BAB II

TINJAUAN PUSTAKA

II.1 Pengertian Karyawan atau Sumber Daya Manusia

Sumber daya manusia (SDM) di dalam sebuah perusahaan memiliki peran penting dalam menentukan kemajuan suatu perusahaan. Dalam hal ini yang menjadi faktor penentu bukanlah kuantitas sumber daya manusia tersebut, melainkan kualitasnya sebagai individu. Karena sumber daya manusia yang berkualitas akan menunjang kinerja perusaahan dan menjadi penentu apakah sebuah perusahaan tersebut akan mengalami kemajuan atau hanya menjadi perusahaan biasa yang pada akhirnya tidak dapat bersaing di pasar dan kemudian mengalami kebangkrutan.

Karyawan atau SDM yang bekerja dalam perusahaan umumnya diterima melalui proses seleksi terlebih dulu. Dalam proses ini, data-data karyawan dikumpulkan sebagai catatan internal. Nama karyawan, tempat dan tanggal lahir, alamat rumah, status pernikahan, pengalaman kerja, dan latar belakang pendidikan merupakan contoh sebagian data yang biasa disimpan bagian SDM. Data-data ini menggambarkan profil karyawan dan biasanya diberikan sendiri oleh karyawan.Selanjutnya, data-data seleksi, seperti tingkat kecerdasan (IQ), tingkat emosional (EQ), aspek psikologis, dan kesehatan.

Tuntutan terhadap kebutuhan SDM dipengaruhi oleh keadaan dan pengaruh teknologi. Teknologi yang berubah dan masih terus akan berubah dalam

arti akan ditemukan berbagai alat yang dapat menggantikan tenaga kerja manusia yang lebih baik, efektif, dan efisien. Hasil ini dapat dilihat dari sejarah peradaban manusia, pada saat ditemukan mesin uap hingga komputer dan robot yang dikendalikan oleh komputer, dan kemudian teknologi komunikasi yang sangat canggih. Pada masa yang akan datang tentu akan ditemukan berbagai teknologi baru yang lebih efisien dan efektif untuk melakukan pekerjaan.

Temuan-temuan di atas akan sangat mempengaruhi dunia kerja, dalam hal semakin mengecilnya kebutuhan perusahaan terhadap tenaga kerja manusia karena telah digantikan oleh mesin-mesin canggih. Jenis keahlian dan keterampilan SDM yang dibutuhkan juga akan berubah ke arah penguasaan teknologi, dalam hal bagaimana mengoperasikan dan memelihara berbagai teknologi canggih itu untuk proses produksi. Dengan keadaan ini tentu saja program-program dan kegiatan SDM harus diarahkan untuk meningkatkan keterampilan yang sesuai dengan tuntutan teknologi (Yeni Febriana Saragih, 2011: 06 - 07).

II.2 *Decision Support Systems.*

Sistem pendukung pengambilan keputusan kelompok (DSS) adalah sistem berbasis komputer yang interaktif, yang membantu pengambil keputusan dalam menggunakan data dan model untuk menyelesaikan masalah yang tidak terstruktur. Sistem pendukung ini membantu pengambilan keputusan manajemen dengan menggabungkan data, model-model dan alat-alat analisis yang komplek, serta perangkat lunak yang akrab dengan tampilan pengguna ke dalam satu sistem yang memiliki kekuatan besar (*powerful*) yang dapat mendukung pengambilan

keputusan yang semi atau tidak terstruktur. DSS menyajikan kepada pengguna satu perangkat alat yang fleksibel dan memiliki kemampuan tinggi untuk analisis data penting. Dengan kata lain, DSS menggabungkan sumber daya intelektual seorang individu dengan kemampuan komputer dalam rangka meningkatkan kualitas pengambilan keputusan. DSS diartikan sebagai tambahan bagi para pengambil keputusan, untuk memperluas kapabilitas, namun tidak untuk menggantikan pertimbangan manajemen dalam pengambilan keputusannya.

Konsep DSS dimulai pada akhir tahun 1960-an dengan *timesharing* komputer. Untuk pertama kalinya seseorang dapat berinteraksi langsung dengan komputer tanpa harus melalui spesialis informasi. Baru pada tahun 1971, istilah DSS diciptakan oleh G. Anthony Gorry dan Michael S. Scott Morton, keduanya professor MIT. Mereka merasa perlunya suatu kerangka kerja untuk mengarahkan aplikasi komputer kepada pengambilan keputusan manajemen dan mengembangkan apa yang telah dikenal sebagai *Garry & Scott Morton Grid.*

Matrik *(Grid)* ini didasarkan pada konsep Simon mengenai keputusan terprogram dan tak terprogram serta tingkat-tingkat manajemen Robert N. Anthony. Gory dan Scott Morton menggambarkan jenis-jenis keputusan menurut struktur masalah, dan terstruktur hingga tidak terstruktur. Anthony menggunakan nama *perencanaan strategis, pengendalian manajemen, dan pengendalian operasional* untuk menjelaskan tingkat manajemen puncak, menengah dan bawah.

Tahap-tahap pengambilan keputusan Simon digunakan untuk menentukan struktur masalah. Masalah terstruktur merupakan suatu masalah yang memiliki struktur pada tiga tahap pertama Simon, yaitu intelijen, rancangan, dan pilihan.

Jadi, dapat dibuat algoritma, atau alternatif diidentifikasi dan dievaluasi, serta suatu solusi dipilih. Masalah tak terstruktur**,** sebaliknya, merupakan suatu masalah yang sama sekali tidak memiliki struktur pada tiga tahap Simon di atas. Masalah semi terstruktur merupakan masalah yang memiliki struktur hanya pada satu atau dua tahap Simon Gory dan Scoot Morton memisahkan masalah yang telah, pada saat itu, berhasil dipecahkan dengan komputer dari masalah yang belum terkena pengolahan komputer. Area yang berhasil dipecahkan dengan komputer dinamakan *sistem keputusan terstruktur (structure decision system-SDS)*, dan area yang belum terkena pengolahan komputer dinamakan *sistem pendukung keputusan (decision support system-DSS).*

Arsitektur sistem pendukung keputusan.

Terdiri dari subsistem sebagai berikut:

1. Subsistem Manajemen Data:

   Dikelola oleh perangkat lunak sistem manajemen database (DBMS / Data Base Management System)

2. Subsistem Manajemen Model:

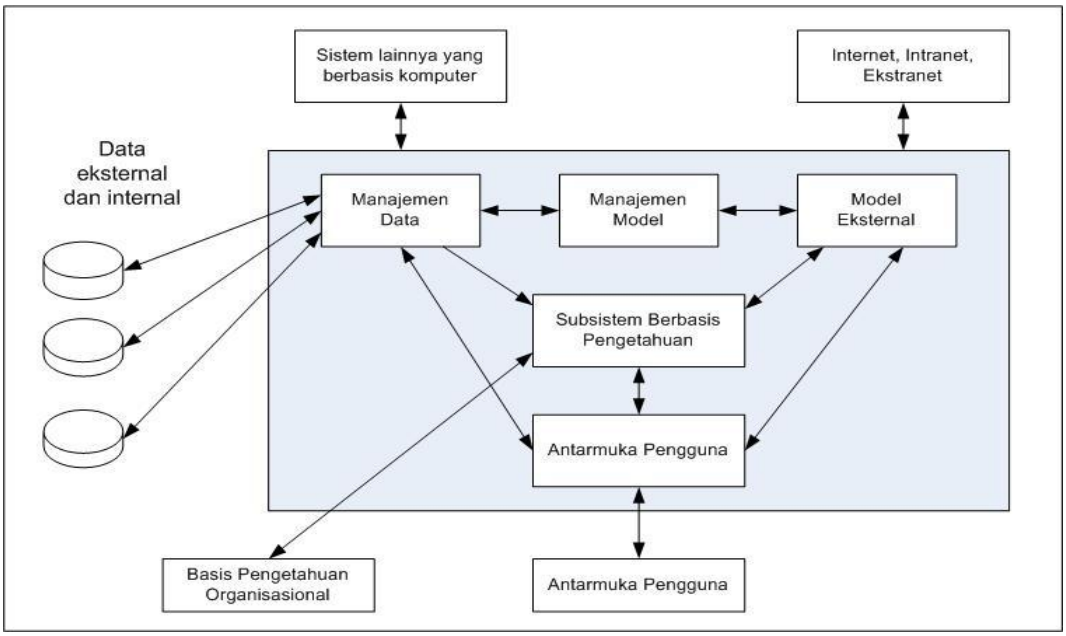
   Paket perangkat lunak (disebut sistem manajemen basis model / MBMS) yang memasukkan model keuangan, statistik, ilmu manajemen atau model kuantitatif lain yang memberikan kapabilitas analitik dan menajemen perangkat lunak yang tepat.

3. Subsistem Antarmuka Pengguna:

   Pengguna berkomunikasi dengan dan memerintahkan sistem pendukung keputusan melalui subsitem ini.

4. Subsistem Manajemen Berbasis-Pengetahuan:

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



Gambar II.1: Arsitektus DSS

a. Tujuan DSS

Perintis DSS yang lain Peter G. W. Keen, bekerjasama dengan Scott Morton mendefinisikan tiga tujuan yang harus dicapai DSS. Tujuan-tujuan ini berhubungan dengan tiga prinsip dasar dari konsep DSS – struktur masalah, dukungan keputusan, dan efektivitas keputusan. Mereka percaya bahwa DSS harus:

1. Membantu manajer membuat keputusan untuk memecahkan masalah semi – terstruktur.
2. Mendukung penilaian manajer bukan mencoba menggantikannya.

3. Meningkatkan efektifitas pengambilan keputusan manajer daripada efisiensinya (Saliman, 2011: 05 - 08).

II.3 Data *Mining*

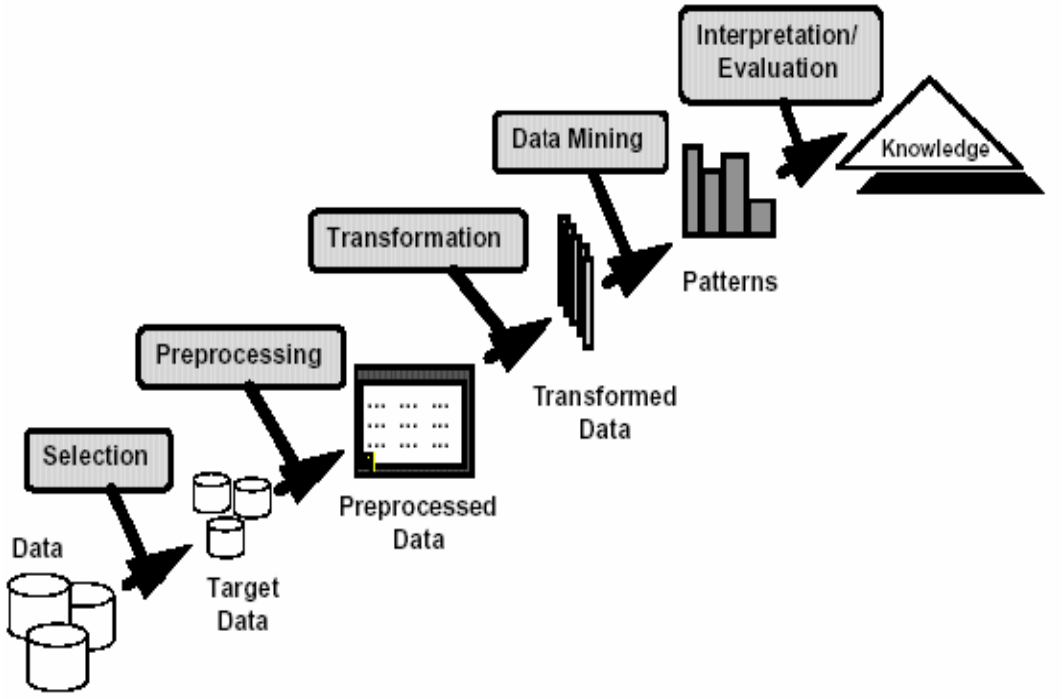
Data Mining merupakan teknologi baru yang sangat berguna untuk membantu perusahaan-perusahaan menemukan informasi yang sangat penting dari *gudang data* mereka. Kakas *data mining* meramalkan tren dan sifat-sifat perilaku bisnis yang sangat berguna untuk mendukung pengambilan keputusan penting. Analisis yang diotomatisasi yang dilakukan oleh data mining melebihi yang dilakukan oleh sistem pendukung keputusan tradisional yang sudah banyak digunakan. *Data Mining* dapat menjawab pertanyaan-pertanyaan bisnis yang dengan cara tradisional memerlukan banyak waktu untuk menjawabnya. *Data Mining* mengeksplorasi basis data untuk menemukan pola-pola yang tersembunyi, mencari informasi pemrediksi yang mungkin saja terlupakan oleh para pelaku bisnis karena terletak di luar ekspektasi mereka.

Data mining didefinisikan sebagai satu set teknik yang digunakan secara otomatis untuk mengeksplorasi secara menyeluruh dan membawa ke permukaan relasi-relasi yang kompleks pada set data yang sangat besar. Set data yang dimaksud di sini adalah set data yang berbentuk tabulasi, seperti yang banyak diimplementasikan dalam teknologi manajemen basis data relasional. Akan tetapi, teknik-teknik *data mining* dapat juga diaplikasikan pada representasi data yang lain, seperti domain data *spatial*, berbasis *text*, dan multimedia (citra). *Data mining* dapat juga didefinisikan sebagai "pemodelan dan penemuan pola-pola yang tersembunyi dengan memanfaatkan data dalam volume yang besar. *Data*

mining menggunakan pendekatan *discovery-based* dimana pencocokan pola (*pattern-matching*) dan algoritma-algoritma yang lain digunakan untuk menentukan relasi-relasi kunci di dalam data yang diekplorasi.

Tugas utama *Data Mining* bahwa pada kebanyakan aplikasinya, gol utama dari *data mining* adalah untuk membuat prediksi dan deskripsi. Prediksi menggunakan beberapa *variabel* atau *field-field* basis data untuk memprediksi nilai-nilai variabel masa mendatang yang diperlukan, yang belum diketahui saat ini. Deskripsi berfokus pada penemuan pola-pola tersembunyi dari data yang ditelaah (Moertini veronika S, 2002: 47 - 50).

Proses pada data mining ini secara lebih detail terdiri dari lima tahap seperti terdapat pada gambar II.1 berikut:



Gambar II.2 : Tahapan-tahapan dalam *Data Mining*

(Sumber: Bertalya, 2009: 4)

Tahap-tahapnya dimulai dari pemrosesan *raw data* (data mentah) sampai pada penyaringan hingga ditemukannya *knowledge*, dijabarkan sebagai berikut:

1. *Selection*, yaitu proses memilih dan memisahkan data berdasarkan beberapa kriteria, misalnya berdasarkan kota tempat tinggal konsumen.
2. *Preprocessing*, yaitu mempersiapkan data, dengan cara membersihkan data, informasi atau *field* yang tidak dibutuhkan, yang jika dibiarkan hanya akan memperlambat proses *query*, misalnya nama pelanggan jika kita sudah mengetahui kode pelanggannya. Selain itu juga, ditahap ini dilakukan penyeragaman format terhadap data yang tidak konsisten, misalnya pada suatu *field* dari suatu tabel, data jenis kelamin diinputkan dengan "L" atau "M", sedangkan pada tabel yang lain, data tersebut diinputkan sebagai "P" atau "W".
3. *Transformation*, pada tahap ini dilakukan transformasi terhadap data dengan menambahkan data tertentu sehingga membuat data menjadi lebih muda untuk digunakan dan dinavigasikan.
4. *Data mining*, tahap ini dipusatkan untuk mendapatkan pola dari data *(extraction of data).*
5. *Interpretation and evaluation*, pola yang telah diidentifikasi oleh sistem kemudian di terjemahkan/diintepretasikan menjadi bentuk *knowledge* yang lebih mudah dimengerti oleh *user* untuk membantu pengambilan keputusan, misalnya menunjukan *item* yang saling berasosiasi melalui grafik atau bentuk lain yang lebih mudah dimengerti (Zainul fanani, 2010: 26 - 27).

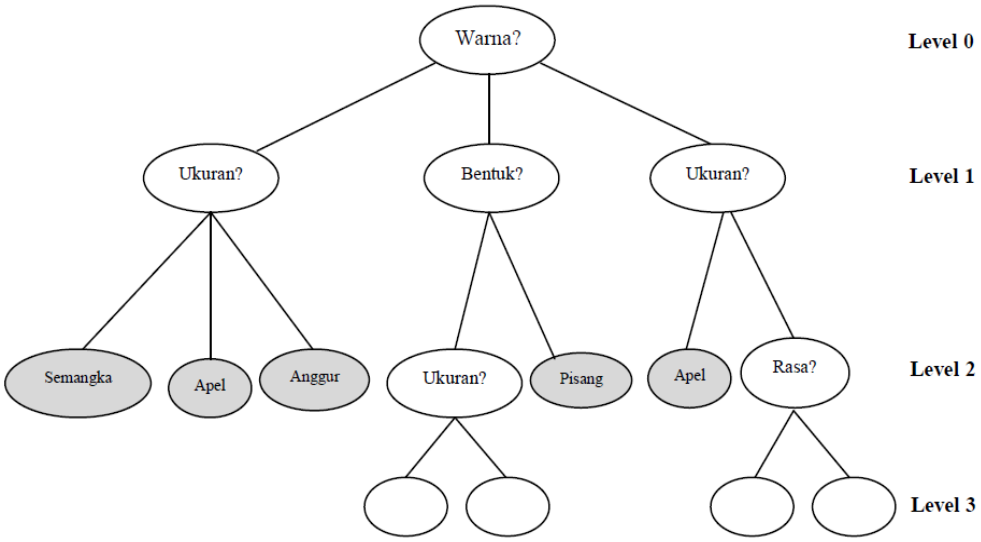
II.3.1 Pengertian *Decision Tree*

Decision tree (pohon keputusan) adalah suatu cara untuk mewakili serangkaian aturan yang mengarah ke kelas atau nilai. (Two Crows Corporation, 2005). *Decision tree* diartikan pula sebagai sebuah alat pendukung keputusan yang menggunakan pohon untuk menggambarkan hierarki dan memodelkan persoalan. *Decision tree* biasanya digunakan dalam penelitian operasional, khususnya dalam analisis keputusan, untuk membantu mengidentifikasi strategi yang paling mungkin untuk mencapai tujuan.

Decision tree sering pula digunakan sebagai alat *deskriptif* untuk menghitung probabilitas kondisional. Setiap *Decision Tree* terdiri dari simpul *(nodes*), cabang (*branches*) dan daun *(leaves*). Simpul yang paling atas disebut simpul akar (*root node*). Setiap pohon selalu dimulai dari simpul akar dan menjulur ke bawah dengan memecah data pada setiap level ke simpul baru. Simpul akar berisi seluruh kumpulan data dan simpul anak berturut-turut mengandung bagian dari kumpulan data tersebut. Seluruh simpul dihubungkan oleh cabang dan setiap simpul yang terletak di ujung cabang disebut simpul akhir (terminal *nodes*) atau daun (Michael Negnevitsky, 2005 : 352)

Sering kali untuk mengklasifikasikan obyek, diperlukan urutan beberapa pertanyaan sebelum bisa menentukan kelompoknya. Jawaban pertanyaan pertama akan mempengaruhi pertanyaan berikutnya dan seterusnya. Dalam decision tree, pertanyaan pertama akan ditanyakan pada simpul akar di *level 0*. Jawaban dari pertanyaan ini dikemukakan dalam cabang-cabang. Jawaban dalam cabang akan disusul dengan pertanyaan kedua lewat simpul yang berikutnya pada *level 1*.

Dalam setiap *level* ditanyakan nilai atribut melalui sebuah simpul. Jawaban dari pertanyaan itu dikemukakan melalui cabang-cabang. Langkah ini akan berakhir di suatu simpul jika di situ sudah jelas kelas atau jenis obyek yang dicari. Apabila dalam satu tingkat suatu obyek sudah diketahui termasuk dalam jenis buah apa, maka berhenti di *level* tersebut. Jika tidak, disusul dengan pertanyaan di *level* berikutnya hinggga jelas ciri-cirinya dan bisa menentukan jenis buahnya. Terlihat seperti gambar II.2 berikut:



Gambar II.3: *Decision tree*

(sumber: Mukhlis Fuadi, 2010: 31)

Decision tree berusaha untuk menciptakan satu set simpul daun semurni mungkin, yaitu di mana masing-masing *record* dalam simpul daun tertentu memiliki klasifikasi yang sama. Dengan cara ini, *decision tree* dapat memberikan tugas klasifikasi dengan ukuran tertinggi yang terbaik. (Larose, 2005 : 109).

Decision tree sesuai digunakan untuk kasus-kasus dimana outputnya bernilai diskrit. Walaupun banyak variasi model *decision tree* dengan tingkat

kemampuan dan syarat yang berbeda, pada umumnya beberapa ciri kasus berikut cocok untuk diterapkan *decision tree:* (Budi Santosa, 2007 : 88)

1. Data/*example* dinyatakan dengan pasangan atribut dan nilainya. Misalnya atribut satu sampel adalah temperatur dan nilainya adalah dingin. Biasanya untuk satu sampel nilai dari satu atribut tidak terlalu banyak jenisnya. Dalam contoh atribut warna ada beberapa nilai yang mungkin yaitu hijau, kuning, merah. Sedang dalam atribut temperatur nilainya bisa dingin, sedang atau panas. Tetapi untuk beberapa kasus bisa saja nilai temperatur berupa nilai numerik.
2. Label/*output* data biasanya bernilai diskrit. *Output* ini bisa berniai ya atau tidak, sakit atau tidak sakit, diterima atau ditolak. Dalam beberapa kasus mungkin saja outputnya tidak hanya dua kelas. Tetapi penerapan *decision tree* lebih banyak untuk kasus *binary.*
3. Data mempunyai *missing value*. Misalkan untuk beberapa sampel, nilai dari suatu atributnya tidak diketahui. Dalam keadaan seperti ini *decision tree* masih mampu memberi solusi yang baik.

Persyaratan tertentu harus dipenuhi sebelum *algoritma decision tree* akan diterapkan pada suatu permasalahan: (Larose, 2005 : 109)

1. *Algoritma decision tree* merepresentasikan *supervised learning,* oleh karena itu memerlukan pra - klasifikasi *variabel* target. *Set* data pelatihan dengan algoritma nilai-nilai *variabel* target harus diberikan.
2. *Set* data pelatihan ini harus kaya dan beragam, penyediaan algoritma dengan persilangan dari berbagai jenis data *record* yang telah diklasifikasikan mungkin

diperlukan di masa depan. Pohon keputusan belajar dengan contoh, dan jika contoh secara sistematis tidak memiliki *subset* data *record* yang jelas, klasifikasi dan prediksi untuk subset ini akan bermasalah atau tidak mungkin.

3. Atribut target kelas harus *diskrit*. Artinya, orang tidak dapat menerapkan keputusan pohon analisis untuk *variabel* sasaran *kontinue*. Sebaliknya, *variabel* target harus mengambil nilai-nilai yang jelas batas-batasnya baik sebagai milik kelas tertentu atau bukan.

II.3.1.1 Manfaat *Decision Tree*

Manfaat utama dari penggunaan *decision tree* adalah kemampuannya untuk mem break down proses pengambilan keputusan yang kompleks menjadi lebih simpel sehingga pengambil keputusan akan lebih menginterpretasikan solusi dari permasalahan. *Decision tree juga* berguna untuk mengeksplorasi data, menemukan hubungan tersembunyi antara sejumlah calon variabel input dengan sebuah variabel target. Pohon keputusan memadukan antara eksplorasi data dan pemodelan, sehingga sangat bagus sebagai langkah awal dalam proses pemodelan bahkan ketika dijadikan sebagai model akhir dari beberapa teknik lain. Sering terjadi tawar menawar antara keakuratan model dengan transparansi model.

Dalam beberapa aplikasi, akurasi dari sebuah klasifikasi atau prediksi adalah satu-satunya hal yang ditonjolkan, misalnya sebuah perusahaan direct mail membuat sebuah model yang akurat untuk memprediksi anggota mana yang berpotensi untuk merespon permintaan, tanpa memperhatikan bagaimana atau mengapa model tersebut bekerja. Penggunaan decision tree dalam kasus seperti ini akan sangat membantu (Fairuz abadi, 2009).

II.3.1.2 Keuntungan *Decision Tree*

Keuntungan dari metode *decision tree* adalah: (Fairuzabadi, 2009)

1. Daerah pengambilan keputusan yang sebelumnya kompleks dan sangat *global*, dapat diubah menjadi lebih simpel dan spesifik.
2. Eliminasi perhitungan-perhitungan yang tidak diperlukan, karena ketika menggunakan metode *decision tree* maka sampel diuji hanya berdasarkan kriteria atau kelas tertentu.
3. Fleksibel untuk memilih fitur dari internal *node* yang berbeda, fitur yang terpilih akan membedakan suatu kriteria dibandingkan kriteria yang lain dalam node yang sama. Kefleksibelan metode *decision tree* ini meningkatkan kualitas keputusan yang dihasilkan jika dibandingkan ketika menggunakan metode penghitungan satu tahap yang lebih konvensional.
4. Dalam analisis *multivariat,* dengan kriteria dan kelas yang jumlahnya sangat banyak, seorang penguji biasanya perlu untuk mengestimasikan baik itu distribusi dimensi tinggi ataupun parameter tertentu dari distribusi kelas tersebut. Metode *decision tree* dapat menghindari munculnya permasalahan ini dengan menggunakan kriteria yang jumlahnya lebih sedikit pada setiap *node* internal tanpa banyak mengurangi kualitas keputusan yang dihasilkan.

II.3.1.3 Kelemahan *Decision Tree*

Kelemahan dari metode *decision tree* adalah: (Fairuzabadi, 2009)

1. Terjadi *overlap* terutama ketika kelas-kelas dan kriteria yang digunakan jumlahnya sangat banyak. Hal tersebut juga dapat menyebabkan meningkatnya waktu pengambilan keputusan dan jumlah memori yang diperlukan.

2. Pengakumulasian jumlah *error* dari setiap tingkat dalam sebuah *decision tree* yang besar.
3. Kesulitan dalam mendesain *decision tree* yang optimal.
4. Hasil kualitas keputusan yang didapatkan dari metode *decision tree* sangat tergantung pada bagaimana pohon tersebut didesain.

II.4 Algoritma ID3.

ID3 adalah algoritma *decision tree learning* (algoritma pembelajaran pohon keputusan) yang paling dasar. Algoritma ini melakukan pencarian secara menyeluruh (*greedy*) pada semua kemungkinan pohon keputusan. Algoritma ID3 berusaha membangun *decision tree* (pohon keputusan) secara *top-down* (dari atas ke bawah), mulai dengan pertanyaan: "Atribut mana yang pertama sekali harus dicek dan diletakkan pada *root*?" Pertanyaan ini dijawab dengan mengevaluasi semua atribut yang ada menggunakan suatu ukuran statistic (yang banyak digunakan adalah *information gain*) untuk mengukur efektifitas suatu atribut dalam mengklasifikasikan kumpulan sampel data.

A. *Entropy*

Untuk menghitung information *gain*, terlebih dahulu kita harus memahami suatu aturan lain yang disebut *entropy.* Di dalam bidang *Information Theory*, kita sering menggunakan *entropy* sebagai suatu parameter untuk mengukur *heterogenitas* (keberagaman) dari suatu kumpulan sampel data. Jika kumpulan sampel data semakin *heterogen*, maka nilai *entropy*-nya semakin besar. Secara matematis, *entropy* dirumuskan sebagai berikut :

$$Entropy = \sum_{i}^{c} -p_i \log_2 p_i$$

di mana c adalah jumblah nilai yang ada pada atribut target (jumblah kelas klasifikasi). Sedangkan pi menyatakan jumlah sampel untuk kelas i.

B. Information Gain

Setelah mendapatkan nilai entropy untuk suatu kumpulan sampel data, maka kita dapat mengukur efektifitas suatu atribut dalam mengklasifikasikan data. Ukuran efektifitas ini disebut sebagai information gain. Secara matematis, *information gain* dari suatu atribut A, dituliskan sebagai berikut:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

di mana :

A : atribut

V : menyatakan suatu nilai yang mungkin untuk atribut A

Values(A) : himpunan nilai-nilai yang mungkin untuk atribut A

|Sv| : jumblah sampel untuk nilai v

|S| : jumlah seluruh data Entropy (Sv) : entropy untuk sampel-sampel yang memiliki nilai v.

II.5 Basis Data

Basis data adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Basis data dapat diasumsikan sebagai lemari arsip yang memiliki prinsip kerja dan tujuan yang sama. Prinsip utamanya mengatur data atau arsip, sedangkan tujuan utamanya adalah kemudahan, kecepatan dalam pengambilan kembali data atau

arsip. Model data adalah kumpulan perangkat konseptual untuk menggambarkan data, hubungan data, semantik (makna) data dan batasan data.

Dalam pembuatan sistem ini model data yang digunakan adalah *Model Entity-Relationship (Model E-R).* Komponen utama pembentukan *Model Entity-Relationship,* yaitu entitas (*entity*) dan relasi (*relation*). Kardinalitas relasi menunjukan jumlah maksimum relasi yang terjadi antara himpunan entitas yang satu dengan hubungan entitas yang lain. Kardinalitas relasi yang terjadi antara himpunan entitas (misalnya A dan B) dapat berupa :

1:1 *(One to One)*

Satu entitas pada himpunan entitas A berhubungan dengan satu entitas pada himpunan entitas B, dan sebaliknya.

1:M *(One to Many*)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya.

M:1 *(Many to One)*

Setiap entitas pada himpinan A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, tetapi tidak sebaliknya.

M:M *(Many to Many)*

Setiap entitas pada himpunan entitas A berhubungan dengan banyak entitas pada himpunan entitas B, dan sebaliknya.

Model Entity-Relationship yang berisi komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan atribut yang

mempresentasikan fakta dari ‘dunia nyata’ dapat digambatkan lebih sistematis dengan menggunakan *Diagram Entity- Relationship*.

Untuk dapat memahami prinsip-prinsip perancangan *database*, yang perlu dipahami terlebih dahulu adalah konsep dasar dan terminologinya. Hal yang sangat mendasar dan harus dipahami adalah pemodelan entitas dan *relationship*. Entitas adalah barbagai hal dalam dunia nyata yang informasinya disimpan dalam *database*. Sebagai contoh,kita dapat menyimpan data pegawai dan pekerja untuk departemen tertentu. Dalam kasus ini, pegawai merupakan suatu entitas dan departemen juga merupakan suatu entitas. *Relationship* adalah hubungan antara entitas.

Sebagai contoh, seorang pegawai berkerja untuk suatu departemen. Bekerja untuk adalah *relationship* antara entitas pegawai dan entitas departemen. *Relationship* terdiri dari 3 derajat berbeda, yaitu *one-to-one,one-to-many (manyto-one), many-to-many. One-to-one* menghubungkan secara tepat dua entitas dengan satu kunci (*key*). Misalnya, dalam satu perusahaan satu orang pegawai memiliki satu komputer kerja. *One-to-many* hubungan antara entitas dimana kunci pada satu table muncul berkali-kali dalam table lainnya. Misalnya, banyak pegawai bekerja untuk satu departemen. *Many-to-many* merupakan yang sering menyebabkan permasalahan dalam prakteknya. Dalam hubungan *many-to-many*, kunci utama*(primary key*) dari table kedua dapat muncul beberapa kali pada table pertama. Misalnya, dalam suatu perusahaan banyak pegawai bekarja untuk banyak departemen. Untuk mengatasi permasalahan tersebut, dibutuhkan table antara atau relasi (Denny Henrry Bonai, 2011: 27-30).

II.6. Perangkat Lunak Pendukung

Perangkat lunak pendukung digunakan sebagai alat untuk membantu penulis dalam proses pembuatan program sistem pendukung keputusan yang akan dirancang diantaranya yaitu :

II.6.1. PHP *(Hypertext Preprocessor)*

PHP merupakan singkatan dari *Hypertext Preprocessor* serta merupakan bahasa pemrograman skrip yang ditempatkan dalam *server* dan diproses di server. Hasilnya dikirimkan ke klien, tempat pemakai menggunakan *browser.*

PHP dirancang untuk membuat aplikasi *web* yang bersifat dinamis. Artinya, ia dapat membentuk suatu tampilan berdasarkan permintaan terkini, menampilkan isi *database* kehalaman *web* pada prinsipnya PHP mempunyai fungsi yang sama dengan skip-skip seperti ASP *(Active Server Page).* PHP bermula saat Ramus Lerdorp membuat sejumlah skrip *Perl* yang dapat mengamati siapa saja yang meliha t-lihat daftar riwayat hidupnya, yakni pada tahun 1994. Skrip-skip ini selanjutnya dikemas menjadi tool yang disebut *"Personal Home Page"* Paket inilah yang menjadi cikal-bakal PHP. Pada tahun 1995, Rasmus Lerdorp menciptakan PHP/FI Versi 2.

Pada versi inilah pemprograman dapat menempelkan kode terstruktur didalam tag HTML, kode PHP bisa berkomunikasi dengan *database* dan melakukan perhitungan-perhitungan yang kompleks. Pada awalnya, PHP dirancang untuk diintegrasikan dengan *web server Apache*. Namun, belakangan PHP juga dapat berkerja dengan *web server* seperti PWS *(Personal Web Server),* ISS *(Internet Information Server),* dan *Xitami.*

Skrip PHP berkedudukan sebagai tag dalam bahasa HTML, sebagaimana diketahui, HTML *(Hypertext Markup Language)* adalah bahasa standart untuk membuat halaman-halaman *web*. (Sumber : Abdul Kadir, 2008: 2 - 3)

PHP adalah program aplikasi yang bersifat *Server Slide*, artinya hanya dapat berjalan pada sisi server saja dan tidak dapat berfungsi tampa adanya sebuah server didalamnya dan merupakan sebuah bahasa pemprograman scripting berlisensi *Open Source*. Script ini dapat bercampur dengan *Script Tag HTML* sehingga karena kemampuannya tersebut, ia disebut sebagai bahasa yang *embeded* pada Tag HTML. (Sumber : Bunafit Nugroho, 2005: 369 - 370)

II.6.2 MySQL *(My Structure Query Language)*

MySQL adalah salah satu jenis *database server software* yang sangat terkenal, kepopulerannya disebabkan *MySQL* menggunakan SQL sebagai bahasa dasar untuk mengakses *database*nya dan bersifat *Open Source* (Anda tidak perlu membayar untuk menggunakannya) perangkat lunak sendiri bisa di-downloand dari http://www.mysql.com. *MySQL* termasuk jenis RDBMS *(Relational Database Management System)*, itu sebabnya, istilah seperti table, baris dan kolom digunakan pada *MySQL*. Pada *MySQL* sebutan *database* mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau beberapa kolom. MySQL menyediakan program mysql yang berfungsi untuk mengakses *database server MySQL* dari sisi klien. (Sumber: Abdul Kadir, 2008: 348)

II.6.3. *SQL Query*

Penggunaan *Query* pada pengolahan database merupakan tindakan yang biasa dilakukan sebagai rutinitas.*Query* itu sendiri merupakan perintah untuk mengakses *database.* Jenis *Query* yang umum dipakai adalah DDL dan MDL (Denny Henrry Bonai, 2011: 33 - 35).

Pada kedua jenis query ini terdapat 4 unsur utama, yaitu Menciptakan, Mengubah, Menghapus dan Menampilkan

DDL merupakan perintah SQL yang digunakan untuk mendefinisikan atau mendeklarasikan objek *database,*menciptakan objek *database* atau menghapus objek *database.* DDL juga dapat digunakan untuk membuat koneksi antar tabel *database* beserta batasannya dengan menentukan indeks sebagai kuncinya. DDL yang dipakai adalah:

1. *CREATE* : Digunakan untuk menciptakan objek *database* yang baru atau menciptakan database itu sendiri.
2. *DROP* : Digunakan untuk menghapus objek *database.*
3. *ALTER* : Digunakan untuk mengubah atribut atau entitas dari objek suatu *database*

DML merupakan *query* yang digunakan untuk memanipulasi data, seperti untuk menampilkan data, mengubah data, menghapus data atau mengisi data. DML yang dipakai adalah:

1. *SELECT* : Merupakan *Query* yang digunakan untuk mengambil data atau menampilkan data.
2. *INSERT* : Digunakan untuk memasukan data kedalam tabel.

3. *DELETE* : Digunakan untuk menghapus data.

4. *UPDATE* : Digunakan untuk melakukan perubahan pada data.

II.6.4 *Apache*

Web server Apache berbasiskan *open source* dan mulai popular di *internet* semenjak tahun 1996 karena *open source*, Apache bebas didistribusikan oleh siapa saja dan ke siapa saja.

Software ini dapat diunduh pada situs http://www.apache.org dan tersedia untuk berbagai *platform (Windows, Linux, dan UNIX).* Supaya dokumen-dokumen *web*, baik berupa HTML ataupun PHP bisa diakses oleh *browser* maka dokumen-dokumen tersebut perlu diletakkan dalam direktori khusus yang diatur oleh *Apache.* (Sumber : Abdul Kadir, 2008: 360)

II.6.4 *Dreamweaver CS3*

Dreamweaver adalah sebuah HTML editor profesional untuk mendesain secara visual dan mengelola situs *web* maupun halaman *web*. Pada *Dreamweaver* CS3, terdapat beberapa kemampuan bukan hanya sebagai *softwere* untuk desain *web* saja tetapi untuk menyunting kode-kode serta pembuaatan aplikasi web dengan menggunakan berbagai bahasa pemrograman *web* seperti: JPS, PHP, ASP, dan *ColdFusion. Dreamweaver* merupakan *softwere* utama yang digunakan oleh web desainer maupun *web programmer* dalam mengembangkan suatu situs *web.* Hal ini disebabkan ruang kerja, fasilitas dan kemampuan *Dreamweaver* yang mampu meningkatkan produktivitas dan efektivitas desain maupun suatu situs *web*.

Dreamweaver CS3 memiliki kemampuan *toolbar,* dimana *Dreamweaver* CS3 dapat digunakan untuk memodifikasi tampilan toolbar atau menambahkan fungsi baru. Selain *user interface* baru, *Dreamweaver* CS3 memiliki kemampuan untuk menyunting kode dengan lebih baik, *Dreamweaver* CS3 juga dapat melakukan *print code* pada jendela *code view,* selain itu juga memiliki fasilitas *code hints* yang membantu dalam urusan tag-tag serta tag *inspector* yang sangat berguna dalam menangani tag-tag HTML. Ruang kerja *Dreamweaver* CS3 memiliki komponen-komponen yang memberikan fasilitas dan ruang untuk menuangkan kreasi saat berkerja. Komponen-komponen yang tersedia oleh ruang kerja.

Window Dreamweaver CS3 ini dibagi menjadi 7 bagian, yaitu :

Insert Bar,

Document Toolbar,

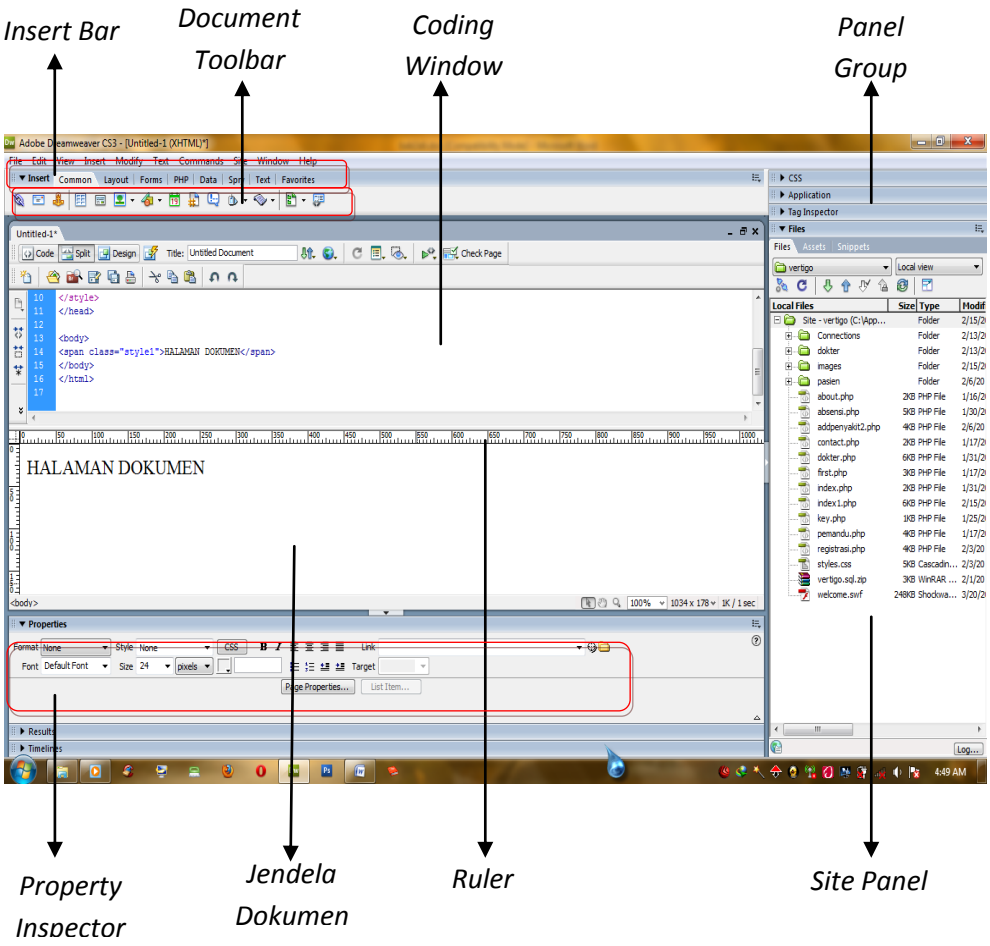
Document Window,

Panel Groups,

Tag Selector,

Property Inspector dan

Site Panel. Terlihat seperti gambar II.3 berikut;



Gambar II.4 Tampilan ruang kerja tipe *coder*

(Sumber : Madcoms, 2008, Hal: 3-4)

Keterangan Gambar :

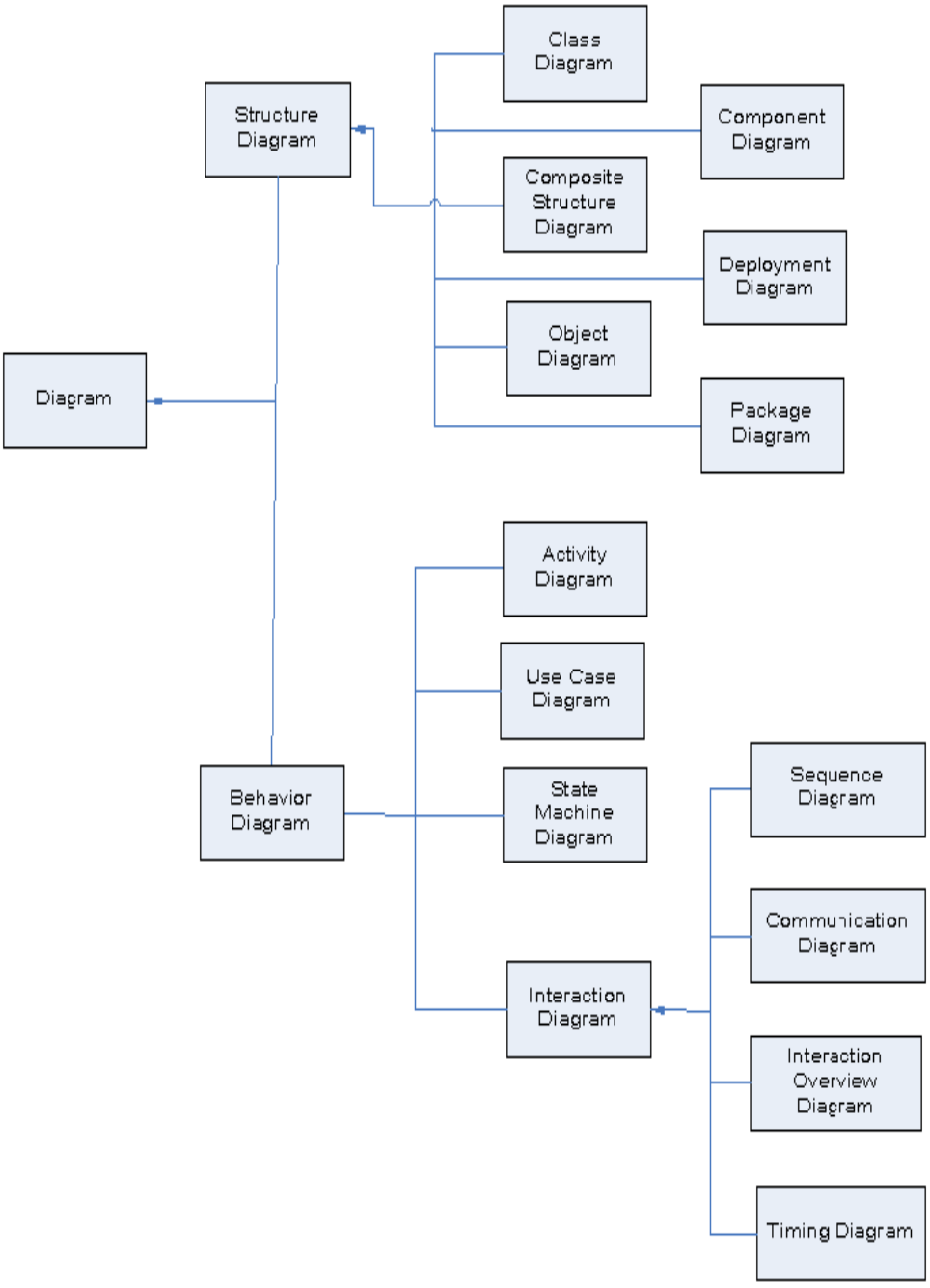
1. *Insert bar*, memuat tombol-tombol yang berfungsi untuk memasukkan/ menyisipkan berbagai jenis obyek seperti gambar, tabel dan layer ke dalam suatu dokumen. Setiap obyek yang dimasukkan dengan meng-klik tombol

insert pada *insert bar* ini adalah seperti halnya memasukkan potongan tag HTML ke dalam halaman yang sedang dibuat.

2. *Document window*, berfungsi untuk menampilkan dokumen di mana kita sekarang bekerja.
3. *Document toolbar*, berisi tombol dan menu *pop-up* yang menyediakan tampilan yang berbeda-beda dari *Document Window.*
4. *Panel groups,* adalah kumpulan panel yang saling berkaitan satu sama lain, yang dikelompokkan di bawah satu judul.
5. *Tag selector*, berfungsi untuk menampilkan *hierarki tag* di sekitar pilihan yang aktif pada *Design View*.
6. *Property inspector*, berfungsi untuk melihat dan mengubah berbagai *property* obyek atau teks.
7. *Files panel*, berfungsi pengaturan file-file atau folder-folder yang membentuk situs web/ direktori kerja.

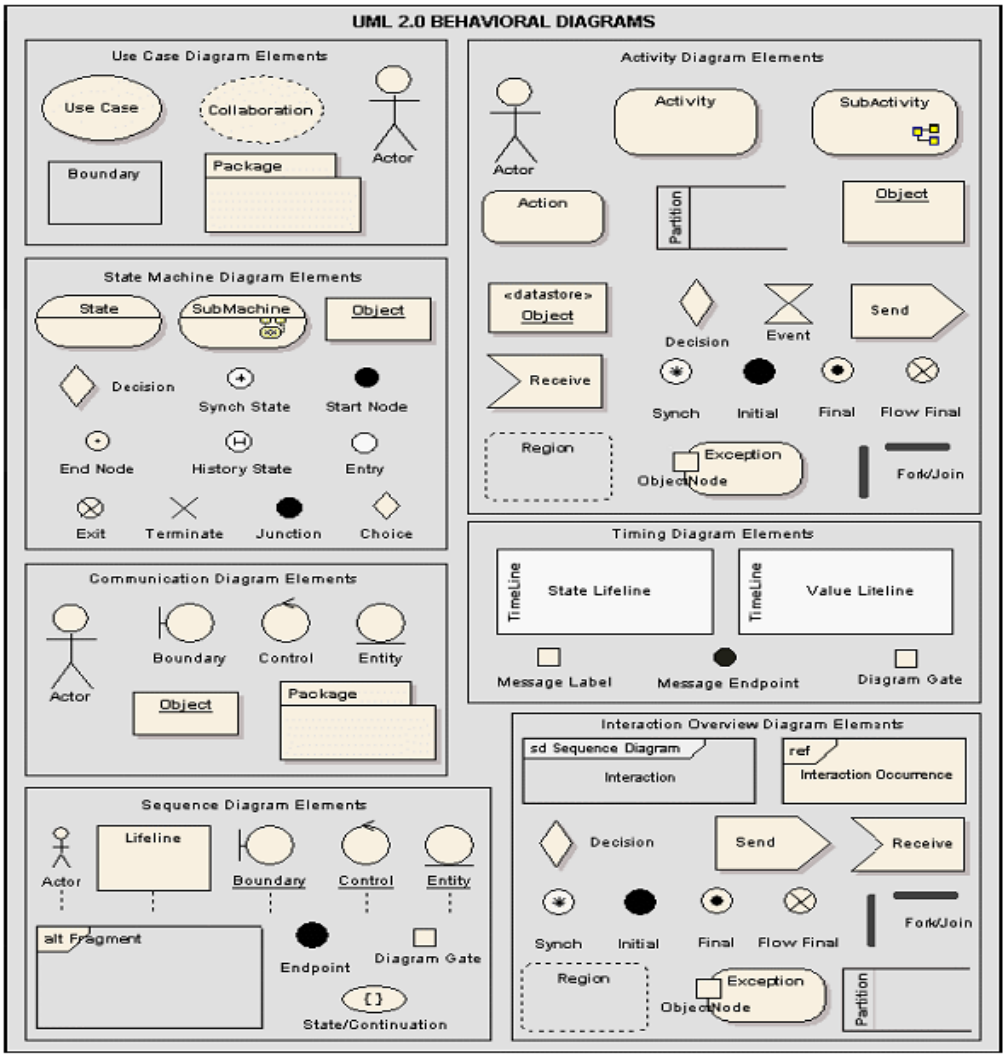
II.7. *Unified Modeling Language* (UML)

Awalnya UML dikembangkan oleh Grady Booch, James Rumbaugh dan Ivar Jacobson. Ketiga orang ini masing-masing merupakan pencipta metodologi *object-oriented Booch, Object Modeling Technique* (OMT) dan *Object-Oriented Software Engineering* (OOSE). UML adalah suatu standar dalam menentukan aturan dan notasi yang spesifik pada suatu sistem *software*. UML bukan merupakan proses penentuan untuk membuat suatu sistem *software,* karena tidak menyediakan suatu metode, proses atau suatu bahasa sederhana.



Gambar II.5: Klasifikasi Jenis *Diagram UML*

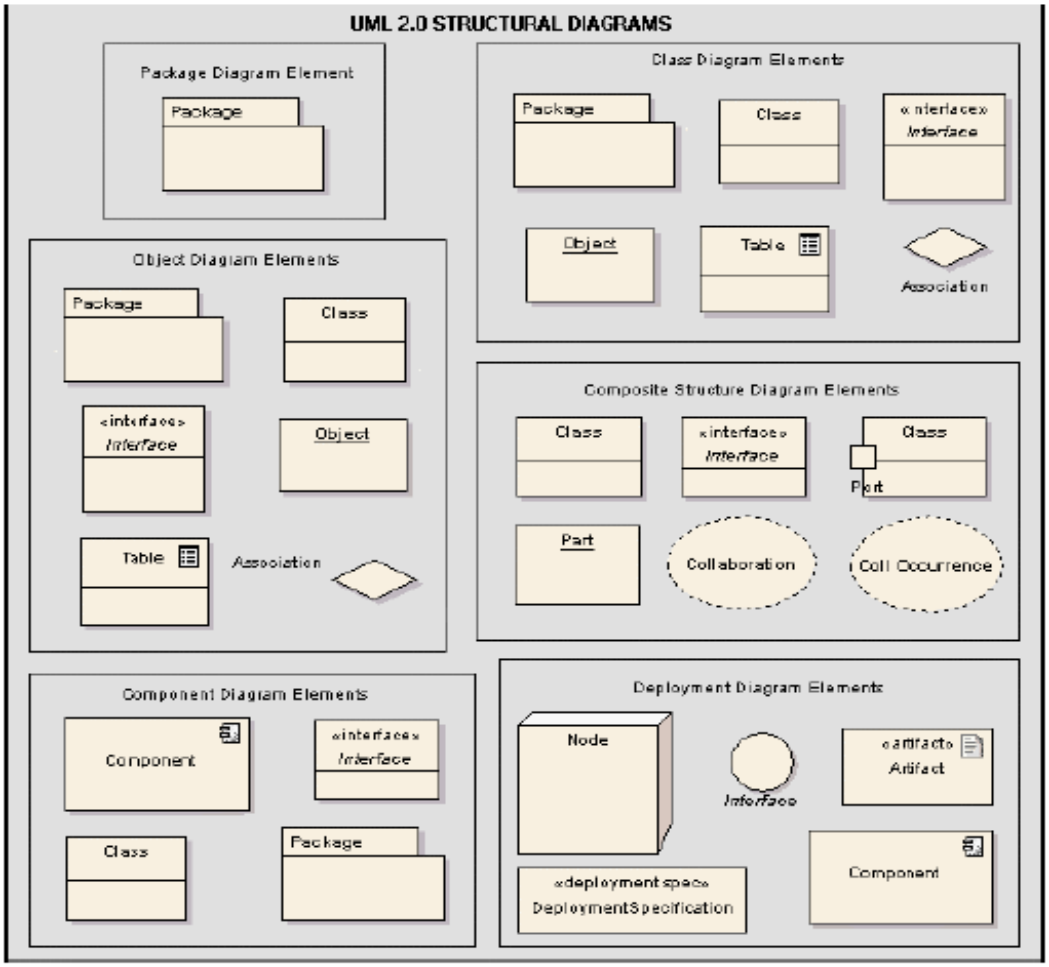
(Sumber: Tri Fajar Yurmama Supiyati, 2008: 4)



Gambar II.6: *Diagram* Prilaku

(Sumber: Tri Fajar Yurmama Supiyati, 2008: 4)

Diagram Perilaku (*behavioral*) seperti yang terlihat pada Gambar II.3 adalah diagram yang menunjukkan hal-hal yang berkaitan dengan kebiasaan atau perilaku dari suatu sistem atau *business process*.



Gambar II.7: *Diagram* Struktural

(Sumber: Tri Fajar Yurmama Supiyati, 2008: 5)

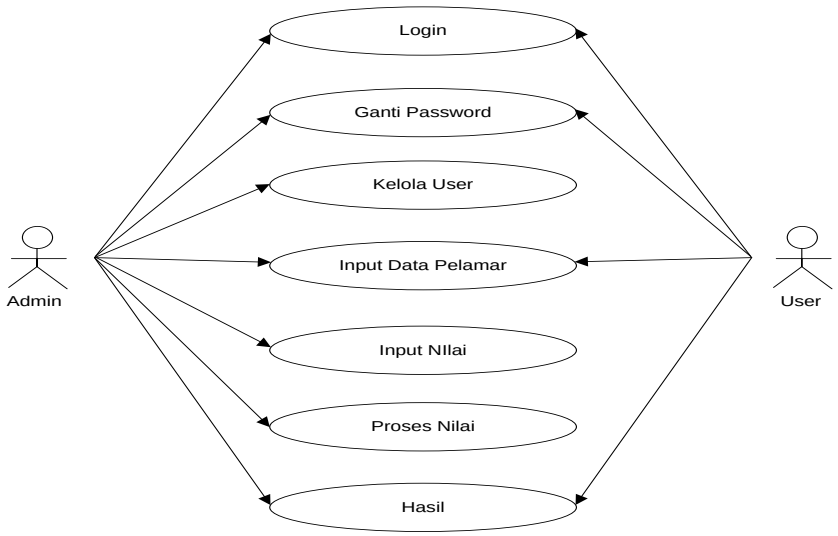
Gambar diatas merupakan diagram struktur yang membentuk suatu sistem atau fungsi di dalamnya. Diagram ini dapat merefeksikan hubungan statis dari suatu struktur, sebagaimana *class* atau *package diagrams*, atau arsitektur yang berjalan. Sedangkan dari sudut pandang pemodelannya UML terdapat 9 diagram, yaitu *Use Case*, *Class*, *Object*, *Sequence*, *Collaboration*, *Statechart*, *Activity*, *Component* dan *Deployment* (Tri fajar YS, 2008: 5).

II.7.1. *Use Case* Diagram.

Use Case dan *use case specification*; *Use case* adalah deskripsi fungsi dari sebuah sistem perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut skenario.

Use case merupakan awal yang sangat baik untuk setiap fase pengembangan berbasis objek, design, testing, dan dokumentasi yang menggambarkan kebutuhan sistem dari sudut pandang di luar sistem.

Perlu diingat bahwa *use case* hanya menetapkan apa yang seharusnya dikerjakan oleh sistem, yaitu kebutuhan fungsional sistem dan tidak untuk menentukan kebutuhan non-fungsional, misalnya: sasaran kinerja, bahasa pemrograman dan lain sebagainya. Terlihat seperti gambar II.7 berikut:



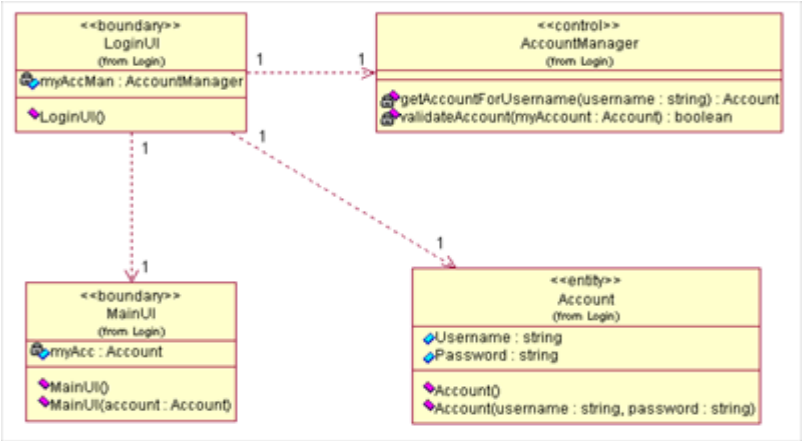
Gambar II.8: Contoh *UseCase* Diagram

(Sumber: Sri Handoko, 2012: 43)

II.7.2 *Class Diagram*

Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat. *Class* memiliki tiga area pokok :

Nama (dan *stereotype*), Atribut, Metoda. Terlihat seperti gambar II.8 berikut;

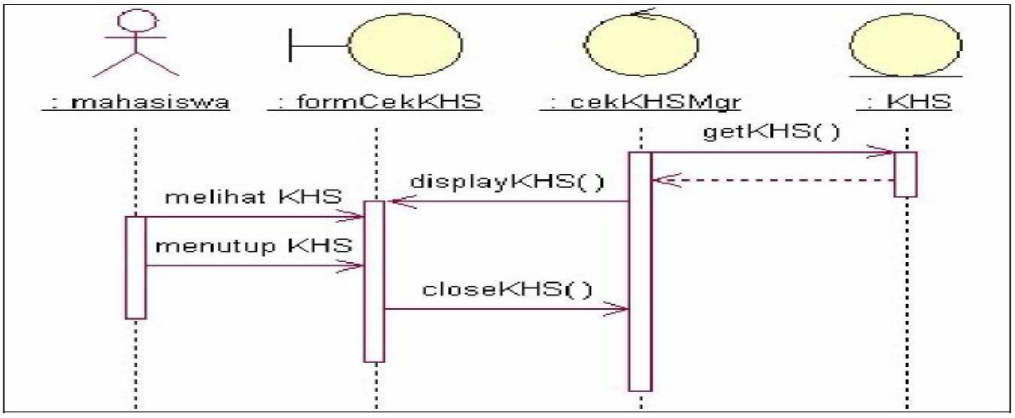


Gambar II.9: Notasi *class*

(Sumber: Haviluddin, 2011: 3)

II.7.3. *Sequence* diagram

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. 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. Terlihat seperti gambar II.9 berikut:

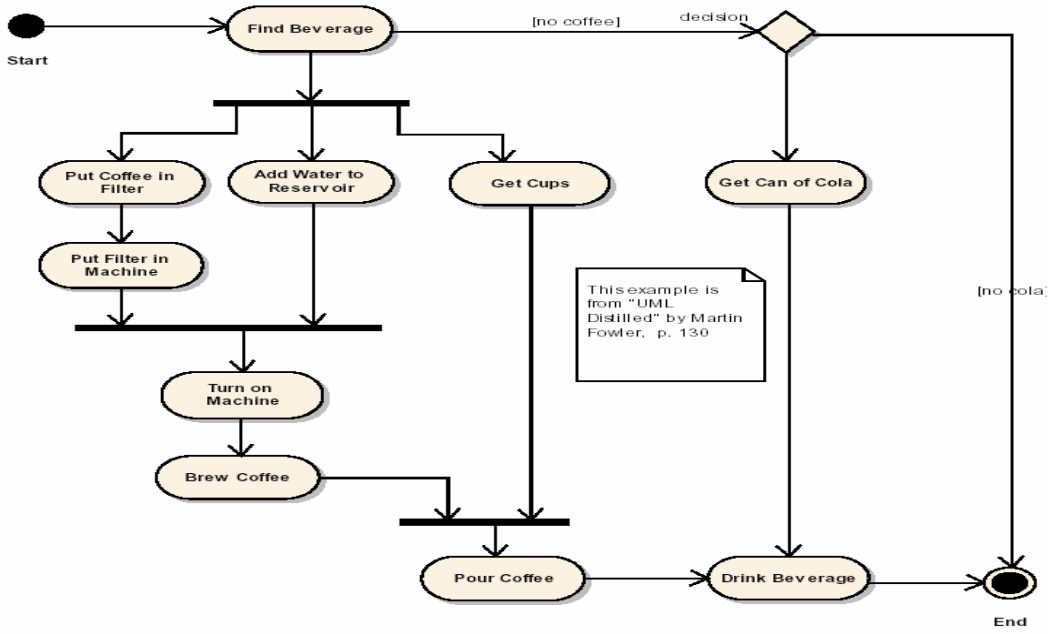


Gambar II.10: Notasi *Sequence* diagram

(Sumber: Haviluddin, 2011: 5)

II.7.4. *Interaction Overview/Activity* diagram

Interaksi *overview* diagram berfokus pada gambaran aliran kendali interaksi dimana *node* adalah interaksi atau kejadian interaksi. Terlihat seperti gambar II.9 berikut;



Gambar II.11: Notasi *Activity* diagram

(Sumber: Sri Dharwiyanti, 2003: 8)