar kepada pembuatnya.
3. MySQL mempunyai performa yang tinggi tapi simpel.

4. *Database* MySQL mengerti bahasa SQL (*Structured Query Language*).
5. MySQL dapat diakses melalui *protocol* ODBC (*Open Database Connectivity*) buatan Microsoft. Ini menyebabkan MySQL dapat diakses oleh banyak *software*.
6. Semua klien dapat mengakses *server* dalam satu waktu, tanpa harus menunggu yang lain untuk mengakses *database*.
7. *Database* MySQL dapat diakses dari semua tempat di internet dengan hak akses tertentu.
8. MySQL merupakan *database* yang mampu menyimpan data berkapasitas besar, sampai berukuran *Gigabyte*.
9. MySQL dapat berjalan di berbagai *operating system* seperti Linux, Windows, Solaris, dan lain-lain.

II.6. *Database*

Database adalah sebuah struktur yang umumnya terbagi dalam 2 hal, yaitu sebuah *database flat* dan *database* relasional. *Database* relasional lebih mudah dipahami daripada *database flat* karena *database* relasional mempunyai bentuk yang sederhana serta mudah dilakukan operasi data. MySQL sendiri adalah sebuah *database* relasional. *Database* yang memiliki struktur relasional terdapat tabel-tabel untuk menyimpan data. Pada setiap tabel terdiri dari kolom dan baris serta sebuah kolom untuk mendefinisikan jenis informasi apa yang harus disimpan (Wahana Komputer, 2010: 2).

Database adalah kumpulan data yang saling terkait yang diorganisasi untuk memenuhi kebutuhan dan struktur sebuah organisasi serta bisa digunakan oleh lebih dari satu orang dan lebih dari satu aplikasi (Kusrini, 2007: 33).

II.7. Normalisasi

Normalisasi adalah suatu teknik dengan pendekatan *bottom-up* yang digunakan untuk membantu mengidentifikasikan hubungan. Dimulai dari menguji hubungan, yaitu *functional dependencies* antara atribut. Pengertian lainnya adalah suatu teknik yang menghasilkan sekumpulan hubungan dengan sifat-sifat yang diinginkan dan memenuhi kebutuhan pada perusahaan (Indrajani, 2015: 7).

Proses normalisasi merupakan proses pengelompokan elemen data menjadi tabel-tabel yang menunjukkan entitas dan relasinya. Proses ini selalu diuji pada beberapa kondisi. Apakah ada kesulitan pada saat menambah *(insert),* menghapus *(delete),* mengubah *(update),* atau membaca *(retrieve)* pada suatu database. Bila ada kesulitan pada pengujian tersebut maka relasi dapat dipecah dalam beberapa tabel lagi (Tata Sutabri, 2012: 138).

Adapun bentuk-bentuk normalisasi menurut Indrajani (2015: 9) adalah sebagai berikut:

1. *Unnormalized Form* (UNF)

   Merupakan suatu tabel yang berisikan satu atau lebih grup yang berulang. Membuat tabel yang *unnormalized*, yaitu dengan memindahkan data dari sumber informasi.

Contoh: nota penjualan yang disimpan ke dalam format tabel dengan baris dan kolom.

2. *First Normal Form* (1NF)

   Merupakan sebuah relasi di mana setiap baris dan kolom berisikan satu dan hanya satu nilai.

   Proses UNF ke 1NF:

   a. Tentukan satu atau sekumpulan atribut sebagai kunci untuk tabel *unnormalized*.
   b. Identifikasikan grup yang berulang dalam tabel *unnormalized* yang berulang untuk kunci atribut.

3. *Second Normal Form* (2NF)

   Berdasarkan pada konsep *full functional dependency*, yaitu A dan B merupakan atribut sebuah relasi. B dikatakan *fully dependent* terhadap A jika B *functional dependent* pada A, tetapi tidak pada propersubset dari A. 2NF merupakan sebuah relasi dalam 1NF dan setiap atribut *non-primary-key* bersifat *fully functionally dependent* pada *primary key*.

   1NF ke 2NF:

   a. Identifikasikan *primary key* untuk relasi 1NF.
   b. Identifikasikan *functional dependencies* dalam relasi.
   c. Jika terdapat *partial dependencies* terhadap *primary key*, maka hapus dengan menempatkan dalam relasi yang baru bersama salinan determinannya.

4. *Third Normal Form* (3NF)

   Berdasarkan pada konsep *transitive dependency*, yaitu suatu kondisi di mana A, B, dan C merupakan atribut sebuah relasi, maka A – B dan B – C, maka *transitively dependent* pada A melalui B. 3NF adalah sebuah relasi dalam 1NF dan 2NF, di mana tidak terdapat atribut *non primary key* yang bersifat *transitively dependent* pada *primary key*.

   2NF ke 3NF:

   a. Identifikasi *primary key* dalam relasi 2NF.
   b. Identifikasi *functional dependencies* dalam relasi.
   c. Jika terdapat *transitive dependencies* terhadap *primary key* hapus dengan menempatkannya dalam relasi yang baru bersama dengan salinan determinannya.

5. *Boyce-code Normal Form* (BCNF)

   Berdasarkan pada *functional dependencies* yang dimasukkan dalam hitungan seluruh *candidate key* dalam suatu relasi. Bagaimanapun BCNF juga memiliki batasan-batasan tambahan disamakan dengan defenisi umum dari 3NF. Suatu relasi dikatakan BCNF, jika dan hanya jika setiap determinan merupakan *candidate key*. Perbedaan antara 3NF dan BCNF yaitu untuk *functional dependent* A – B, 3NF memungkinkan *dependency* ini dalam suatu relasi jika adalah atribut *primary key* dan A bukan merupakan *candidate key*. Sedangkan BCNF menetapkan dengan jelas bahwa untuk *dependency* ini agar ditetapkan dalam relasi A, maka A harus merupakan *candidate key*. Setiap relasi dalam BCNF juga merupakan 3NF, tetapi relasi dalam 3NF belum tentu

termasuk ke dalam BCNF. Dalam BCNF kesalahan jarang sekali terjadi, Kesalahan dapat terjadi pada relasi yang:

a. Terdiri atas 2 atau lebih *composite candidate key*.
b. *Candidate key overlap,* sedikitnya satu atribut.

II.8. *Unified Modeling Language* (UML)

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual. Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek (Jurnal Informatika Mulawarman, Vol 6, No. 1, 2011).

Dengan menggunakan UML, tim pengembang *software* akan mempunyai banyak keuntungan, seperti memudahkan komunikasi dengan sesama anggota tim tentang *software* apa yang akan dibuat, memudahkan integrasi ke dalam area pengerjaan *software* karena bahasa ini berbasiskan *meta-models* dimana *metamodels* bisa mendefinisikan proses-proses untuk mengkonstruksikan konsep-konsep yang ada (ULTIMA InfoSys, Vol. IV, No. 1, 2013).

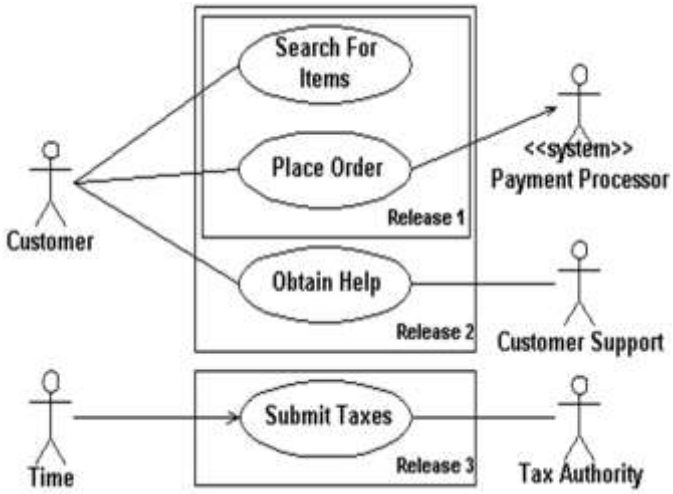
II.8.1. *Use Case* Diagram

Merupakan suatu diagram yang berisi *use case, actor,* serta *relationship* diantaranya. *Use case* diagram merupakan titik awal yang baik dalam memahami dan menganalisis kebutuhan sistem pada saat perancangan. Notasi yang

digunakan dalam *Use Case* adalah persegi panjang yang merupakan *system boundary*, oval yang merupakan suatu proses, dan gambar orang yang berinteraksi dalam proses tersebut (Indrajani, 2015: 45).

Use Case memiliki dua istilah, yaitu:

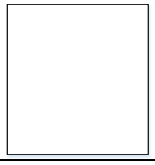
1. *System use case*; interaksi dengan sistem.
2. *Business use case*; interaksi bisnis dengan konsumen atau kejadian nyata.

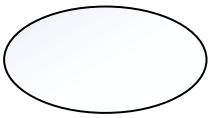


Gambar II.2. Notasi *Use Case* Diagram
(Sumber : Jurnal Informatika Mulawarman, Vol. 6, No. 1, 2011)

Adapun komponen-komponen yang digunakan dalam membuat *use case* diagram dapat dilihat pada tabel II.1.

Tabel II.1. Komponen *Use Case* Diagram

Gambar	Keterangan
	*System Boundary* menggambarkan batasan antara sistem dengan *actor.*

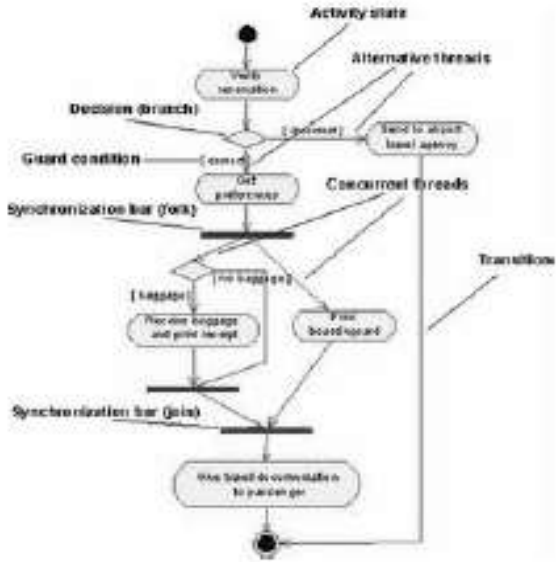
	Simbol ini menggambarkan interaksi antara *actor* dengan *software* aplikasi tersebut.
	*Actor* menggambarkan pengguna dari sistem, dapat berupa manusia atau sistem terotomatisasi lain yang berinteraksi dengan sistem lain untuk berbagi, mengirim, dan menerima informasi.
	Menggambarkan hubungan antar *actor* dan *use case.*

(Sumber : Indrajani, 2015 : 46)

II.8.2. *Activity* Diagram

Digunakan untuk menganalisis *behavior* dengan *use case* yang lebih kompleks dan menunjukkan interaksi-interaksi di antara mereka satu sama lain. *Activity* diagram sebenarnya memiliki kesamaan dengan *statechart* diagram dalam hal menggambarkan aliran data pada model bisnis, tetapi *activity* diagram biasanya digunakan untuk menggambarkan aktivitas bisnis yang lebih kompleks, di mana digambarkan hubungan antar satu *use case* dengan *use case* lainnya (Indrajani, 2015: 46).

Activity diagram menggambarkan aktifitas-aktifitas, objek, *state*, *transisi state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas (Jurnal Informatika Mulawarman, Vol. 6, No. 1, 2011).



Gambar II.3. Notasi *Activity* Diagram
(Sumber : Jurnal Informatika Mulawarman, Vol. 6, No. 1, 2011)

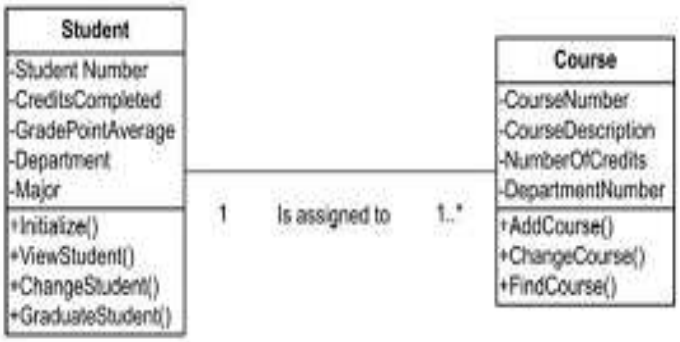
II.8.3. *Class* Diagram

Diagram ini digunakan untuk menggambarkan perbedaan yang mendasar antara *class*-*class*, hubungan antar-*class*, di mana sub-sistem *class* tersebut. Pada *class* diagram terdapat nama *class, attributes, operations,* serta *association* (hubungan antar-*class*) (Indrajani, 2015: 49).

Berbagai simbol yang hadir didalam *class* diagram antara lain adalah (ULTIMA InfoSys, Vol. IV, No. 1, 2013) :

1. *Class*, yang berfungsi untuk merepresentasikan tipe dari data yang dimilikinya. *Class* diagram dapat ditampilkan dengan menunjukkan atribut dan operasi yang dimilikinya atau hanya menunjukkan nama *class*-nya saja. Dapat juga kita tuliskan nama *class* dengan atributnya saja atau nama *class* dengan operasinya.

2. *Attribute*, merupakan data yang terdapat didalam *class* dan *instance*-nya dengan operator.
3. *Operation*, berfungsi untuk merepresentasikan fungsi-fungsi yang ditampilkan oleh *class* dan *instance*-nya dengan operator.
4. *Association*, digunakan untuk menunjukkan bagaimana dua *class* berhubungan satu sama lainnya. *Association* ditunjukkan dengan sebuah garis yang terletak diantara dua *class*. Didalam setiap *association* terdapat *multiplicity*, yaitu simbol yang mengindikasikan berapa banyak *instance* dari *class* pada ujung *association* yang satu dengan *instance class* di ujung *association* lainnya.
5. *Generalizations*, berfungsi untuk mengelompokkan *class* ke dalam hirarki *inheritance*.
6. *Aggregation*, merupakan bentuk khusus dari *association* yang merepresentasikan hubungan "*part-whole*". Bagian "*whole*" dari hubungan ini sering disebut dengan *assembly* atau *aggregate*. *Class* yang satu dapat dikatakan merupakan bagian dari *class* yang lain yang ikut membentuk *class* tersebut.
7. *Composition*, merupakan jenis *aggregation* yang lebih kuat diantara dua *class* yang memiliki *association* dimana jika *whole* ditiadakan, maka *part*-nya juga ikut ditiadakan. Berbeda dengan *aggregation*, *part* akan tetap bisa berdiri sendiri meskipun bagian *whole*-nya ditiadakan.
8. Penggunaan operator (+) dalam *class* diagram diartikan dengan *public*, operator (-) diartikan *private*, dan operator (#) diartikan *protected*.

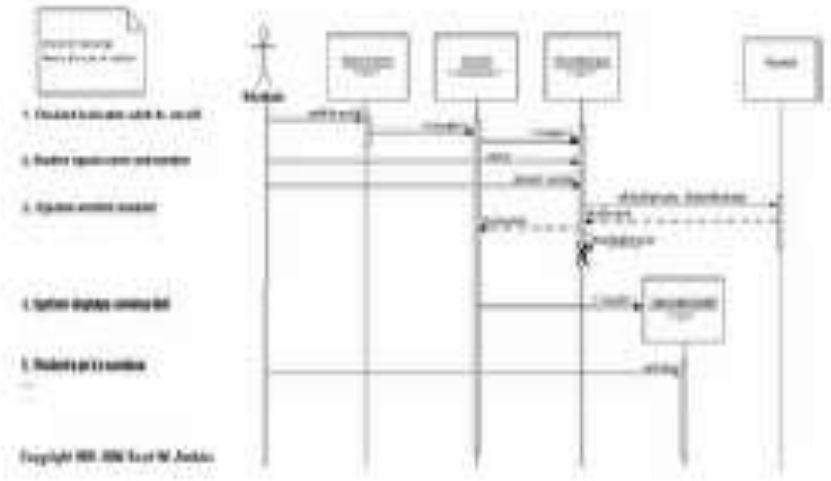


Gambar II.4. Contoh *Class* Diagram
(Sumber : ULTIMA InfoSys, Vol. IV, No. 1, 2013)

II.8.4. *Sequence* Diagram

Merupakan suatu diagram interaksi yang menggambarkan bagaimana objek-objek berpartisipasi dalam bagian interaksi (*particular interaction*) dan pesan yang ditukar dalam urutan waktu. *Sequence* diagram dapat digambarkan dalam beberapa level secara detail dan untuk tujuan yang berbeda pada beberapa langkah yang dikembangkan secara *lifecycle* (Indrajani, 2015: 50).

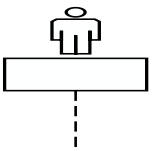
Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram* (Jurnal Informatika Mulawarman, Vol. 6, No. 1, 2011).

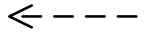


Gambar II.5. Notasi *Sequence* Diagram
(Sumber : Jurnal Informatika Mulawarman, Vol 6, No. 1, 2011)

Adapun komponen-komponen yang digunakan dalam membuat *sequence* diagram dapat dilihat pada tabel II.2.

Tabel II.2. Komponen *Sequence* Diagram

Gambar	Keterangan
	*Object Lifeline*: menggambarkan *object* apa saja yang terlibat.
	*Actor*: menggambarkan *actor* yang terlibat.
	*Activation*: menggambarkan hubungan antara *object* dengan *message*.
	Message (call): menggambarkan alur *message* yang merupakan kejadian dari objek pengirim *lifeline* ke objek penerima *lifeline*.

	*Message (Return)*: menggambarkan alur pengembalian *message* ke objek pemanggil dan tanda bahwa objek penerima telah menyelesaikan prosesnya.

(Sumber : Indrajani, 2015 : 51)