BAB II

LANDASAN TEORI

2.1 Sistem Pendukung Keputusan

Sistem Pendukung Keputusan (SPK) 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 seharusnya keputusan dibuat.[4]

SPK biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk mengevaluasi suatu peluang. SPK yang seperti itu disebut aplikasi SPK. Aplikasi SPK digunakan dalam pengambilan keputusan. Aplikasi SPK menggunakan CBIS (*Computer Based Information System*) yang fleksibel, interaktif, dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi atas masalah manajemen spesifik yang tidak terstruktur. Aplikasi SPK menggunakan data, memberikan antarmuka pengguna yang mudah, dan dapat menggabungkan pemikiran pengambil keputusan. [4]

SPK lebih ditujukan untuk mendukung manajemen dalam melakukan pekerjaan yang bersifat analitis dalam situasi yang kurang terstruktur. SPK tidak dimaksudkan untuk mengotomatisasikan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambil keputusan untuk melakukan berbagai analisis menggunakan model-model yang tersedia.

Tujuan dari SPK adalah:[4]

a. Membantu manajer dalam pengambilan keputusan atas masalah semi-terstruktur.
b. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.
c. Meningkatkan efektivitas keputusan yang diambil manajer lebih daripada perbaikan efisiensinya.
d. Kecepatan komputasi.
e. Peningkatan produktivitas.
f. Dukungan kualitas.
g. Berdaya saing.
h. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan.

2.2 Software Pendukung

2.2.1 Netbeans 5.5

Netbeans sebagai *IDE* ditujukan untuk memudahkan pemrograman Java. Dalam Netbeans, pemrograman dilakukan berbasiskan visual dan *event driven*. Seperti *IDE* lain, misalnya Borland Delphi dan Microsoft Visual Studio. Untuk membuat dialog atau *User Interface*, kita tidak perlu membuat teks program secara manual baris per baris, tetapi cukup klik pada *component pallete*. Teks program akan dihasilkan secara otomatis.

Untuk memakai Netbeans *IDE* maka kita harus terlebih dahulu mempunyai driver JDK yang akan mendukung pembuatan perangkat lunak

dengan menggunakan Netbeans *IDE*, maka sebelum meng*install* Netbeans terlebih dahulu kita harus menginstal driver JDK. Netbeans yang dipakai pada pengembangan perangkat lunak ini adalah Netbeans 5.5 dan JDK 1.6.

2.2.2 MySQL

MySQL adalah perangkat lunak *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi *GPL* (*General Public License*). Dimana setiap orang bebas untuk menggunakannya, tapi tidak boleh dijadikan produk turunan yang bersifat *Closed Source* atau komersial.

MySQL sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu *SQL* (*Structure Query Language*). *SQL* adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Keandalan suatu sistem *database* (DBMS) dapat diketahui dari cara kerja *Optimizer*-nya dalam melakukan proses perintah-perintah *SQL*, yang dibuat oleh *user* maupun program-program aplikasinya. Sebagai *database server*, MySQL dapat dikatakan lebih unggul dibanding *database server* lainnya dalam *query* data. Hal ini terbukti untuk *query* yang dilakukan oleh *single user*, kecepatan *query* MySQL bisa sepuluh kali lipat lebih cepat dari PostgreSQL dan lima kali lebih cepat dibanding Interbase.

2.3 Metode *Profile Matching*

Masalah yang sering terjadi dalam proses penilaian kinerja karyawan diantaranya adalah subjektivitas pengambilan keputusan, terutama jika beberapa karyawan yang ada memiliki kemampuan (dan beberapa pertimbangan lain) yang tidak jauh berbeda. Jika proses pengambilan keputusan tersebut dibantu oleh sebuah sistem pendukung keputusan yang terkomputerisasi, subjektivitas dalam pengambilan keputusan diharapkan bisa dikurangi dan diganti dengan pelaksanaan seluruh kriteria bagi seluuh karyawan. Dengan demikian, karyawan dengan kemampuan (dan pertimbangan lain) terbaliklah yang diharapkan akan terpilih. Konsep dari model ini adalah mencari oang yang memiliki profil sedekat mungkin dengan profil jabatan yang kosong [4].

Maksud dari pencocokan profil (*profile matching)*, adalah sebuah mekanisme pengambilan keputusan dengan mengasumsikan bahwa terdapat tingkat variabel prediktor yang ideal yang harus dimiliki oleh pelamar, bukannya tingkat minimal yang harus dipenuhi atau dilewati. Dalam pencocokan profil dilakukan identifikasi terhadap semua kelompok karyawan. Para karyawan dalam kelompok tersebut diukur menggunakan beberapa kriteia penilaian. Jikalau pelaksana yang baik memperoleh skor yang berbeda dari pelaksana yang buruk atas sebuah karakteristik, maka variabel tersebut berfaedah untuk memilih pelaksana yang baik. Begitu beberapa variabel yang membedakan antara pelaksana-pelaksana yang baik dan buruk telah teridentifikasi, profil ideal dari karyawan yang berhasil bisa dibuat [4].

2.4 Spesifikasi *Five-Fold Grading System*

Five Fold Grading System merupakan spesifikasi perorangan yang mendeskripsikan seseorang yang ideal pada suatu jabatan tertentu. Berbagai karakteristik dari perorangan diidentifikasi seperti pengetahuan, kecerdasan, keterampilan, pengalaman, umur, karakter fisik dan kepribadian. Spesifikasi ini di buat pertama kali oleh Munro Fraser dan kemudian dijadikan standar bagi banyak perusahaan didunia. Karakteristik yang terdapat pada *Five Fold Grading System* adalah:

1. Pengaruh kepada yang lain : penampilan fisik, penampilan, cara bicara, sikap.
2. Kualifikasi yang dimiliki : pendidikan, pelatihan, keterampilan, pengalaman kerja.
3. Kemampuan bawaan (bakat) : kecepatan memahami dan bakat dalam belajar.
4. Motivasi : tujuan individu, konsistensi, kemantapan dalam mencapai tujuan, rata-rata keberhasilan.
5. Penyesuaian : kestabilan emosional, kemampuan untuk menanggulangi tugas, kemampuan untuk menyesusikan diri

2.5 Basis Data

Basis data adalah kumpulan data yang (arsip) yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (*redudansi*) yang tidak perlu, untuk memenuhi berbagai kebutuhan. Atau bisa diartikan sebagai kumpulan *file*/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

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. Efisiensi ruang penyimpanan (*Space)*
3. Keakuratan (*Accuracy)*
4. Ketersediaan (*Avaibility)*
5. Kelengkapan (*Completeness)*
6. Keamanan (*Security*)
7. Kebersamaan (*Sharability*)

2.6 *Unified Modelling Language (UML)*

Unified Modeling Language (UML) merupakan standar yang relatif terbuka yang dikendalikan oleh *OMG* (*Object Menegement Group*), sebuah konsorium terbuka yang terdiri dari banyak perusahaan. OMG dibentuk untuk membuat standar-standar yang mendukung interoperabilitas sistem berorientasi objek. UML lahir dari penggabungan bahasa pemodelan grafis pada akhir 1980-an dan awal 1990-an.

UML terdiri dari 13 jenis diagram resmi dan mengklasifikasikan mereka, meskipun jenis-jenis diagram ini merupakan cara orang-orang memperlakukan UML, para perancang UML tidak memandang diagram sebagai bagian yang sentral. Dan hasilnya, jenis-jenis diagram bukanlah hal yang mutlak.

1. Use Case Diagram

Use case diagram menggambarkan sebuah fungsi yang dibutuhkan oleh sebuah sistem. Dalam hal ini ada kondisi yang agak beda, yaitu disini tingkah laku dan domain sistem. Sebuah use case merepresentasikan sebuah interaksi antara pengguna dengan sebuah sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, membuat sebuah daftar aktifitas, dan sebagainya.

Notasi-notasi yang digunakan pada use-case diagram dapat dilihat pada daftar simbol.

2. Class Diagram

Class merupakan inti dari pengembangan dan desain berorientasi objek, karena nantinya class ini akan menghasilkan sebuah objek. Class menggambarkan keadaan (atribut/property) suatu sistem dan metode/fungsi yang ada pada sistem tersebut.

Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga domain yaitu : Nama (dan stereotype), Atribut dan Metode.

Atribut dan metode dapat memiliki salah satu sifat berikut :

1. Private, tidak dapat dipanggil diluar class yang bersangkutan
2. Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya.
3. Public, dapat dipanggil oleh siapa saja.

Notasi-notasi yang digunakan pada Class Diagram dapat dilihat pada daftar simbol.

2.7 *Object Oriented Programming (OOP)*

OOP merupakan terobosan baru dalam pemrograman setelah pemrograman prosedural, pemrograman modular dan pemrograman abstraksi data. Ada beberapa konsep utama OOP antara lain :

Pemrograman berorientasi-objek menekankan konsep berikut:

1. Class : kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh *'class of dog*' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing.
2. Objek : membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

3. Abstraksi : kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.
4. Enkapsulasi : memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.
5. *Polimorfisme* : melalui pengiriman pesan. tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia

akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

6. *Inheritance* - Mengatur polimorfisme dan enskapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada, objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut(terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bagian administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat

menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek -objek yang ada karena setiap objek memiliki *job description* nya sendiri.