BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Perusahaan

2.1.1 Sejarah PT. Dwikappa Asri Utama

PT. Dwikappa Asri Utama adalah salah satu perusahaan swasta di Bandung yang bergerak di bidang pengadaan alat dan jasa. Dalam usahanya PT. Dwikappa Asri Utama menerima jasa pengadaan, penyewaan dan pemeliharaan alat untuk proyek-proyek seperti *hydrotest, hottaping, cool cutting* dan *bolting*.

Hydrotest merupakan kegiatan pengukuran tekanan instalasi pipa dengan menggunakan media air, sedangkan *hottaping* merupakan kegiatan pembuatan pipa baru. *Cool cutting* merupakan pengerjaan pemotongan pipa-pipa yang digunakan dalam instalasi dan *bolting* adalah pengerjaan pengencangan baut-baut pada sambungan pipa atau antar *flange*.

PT. Dwikappa Asri Utama mengawali perjalanannya pada tahun 1996 dan sampai dengan sekarang PT. Dwikappa Asri Utama bertempat di Jl. Merdeka no. 46 Bandung.

2.1.2 Visi dan Misi PT. Dwikappa Asri Utama

PT. Dwikappa Asri Utama mempunyai visi dan misi dalam menjalankan tugasnya sebagai perusahaan yang bergerak di bidang pengadaan alat dan jasa. Adapun visi dan misi PT. Dwikappa Asri Utama adalah:

1. Visi

   Menjadi perusahaan yang terkemuka dalam pengadaan alat ukur tekanan air bawah laut serta jasa teknisinya.

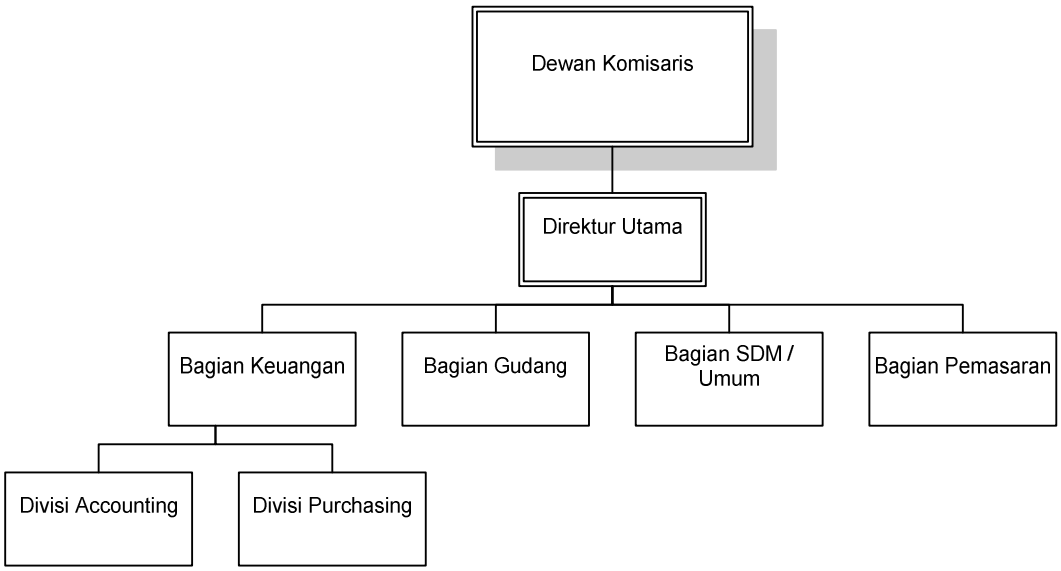
2. Misi

   Meningkatkan nilai perusahaan melalui peningkatan kualitas SDM, sistem manajemen perusahaan dan pertumbuhan profitabilitas usaha.

2.1.3 Struktur Organisasi PT. Dwikappa Asri Utama

Organisasi perusahaan sangat penting dalam menjamin kelangsungan dan kelancaran mekanisme kerja perusahaan, adanya organisasi dimaksudkan untuk menciptakan suatu sistem pembagian kerja atau tugas yang sesuai dengan kebutuhan perusahaan sehingga mempermudah kegiatan operasional perusahaan dalam mencapai tujuan.

Struktur organisasi PT. Dwikappa Asri Utama dapat dilihat pada Gambar 2.1.



Gambar 2.1 Struktur Organisasi PT. Dwikappa Asri Utama

Uraian tugas dari struktur organisasi PT. Dwikappa Asri Utama adalah sebagai berikut:

1. Direktur Utama

   Direktur utama bertugas membawahi semua struktur yang ada dibawahnya dan bertanggung jawab kepada dewan komisaris.

2. Bagian Keuangan

   Bagian keuangan bertugas mengkoordinasi masalah keluar masuknya uang dan bertanggung jawab kepada direktur utama.

   2.1. Divisi *Accounting*

   Tugas dari divisi *accounting* adalah sebagai berikut:

   a. Mencatat dan mengatur pengeluaran dan pemasukkan uang, serta membuat analisa mengenai laba rugi perusahaan.
   b. Membuat invoice kepada perusahaan yang menjadi klien perusahaan.

   2.2. Divisi *Purchasing*

   Tugas dari divisi *purchasing* adalah sebagai berikut:

   a. Mengatur segala kebutuhan akan alat serta menginventarisir alat selama dipakai maupun tidak.
   b. Mengontrol alat dalam keadaan siap pakai atau harus direparasi dahulu sebelum pemakaian.

3. Bagian Gudang

   Mengkoordinasi semua alat yang dipakai untuk pekerjaan *hydrotest, hottaping, cool cutting* dan *bolting.*

4. Bagian SDM / Umum

   Bagian SDM / umum bertugas mengurus masalah yang berhubungan dengan karyawan dan tenaga ahli.

5. Bagian Pemasaran

   Bagian pemasaran bertugas mengikuti semua tender untuk mendapatkan pekerjaan pada suatu proyek.

2.2 Landasan Teori

2.2.1 Konsep Dasar Sistem

Dalam perancangan suatu sistem informasi diarahkan kepada pemanfaatan teknologi secara maksimal yang terdiri dari beberapa elemen atau komponen yang membentuk jaringan kerja dan mempunyai tujuan yang ingin dicapai. Pendekatan yang menekankan pada prosedur, mendefinisikan sebuah sistem sebagai berikut [1]:

"Suatu kesatuan yang terdiri dari dua atau lebih komponen atau subsistem yang berinteraksi untuk mencapai tujuan".

Suatu sistem dapat terdiri dari sistem bagian (*subsystems*). Sebagai misal, sistem komputer dapat terdiri dari subsistem yang lebih kecil lagi atau terdiri dari komponen-komponen. Subsistem perangkat keras (*hardware*) dapat terdiri dari alat masukkan, alat pemroses, alat keluaran dan simpanan luar. Subsistem-subsistem saling berinteraksi dan saling berhubungan membentuk satu kesatuan hingga tujuan/sasaran sistem tersebut dapat tercapai. Interaksi dari subsistem-subsistem sedemikian rupa, sehingga dicapai suatu kesatuan yang terpadu atau terintegrasi (*integrated*).

2.2.1.1 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifat-sifat tertentu, yaitu :

1. Komponen Sistem

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponem sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem. Setiap subsistem mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan.

2. Batas Sistem

   Batas sistem (*boundary*) merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem

   Lingkungan luar (*environmant*) dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

4. Penghubung Sistem

   Penghubung (*interface*) merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya melalui penghubung ini memungkinkan sumber-sumber daya yang mengalir dari suatu subsistem

ke subsistem yang lainnya. Keluaran (*output*) dari suatu subsistem akan menjadi masukan (*input*) untuk subsistem yang lainnya dengan melalui penghubung. Dengan penghubung satu subsistem dapat berintegrasi dengan subsistem yang lainnya membentuk satu kesatuan.

5. Masukan Sistem

   Masukan (*input*) adalah energi yang dimasukkan kedalam sistem. Masukan dapat berupa masalah perawatan (*maintenance input*) dan masukkan sinyal (*signal input*). *maintenance input* adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. *Signal input* adalah energi yang diproses untuk didapatkan keluaran.

6. Keluaran Sistem

   Keluaran (*output*) adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. Misalnya untuk sistem komputer, panas yang dihasilkan adalah keluaran yang tidak berguna dan merupakan hasil sisa pembuangan, sedang informasi adalah keluaran yang dibutuhkan.

7. Pengolah Sistem

   Suatu sistem dapat mempunyai suatu bagian pengolah yang akan merubah masukkan menjadi keluaran. Suatu sistem produksi akan mengolah masukkan berupa bahan baku dan bahan-bahan yang lain menjadi keluaran berupa barang jadi.

8. Sasaran Sistem

   Suatu sistem pasti mempunyai tujuan (g*oal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak

akan ada gunanya. Sasaran dari sistem sangat menentukan sekali masukkan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem.

2.2.1.2 Klasifikasi Sistem

Sistem dapat diklasifikasikan dari beberapa sudut pandangan, diantaranya adalah sebagai berikut:

1. Sistem diklasifikasikan sebagai sistem abstrak (*abstrack system*) dan sistem fisik (*physical system*). Sistem abstrak adalah sistem yang berupa pemikiran/ide-ide yang tidak tanpak secara fisik. Sistem fisik merupakan sistem yang ada secara fisik.
2. Sistem diklasifikasikan sebagai sistem alamiah (*natural system*) sistem buatan manusia (*human made system*). Sistem alamiah adalah sistem yang terjadi melaui proses alam, tidak dibuat manusia. Sistem buatan manusia adalah sistem yang dirancang oleh manusia. Sistem buatan manusia yang melibatkan interaksi antara manusia dengan mesin disebut dengan *human-machine system* atau ada yang menyebutnya dengan *man machine system.*
3. Sistem diklasifikasikan sebagai sistem tertentu (*deterministic system*) dan sistem tak tentu (*probabilistic system*). Sistem tertentu beroperasi dengan tingkah laku yang sudah dapat diprediksi. Interaksi antara bagiannya sudah dapat dideteksi dengan pasti sehingga keluaran dari sistem dapat diramalkan. Sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilistik.
4. Sistem diklasifikasikan sebagai sistem tertutup (*closed system*) dan sistem terbuka (*open system*). Sistem tertutup merupakan sistem yang tidak

berhubungan dengan lingkungan luarnya. Secara teoritis sistem tertutup ini ada, tetapi kenyataannya tidak ada sistem yang benar-benar tertutup, yang ada hanyalah *relatively closed system* (secara relatif tertutup, tidak benar-benar tertutup). Sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya. Sistem ini menerima masukkan dan menghasilkan keluaran untuk lingkungan luar/subsistem yang lainnya.

2.2.2 Konsep Dasar Informasi

Sumber dari informasi adalah data. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata. Kejadian-kejadian (*event*) adalah sesuatu yang terjadi pada saat tertentu. Informasi dapat didefinisikan sebagai [1] :

"Data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya".

2.2.2.1 Siklus Informasi

Data merupakan bentuk yang masih mentah yang belum dapat berceritera banyak, sehingga perlu diolah lebih lanjut. Data diolah melalui suatu model untuk dihasilkan informasi.

Data yang diolah melalui suatu model menjadi suatu informasi, penerima kemudian menerima informasi tersebut, membuat suatu keputusan dan melakukan tindakan, yang berarti menghasilkan suatu tindakan yang lain yang akan membuat sejumlah data kembali. Data tersebut akan ditangkap sebagai input, diproses kembali lewat suatu model dan seterusnya membentuk suatu siklus.

2.2.2.2 Kualitas Informasi

Kualitas dari suatu informasi (*quality of information*) tergantung dari tiga hal, yaitu informasi harus akurat (*accurate*), tepat waktunya (*timeliness*) dan relevan (*relevance*).

1. Akurat, berarti informasi harus bebas dari kesalahan-kesalahan dan tidak bias atau menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya.
2. Tepat pada waktunya, berarti informasi yang datang pada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi. Bila pengambilan keputusan terlambat, maka dapat berakibat fatal untuk organisasi.
3. Relevan, berarti informasi tersebut mempunyai manfaat untuk pemakainya, dengan cara mengurangi ketidakpastian, menaikkan kemampuan untuk memprediksi, atau menegaskan ekspektasi semula.

2.2.2.3 Nilai Informasi

Nilai dari informasi (*Value of Information)* ditentukan dari dua hal yaitu manfaat dan biaya mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya. Akan tetapi perlu diperhatikan bahwa informasi yang digunakan didalam suatu sistem informasi umumnya digunakan untuk beberapa kegunaan. Sebagian informasi tidak dapat ditaksir keuntungannya dengan suatu nilai tetapi dapat ditaksir nilai keefektifannya.

2.2.3 Konsep Dasar Sistem Informasi

Informasi merupakan hal yang sangat penting didalam mengambil keputusan. Informasi tersebut didapatkan dari sistem informasi (*information system*) atau disebut juga dengan *processing systems* atau *information processing systems* atau *information generating systems.* Sistem informasi adalah [1]:

"Suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan".

2.2.3.1 Komponen Sistem Informasi

Sistem informasi terdiri dari komponen-komponen yang disebutnya dengan istilah blok bangunan (*building block*), yaitu blok masukan (*input block*), blok model (*model block*), blok keluaran (*output block*), blok teknologi (*technology block*), blok basis data (*database block*), dan blok kendali (*control block*). Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasarannya.

1. Blok Masukan

   Input mewakili data yang masuk kedalam sistem informasi, input disini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.

2. Blok Model

   Blok ini terdiri dari kombinasi prosedur, logika dan model matematik yang akan memanipulasi data input dan data yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

3. Blok Keluaran

Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.

4. Blok Teknologi

Teknologi merupakan “kotak alat” (*tool-box*) dalam sistem informasi. Teknologi digunakan untuk menrima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri dari tiga bagian utama, yaitu teknisi (*humanware* atau *brainware*), perangkat lunak (*software*), dan perangkat keras (*hardware*).

5. Blok Basis Data

Basis data (*database*) merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan dalam basis data untuk keperluan penyediaan informasi lebih lanjut.

6. Blok Kendali

Banyak hal yang dapat merusak sistem informasi, seperti misalnya bencana alam, api, temperatur, air, kecurangan-kecurangan, kegagalan-kegagalan sistem itu sendiri, kesalahan-kesalahan, sabotase dan lain sebagainya. Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat langsung cepat diatasi.

2.2.4 Basis Data

Basis data adalah kumpulan file- file yang mempunyai kaitan antara satu file dengan file lain sehingga membentuk satu bangunan data untuk menginformasikan satu perusahaan, instansi dalam batasan tertentu[2].

Basis data relasional adalah basis data yang mengorganisasikan data dalam bentuk tabel. Tabel dibentuk dengan mengelompokan data yang mempunyai subjek yang sama.

Basis data pada prinsipnya ditujukan untuk pengaturan data agar terdapat kemudahan dalam pengambilan kembali data tersebut. Berikut ini terdapat beberapa tujuan dari basis data diantaranya yaitu:

1. Kecepatan dan kemudahan *(speed).*
2. Efesiensi ruang penyimpanan *(space).*
3. Keakuratan *(accuracy).*
4. Ketersediaan *(avaibility).*
5. Kelengkapan *(completeness).*
6. Keamanan *(security).*
7. Kebersamaan *(sharability).*

Kumpulan file yang saling berkaitan bersama dengan program untuk pengelolaannya disebut sebagai DBMS *(Database Management System). Database* adalah kumpulan datanya, sedangkan program pengelolaannya berdiri sendiri dalam satu paket program yang komersial untuk membaca data, mengisi data, menghapus data, melaporkan data dalam *database.*

Beberapa definisi dalam perencanaan *database* yang perlu untuk diketahui adalah sebagai berikut:

a. *Entity* : Orang, tempat kejadian, kosep yang informasinya direkam.

b. *Attribute* : Setiap *entity* mempunyai atribut untuk mewakili satu *entity*/ atribut disebut juga sebagai data elemen, data field atau data item.

c. *Data Value* : Data aktual atau informasi yang disimpan pada tiap data elemen.

d. *Record/tufle* : Kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang *entity* secara lengkap.

e. *File* : Kumpulan *record-record* sejenis yang mempunyai panjang elemen yang sama, atribut yang sama namun berbeda *data value*nya.

2.2.4.1 Arsitektur Sistem Basis Data

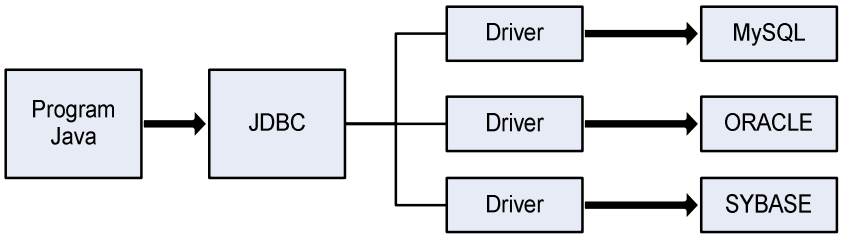
Arsitektur sistem merujuk pada konfigurasi sistem secara keseluruhan yang akan menjadi tempat hidup dari DBMS basis data dan aplikasi yang memanfaatkannya. Bagaimana tempat wujud dari tempat hidup tersebut yang juga akan menentukan bagaimana para pemakai sistem berinteraksi dengannya, sudah seharusnya ditetapkan sejak memulai perancangan basis data atau paling tidak sebelum melaksanakan tahap implementasi basis data.

Dari waktu ke waktu, sejalan dengan kemajuan teknologi, arsitektur yang dapat kita pilih semakin beragam atau semakin banyak variannya, yang berubah juga keunggulannya pada tiap jenis arsitektur tersebut. Arsitektur saat ini yang dikatakan paling baik dan banyak ditempatkan di berbagai tempat dan di masa yang akan datang dapat beralih ke arsitektur lain yang memang sama sekali baru atau merupakan pengembangan dari arsitektur yang lama, yang menjadi utama adalah menentukan arsitektur yang cocok yang tentu saja bukan hanya keunggulan teknologinya tetapi disamping itu biaya yang sesuai dengan anggaran organisasi yang akan mengaplikasikannya.

2.2.5 *Java Database Connectivity (JDBC)*

Java Database Connectivity (JDBC) adalah *Application Programming Interface (API)* yang dikembangkan oleh *Sun Microsystem* untuk menyediakan akses data universal dalam bahasa pemrograman JAVA. JDBC merupakan bagian dari Java 2 (dua) SDK *Standard Edition*. Inti JDBC adalah *package.sql* dan *javax.sql* (bagian dari J2EE). JDBC menyediakan mekanisme untuk koneksi *database*, sintaks untuk mengirimkan *query* dan melakukan transaksi serta struktur data yang mewakili hasil *query*.

Diagram JDBC seperti terlihat pada Gambar 2.2 di bawah ini.



Gambar 2.2 Diagram JDBC

JDBC merupakan koleksi dari API yang terdiri dari sekumpulan *class* dan *interface* yang ditulis dalam pemrograman Java sebagai *standart API* untuk *developer database* maupun *software database* sehingga memungkinkan pembuatan aplikasi *database* yang *portable* dengan bahasa Java (Rikyanto, 2002).

Dengan JDBC kita dapat membuat program aplikasi *database* yang dapat berganti *server* tanpa mengubah banyak kode dari program yang sudah ada sehingga memberikan portabilitas tinggi bagi aplikasi yang telah dibuat.

Dalam JDBC terdapat tujuh langkah standar untuk menangani *database*:

1) Memanggil/*load driver JDBC.*
2) Mendefinisikan koneksi *database.*
3) Melakukan koneksi *database*.

4) Membuat objek *statement*.

5) Melakukan *query dan update.*

6) Memproses hasil.

7) Menutup koneksi.

2.2.6 SQL *(Structured Query Language)*

SQL *(Structured Query Language)* merupakan standar bahasa yang digunakan untuk memanipulasi/*Data Manipulation Language (*DML*)* dan mendefinisikan/*Data Definition Language (*DDL*)* objek-objek basis data sesuai dengan permintaan *(request)* dari pengguna sistem. Dimana DML *(Data Manipulatoin Language)* adalah sekumpulan perintah-perintah *query* yang digunakan untuk melakukan manipulasi data seperti penyimpanan data ke suatu tabel, lalu kemudian mengubahnya atau menghapusnya atau hanya sekedar menampilkan saja, sedang DDL *(Data Definition Language)* merujuk pada kumpulan perintah *query* yang digunakan untuk mendefinisikan objek-objek basis data seperti pembuatan tabel basis data atau indeks primer/sekunder dan lain-lain.

2.2.7 *Tools* dan Analisis Perancangan Sistem

2.2.7.1 Pemodelan Berorientasi Objek

Pemodelan Berorientasi Objek adalah sebuah pendekatan untuk pengembangan suatu *software* dimana dalam struktur *software* tersebut didasarkan kepada interaksi objek dalam penyelesaian suatu proses/tugas[3].

2.2.7.1.1 Pengertian Objek

Dalam paradigma berbasis objek, objek adalah representasi sebuah entitas yang memiliki makna tertentu yang menjadi perhatian si pemandang.

Objek dapat menerima pesan *(message)*, mengolah data, dan mengirimkan pesan ke objek lain. Objek bersifat independen, artinya tiap objek dapat dipandang sebagai sebuah entitas mandiri yang memiliki peran atau tanggung jawab tertentu.

2.2.7.1.2 Karakteristik Pemodelan Berorientasi Objek

Pemodelan berorientasi objek memiliki karakteristik sebagai berikut:

1. *Encapsulation* (Pembungkusan)

   Variabel dan *method* dalam suatu objek dibungkus agar terlindungi.

2. *Inheritance* (Pewarisan)

   Sebuah *class* bisa mewariskan atribut dan *method*nya ke *class* yang lain. *Class* yang mewarisi disebut *superclass. Class* yang diberi warisan disebut *subclass*. Sebuah *subclass* bisa mewariskan atau berlaku sebagai *superclass* bagi *class* yang lain disebut *multilevel inheritance*.

3. *Polymorphism* (Polimorfisme – Perbedaan bentuk)

   Polimorfisme artinya penyamaran dimana suatu bentuk dapat memiliki lebih dari satu bentuk.

2.2.7.2 Unified Modeling Language (UML)

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis *object-oriented*. UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem *software*[5].

Untuk dapat memahami UML membutuhkan bentuk konsep dari sebuah bahasa model, dan mempelajari 3 (tiga) elemen utama dari UML seperti *building block*, aturan-aturan yang menyatakan bagaimana *building block* diletakkan secara bersamaan, dan beberapa mekanisme umum (*common*). Tiga macam yang terdapat dalam *building block* adalah katagori benda/*things*, hubungan, dan diagram. Benda/*things* adalah abstraksi yang pertama dalam sebuah model, hubungan sebagai alat komunikasi dari benda-benda, dan diagram sebagai kumpulan / group dari benda-benda/*things.*

1) Benda/*Things*

Benda adalah hal yang sangat mendasar dalam model UML, juga merupakan bagian paling statik dari sebuah model, serta menjelaskan elemen-elemen lainnya dari sebuah konsep dan atau fisik. Bentuk dari beberapa benda/*thing*s adalah sebagai berikut:

Pertama, adalah sebuah kelas yang diuraikan sebagai sekelompok dari *object* yang mempunyai atribut, operasi, hubungan yang semantik. Sebuah kelas mengimplementasikan satu atau lebih *interfaces.* Sebuah kelas dapat digambarkan sebagai sebuah persegi panjang, yang mempunyai sebuah nama, atribut, dan metoda pengoperasiannya.

Kedua, yang menggambarkan *'interface'* merupakan sebuah antar-muka yang menghubungkan dan melayani antar kelas dan atau elemen. *'Interface'*/antar-muka mendefinisikan sebuah set/kelompok dari spesifikasi pengoperasian, umumnya digambarkan dengan sebuah lingkaran yang disertai dengan namanya. Sebuah antar-muka berdiri sendiri dan umumnya merupakan pelengkap dari kelas atau komponen.

Ketiga, adalah *collaboration* yang didefinisikan dengan interaksi dan sebuah kumpulan/kelompok dari kelas-kelas/elemen-elemen yang bekerja secara bersama-sama. C*ollaborations* mempunyai struktur dan dimensi. Pemberian sebuah kelas memungkinkan berpartisipasi didalam beberapa *collaborations* dan digambarkan dengan sebuah '*elips*' dengan garis terpotong-potong.

Keempat, sebuah '*use case*' adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. '*use case*' digunakan untuk membentuk tingkah-laku benda/*things* dalam sebuah model serta direalisasikan oleh sebuah collaboration. Umumnya *'use case'* digambarkan dengan sebuah *'elips'* dengan garis yang solid, biasanya mengandung nama.

Kelima, sebuah node merupakan fisik dari elemen-elemen yang ada pada saat dijalankannya sebuah sistem, contohnya adalah sebuah komputer, umumnya mempunyai sedikitnya *memory* dan *processor*. Sekelompok komponen mungkin terletak pada sebuah *node* dan juga mungkin akan berpindah dari node satu ke node lainnya. Umumnya node ini digambarkan seperti kubus serta hanya mengandung namanya.

2) Hubungan / *Relationship*

Terdapat empat macam hubungan didalam penggunaan UML, yaitu; *dependency, association, generalization,* dan *realization.*

Pertama, sebuah *dependency* adalah hubungan semantik antara dua benda/things yang mana sebuah benda berubah mengakibatkan benda satunya akan berubah pula. Umumnya sebuah *dependency* digambarkan sebuah panah dengan garis terputus-putus.

Kedua, sebuah *association* adalah hubungan antar benda struktural yang terhubung diantara obyek. Kesatuan obyek yang terhubung merupakan hubungan khusus, yang menggambarkan sebuah hubungan struktural diantara seluruh atau sebagian. Umumnya *assosiation* digambarkan dengan sebuah garis yang dilengkapi dengan sebuah label, nama, dan status hubungannya.

Ketiga, sebuah *generalization* adalah menggambarkan hubungan khusus dalam obyek anak/*child* yang menggantikan obyek *parent* / induk . Dalam hal ini, obyek anak memberikan pengaruhnya dalam hal struktur dan tingkah lakunya kepada obyek induk.

Keempat, sebuah *realization* merupakan hubungan semantik antara pengelompokkan yang menjamin adanya ikatan diantaranya. Hubungan ini dapat diwujudkan diantara *interface* dan kelas atau *elements*, serta antara *use cases* dan *collaborations*.

3) Diagram

UML terdiri atas pengelompokkan diagram-diagram sistem menurut aspek atau sudut pandang tertentu. Diagram adalah yang menggambarkan permasalahan maupun solusi dari permasalahan suatu model.

Diagram-diagram dari UML adalah *use-case, class, object, state, sequence, collaboration, activity, component,* dan *deployment diagram.*

Berikut ini adalah penjelasan mengenai berbagai diagram UML serta tujuannya[8]:

1. Model *Use Case* Diagram

   Use Case Diagram secara grafis menggambarkan interaksi antara sistem, sistem eksternal, dan pengguna. Dengan kata lain *Use Case* diagram

secara grafis mendeskripsikan siapa yang akan menggunakan sistem dan dalam cara apa pengguna (*user*) mengharapkan interaksi dengan sistem itu. *Use Case* secara naratif digunakan untuk secara tekstual menggambarkan sekuensi langkah-langkah dari setiap interaksi.

2. Diagram Struktur Statis

   UML menawarkan dua diagram untuk memodelkan struktur statis sistem informasi, yaitu:

   a. *Class Diagram*: menggambarkan struktur objek sistem. Diagram ini menunjukan *class object* yang menyusun sistem dan juga hubungan antara *class object* tersebut.

   b. *Object Diagram*: serupa dengan *class diagram*, tetapi objek diagram memodelkan *instance object actual* dengan menunjukan nilai-nilai saat ini dari atribut instance. *Object Diagram* menyajikan "snapshot/potret" tentang objek sistem pada point waktu tertentu. Diagram ini tidak digunakan sesering Class Diagram, tetapi saat digunakan dapat membantu seorang *developer* memahami struktur sistem secara lebih baik.

3. Diagram Interaksi

   Diagram interaksi memodelkan sebuah interaksi, terdiri dari satu set objek, hubungan-hubungannya, dan pesan yang terkirim di antara objek. Model diagram ini memodelkan *behavior* (kelakuan) sistem yang dinamis dan UML memiliki dua diagram untuk tujuan ini, yaitu:

   a. Diagram rangkaian/*Sequence Diagram*: secara grafis menggambarkan bagaimana objek berinteraksi dengan satu sama

lain melalui pesan pada sekuensi sebuah *use case* atau operasi. Diagram ini mengilustrasikan bagaimana pesan terkirim dan diterima di antara objek dan dalam sekuensi atau timing apa.

b. Diagram kolaborasi: serupa dengan diagram rangkaian/sekuensi, tetapi tidak fokus pada timing atau sekuensi pesan. Diagram ini justru menggambarkan interaksi (atau kolaborasi) antara objek dalam sebuah format jaringan. Diagram rangkaian maupun diagram kolaborasi merupakan isomorphic artinya kita dapat mengubah dari satu diagram ke diagram lain.

4. Diagram *State*

UML memiliki sebuah diagram untuk memodelkan behavior objek khusus yang kompleks (*statechart*) dan sebuah diagram untuk memodelkan behavior dari sebuah use case atau sebuah metode, yaitu:

a. Diagram *statechart*: digunakan untuk memodelkan behavior objek khusus yang dinamis. Diagram ini mengilustrasikan siklus hidup objek-berbagai keadaan yang dapat diasumsikan oleh objek dan *event-event* (kejadian) yang menyebabkan objek beralih dari satu state ke state lain.

b. Diagram aktivitas/*Activity Diagram*: secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas baik proses bisnis maupun *use case*. *Activity diagram* dapat juga digunakan untuk memodelkan *action* yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari *action* tersebut.

5. Diagram Implementasi

   Diagram implementasi juga memodelkan struktur sistem informasi, yaitu:

   a. Diagram komponen/*Component Diagram*: digunakan untuk menggambarkan organisasi dan ketergantungan komponen-komponen software sistem. Komponen diagram dapat digunakan untuk menunjukan bagaimana kode pemrograman dibagi menjadi modul-modul (atau komponen).

   b. Diagram penguraian/*Deployment*: digunakan untuk mendeskripsikan arsitektur fisik dalam istilah ”node” untuk hardware dan software dalam sistem. Diagram ini menggambarkan konfigurasi komponen-komponen software real-time, prosesor, dan peralatan yang membentuk arsitektur sistem.

Dalam tahap perancangan suatu sistem diperlukan adanya teknik-teknik penyusunan sistem untuk menganalisa dan mendokumentasikan data yang mengalir didalam sistem tersebut.

2.2.8 *Fuzzy Multiple Attribute Decision Making* (FMADM)

Fuzzy Multiple Attribute Decision Making (FMADM) adalah suatu metode yang digunakan untuk mencari alternatif optimal dari sejumlah alternatif dengan kriteria tertentu. Inti dari FMADM adalah menentukan nilai bobot untuk setiap atribut, kemudian dilanjutkan dengan proses perankingan yang akan menyeleksi alternatif yang sudah diberikan. Pada dasarnya, ada 3 pendekatan untuk mencari nilai bobot atribut, yaitu pendekatan subyektif, pendekatan obyektif dan pendekatan integrasi antara subyektif & obyektif. Masing-masing pendekatan

memiliki kelebihan dan kelemahan. Pada pendekatan subyektif, nilai bobot ditentukan berdasarkan subyektifitas dari para pengambil keputusan, sehingga beberapa faktor dalam proses perankingan alternatif bisa ditentukan secara bebas. Sedangkan pada pendekatan obyektif, nilai bobot dihitung secara matematis sehingga mengabaikan subyektifitas dari pengambil keputusan. (Kusumadewi, 2007).

Ada beberapa metode yang dapat digunakan untuk mnyelesaikan masalah FMADM antara lain (Kusumadewi, 2006):

a. *Simple Additive Weighting Method* (SAW)

b. *Weighted Product* (WP)

c. ELECTRE

d. *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS)

e. *Analytic Hierarchy Process* (AHP)

2.2.8.1 Algoritma FMADM

Algoritma FMADM adalah sebagai berikut:

1. Memberikan nilai setiap alternatif (A_i) pada setiap kriteria (C_j) yang sudah ditentukan, dimana nilai tersebut diperoleh berdasarkan nilai *crisp*; i=1,2,…m dan j=1,2,…n.
2. Memberikan nilai bobot (W) yang juga didapatkan berdasarkan nilai *crisp*.
3. Melakukan normalisasi matriks dengan cara menghitung nilai rating kinerja ternormalisasi (r_{ij}) dari alternatif A_i pada atribut C_j berdasarkan persamaan yang disesuaikan dengan jenis atribut (atribut keuntungan/benefit=maksimum atau atribut biaya/cost=minimum). Apabila berupa artibut keuntungan maka nilai *crisp* (X_{ij}) dari setiap kolom atribut dibagi dengan nilai *crisp* MAX

(MAX X_{ij}) dari tiap kolom, sedangkan untuk atribut biaya, nilai *crisp* MIN (MIN X_{ij}) dari tiap kolom atribut dibagi dengan nilai *crisp* (X_{ij}) setiap kolom.

4. Melakukan proses perankingan dengan cara mengalikan matriks ternormalisasi (R) dengan nilai bobot (W).
5. Menentukan nilai preferensi untuk setiap alternatif (V_i) dengan cara menjumlahkan hasil kali antara matriks ternormalisasi (R) dengan nilai bobot (W). Nilai V_i yang lebih besar mengindikasikan bahwa alternatif A_i lebih terpilih. (Kusumadewi, 2007).

2.2.8.2 Langkah Penyelesaian

Dalam penelitian ini menggunakan FMADM metode *Simple Additive Weigthing* (SAW). Adapun langkah-langkahnya adalah:

1. Menentukan kriteria-kriteria yang akan dijadikan acuan dalam pengambilan keputusan, yaitu C_i.
2. Menentukan rating kecocokan setiap alternatif pada setiap kriteria.
3. Membuat matriks keputusan berdasarkan kriteria (C_i), kemudian melakukan normalisasi matriks berdasarkan persamaan yang disesuaikan dengan jenis atribut sehingga diperoleh matriks ternormalisasi R.
4. Hasil akhir diperoleh dari proses perankingan yaitu penjumlahan dari perkalian matriks ternormalisasi R dengan vektor bobot sehingga diperoleh nilai terbesar yang dipilih sebagai alternatif terbaik (A_i) sebagai solusi. (Kusumadewi, 2006).

2.2.9 Metode *Simple Additive Weigthing* (SAW)

Metode *Simple Additive Weigthing* (SAW) sering juga dikenal istilah metode penjumlahan terbobot. Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut. Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada.

$$r_{ij} = \begin{cases} \dfrac{x_{ij}}{\underset{i}{Max}\, x_{ij}} & \textit{jika } j \text{ adalah atribut keuntungan (benefit)} \\ \dfrac{\underset{i}{Min}\, x_{ij}}{x_{ij}} & \text{jika } j \text{ adalah atribut biaya (cost)} \end{cases} \qquad (2.1)$$

Dimana r_{ij} adalah rating kinerja ternormalisasi dari alternatif A_i pada atribut C_j; i=1,2,...,m dan j=1,2,...,n.

Nilai preferensi untuk setiap alternatif (V_i)diberikan sebagai:

$$V_i = \sum_{j=1}^{n} w_j r_{ij} \qquad (2.2)$$

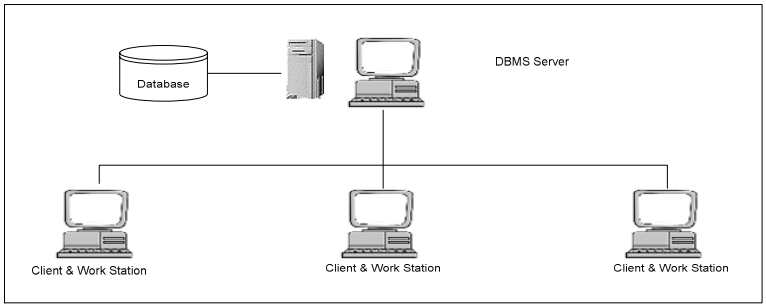
Nilai V_i yang lebih besar mengindikasikan bahwa alternatif A_i lebih terpilih.

2.2.10 Sistem *Client-Server*

Sistem *Client Server* atau disebut juga sistem tersentralisasi diterapkan pada sebuah sistem jaringan. Sistem *Client Server* ini ditujukan untuk mengatasi kelemahan-kelemahan yang terdapat pada sistem sebelumnya. Sistem *Client Server* terdiri dari dua komponen utama yaitu *client* dan *server*. *Client* berisi aplikasi basis data dan *server* berisi DBMS dan basis data. Setiap aktivitas yang dikehendaki para pemakai akan lebih dulu ditangani oleh *client. Client* selanjutnya mengupayakan agar semua proses “sebisa mungkin” ditangani sendiri.

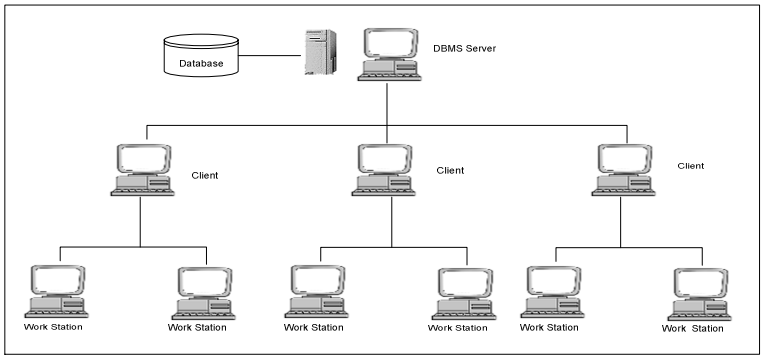
Bila ada proses yang harus melibatkan data yang tersimpan pada basis data barulah *client* mengadakan hubungan dengan *server*.

Pada sistem *Client Server* untuk memenuhi kebutuhan, *client* akan mengirimkan *message* (perintah) *query* pengambilan data. Selanjutnya, *server* yang menerima *message* tersebut akan menjalankan *query* tersebut dan hasilnya akan dikirimkan kembali ke *client*. Dengan begitu transfer datanya jauh lebih efisien. Adapun bentuk dari Sistem *Client Server* yang sederhan adalah sebagai berikut :



Gambar 2.3. Sistem *Client Server* Sederhana

Disamping bentuk *client server* sederhana terdapat pula bentuk *client server* yang lebih komplek yang digamarkan sebagai berikut :



Gambar 2.4. Sistem *Client Server* Kompleks

Dari kedua gambar diatas, dapat dilihat adanya dua macam implementasi sistem *client server*. Bentuk yang sederhana dapat diterapkan pada sebuah

jaringan komputer lokal (LAN) dimana fungsi *client* (untuk menangani sebagian besar proses pengolahan data seperti perhitungan, perulangan, pembandingan, dan lain-lain.) dan fungsi *work station* (untuk menangani interaksi dengan pemakai, menerima data masukan dan menayangkan hasil pengolahan) disatukan.

2.2.11 Software Pendukung

2.2.11.1 *Eclipse*

Eclipse adalah IDE (*integrated development environment*) *tools Java editor* profesional, untuk mendesain, melakukan *coding,* dan pengembangan dari *JSE, J2EE,* dan *web application. Tools* ini memiliki fitur *visual editing* dengan menambahkan *plugin WindowBuilder* yang memungkinkan *user* membuat aplikasi tanpa harus menulis barisan kode *Java*. *Eclipse* membantu membangun aplikasi desktop dan *web applications* dengan menggunakan bahasa pemrograman *Java.*

2.2.11.2 MySql

SQL adalah sebuah konsep pengoprasian database, terutama untuk pemilihan/seleksi dan pemasukan data, yang memungkinkan pengoprasian data dikerjakan dengan mudah secara otomatis. Salah satu aplikasi *database* yang menggunakan bahasa SQL adalah *MySql*.

MySql adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public Lisence*), dimana setiap orang bebas untuk menggunakan *MySql,* namun tidak boleh dijadikan produk turunan yang bersifat *closed source*/komersial.

MySql sebenarnya merupakan turunan salah satu konsep utama SQL dalam *database* sejak lama.

2.2.11.2.1 Perintah-Perintah SQL

1. Menambah Data (Insert)

Perintah SQL yang digunakan untuk menambah data ke database adalah perintah INSERT. Perintah ini mempunyai bentuk umum sebagai berikut:

INSERT [INTO] nama_tabel [(daftar field/kolom)] VALUES (daftar_data)

2. Memilih/Mengakses Data (Select)

Perintah SQL yang digunakan untuk memilih data adalah perintah SELECT. Bentuk umum perintah ini adalah sebagai berikut:

SELECT daftar_select FROM daftar_tabel

[WHERE kondisi_pencarian]

[GROUP BY daftar_group_by]

[HAVING kondisi_pencarian]

[ORDER BY daftar_order[ASC │ DESC]]

3. Mengedit/Mengubah Data (Update)

Perintah SQL yang digunakan untuk mengedit data adalah perintah UPDATE SET. Perintah ini mempunyai bentuk umum sebagai berikut:

UPDATE nama_table SET field1=databaru1 [,data2=databaru2]

[WHERE kondisi_update]

4. Menghapus Data (Delete)

Perintah SQL yang digunakan untuk menghapus data adalah perintah DELETE FROM. Perintah ini mempunyai bentuk umum sebagai berikut:

DELETE FROM tabel_hapus WHERE kondisi_hapus