BAB II
LANDASAN TEORI

2.1 Sistem

Terdapat dua kelompok pendekatan didalam mendefinisikan sistem, yaitu yang menekankan pada prosedurnya dan yang menekankan pada komponen atau elemennya.

Menurut Djon Irwanto (2005 : 2) Sistem adalah sekumpulan komponen yang mengimplementasi model dan fungsionalitas yang dibutuhkan. Komponen-komponen tersebut saling berinteraksi didalam sistem guna mentrans*formasi* *input* yang diberikan kepada sistem tersebut menjadi *output* yang berguna dan bernilai bagi *actor*-nya.

Menurut James (2006 : 29) **:** Sistem adalah sekelompok komponen yang saling berhubungan, bekerja bersama untuk mencapai tujuan bersama dengan menerima *input* serta menghasikan *output* dalam proses *transformasi* yang teratur.

Pendapat yang lebih menekankan pada elemen atau komponenya mendefinisikan sistem sebagai berikut. Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. Kedua kelompok definisi tersebut adalah benar dan tidak bertentangan, yang berbeda adalah cara pendekatannya.

Menurut Chr.Jimmy L.Gaol (2008 : 9) : Sistem adalah hubungan satu unit dengan unit-unit lainnya yang saling berhubungan erat dan tidak dapat dipisahkan serta menuju suatu kesatuan dalam rangka mencapai tujuan yang telah ditetapkan.

2.1.1 Elemen Sistem

Elemen sistem adalah bagian terkecil dari sebuah sistem yang diidentifikasi.

Adapun elemen-elemen yang menyusun sebuah sistem terdiri dari :

1. Tujuan

   Merupakan tujuan dari sistem tersebut berupa tujuan usaha, kebutuhan, masalah, dan prosedur pencapaian tujuan.

2. Batasan

   Merupakan batasan-batasan yang ada dalam mencapai tujuan dari sistem, dimana batasan ini dapat berupa peraturan-peraturan, biaya-biaya, personil dan peralatan.

3. *Control*

   Merupakan pengawasan dan pelaksanaan pencapaian tujuan sistem yang di dapat berupa *control* masukan data (*input*), keluaran data (*output),* dan *control* pengoperasian.

4. *Input*

   Merupakan bagian dari sistem yang bertugas untuk menerima data masukan dimana data berupa asal masukan, frekwensi pemasukan data, dan jenis pemasukan data.

5. Proses

   Merupakan bagian yang memproses masukan data menjadi in*form*asi sesuai dengan keinginan penerima, proses dapat berupa klarifikasi, peringkasan maupun pencarian.

6. *Output*

   Merupakan keluaran atau tujuan akhir dari sistem, *ouput* dapat berupa laporan atau grafik.

7. Umpan Balik

   Merupakan elemen-elemen sistem yang tugasnya adalah mengetahui apakah sistem sudah sesuai dengan yang diinginkan. Umpan balik bisa berupa perbaikan ataupun pemeliharaan.

2.1.2 Karakteristik Sistem

Sebuah sistem memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik yang dimaksud adalah sebagai berikut :

1. Komponen Sistem (*Component)*

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling bekerja sama membentuk satu kesatuan.

2. Batasan Sistem (*Boundary)*

   Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lain 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 di luar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut lingkungan luar sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

4. Penghubung Sistem *(Interface)*

   Media yang menghubungkan sistem dengan subsistem lain disebut penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya yang mengalir dari subsistem ke subsistem lain. Bentuk keluaran dari subsistem akan menjadi masukan untuk subsistem lain melalui penghubung tesebut.

5. Masukan Sistem (*Input)*

   Energi yang dimasukkan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal (*signal input*). Contoh, di dalam suatu unit sistem komputer. "Program" adalah *maintanance input* yang digunakan untuk mengoperasikan komputernya dan "data" adalah *signal input* untuk diolah menjadi informasi.

6. Keluaran Sistem (*Output)*

   Hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain. Contoh, sistem i*nform*asi. Keluaran yang dihasilkan adalah i*nform*asi.

7. Pengolah Sistem (Proses)

   Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Contoh, sistem akuntansi. Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran Sistem (*Objective)*

   Suatu sistem memiliki tujuan dan sasaran yang pasti. Kalau suatu sistem tidak memilki sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan.

2.1.3 Tipe Sistem

Sistem dapat diklasifikasikan dari beberapa sudut pandangan, diantarnya :

1. Sistem abstrak dan sistem fisik

   Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik, misalnya sistem teologia, yaitu sistem yang berupa pemikiran hubungan antara manusia dengan Tuhan, sedangkan sistem fisik merupakan sistem yang ada secara fisik, misalnya sistem komputer, sistem produksi, sistem penjualan, dan sistem administrasi personalia.

2. Sistem alamiah dan sistem buatan manusia

   Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat oleh manusia, misalnya sistem perputaran bumi. Sedangkan sistem buatan manusia merupakan sistem yang melibatkan interkasi manusia dengan mesin, yang disebut dengan *human machine system.* Sistem

*inform*asi berbasis komputer merupakan contoh *human machine system* karena menyangkut penggunaan komputer yang berinteraksi dengan manusia.

3. Sistem deterministik dan sistem probabilistik

   Sistem deterministik adalah sistem yang beroperasi dengan tingkah laku yang dapat diprediksi. Sedangkan sistem probabilistik adalah sistem yang kondisi masa depannya tidak dapat diprediksi.

4. Sistem terbuka dan sistem tertutup

   Sistem terbuka adalah sistem yang berhubungan dan dipengaruhi oleh lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk subsistem lainnya. Sedangkan sistem tertutup adalah sistem yang tidak berhubungan dan tidak terpengaruh oleh lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa campur tangan pihak luar.

2.2 Informasi

*Inform*asi sebagai salah satu sumber daya yang tersedia bagi seseorang dan memiliki nilai yang sama dengan suber daya lainnya. "*Inform*asi adalah hasil pengolahan data yang memberikan arti dan manfaat." Azhar Susanto (2006 : 40). Akan tetapi tidak sumua hasil dari pengolahan tersebut bisa menjadi *inform*asi, hasil pengolahan data yang tidak memberikan makna atau arti serta tidak bermanfaat bagi seseorang bukanlah merupakan *inform*asi bagi orang tersebut.

Suatu *inform*asi baru bisa dikatakan sebuah *inform*asi yang berkualitas bila didalamnya mengandung ciri-ciri sebagai berikut:

1. Akurat

   Artinya *inform*asi harus mencerminkan keadaan yang sebenarnya. Pengujian terhadap hal ini biasanya dilakukan melalui pengujian yang dilakukan oleh dua orang atau lebih yang berbeda dan pabila hasil pengujian tersebut menghasilkan hasi yang sama maka dianggap data tersebut akurat.

2. Tepat Waktu

   Artinya i*nforma*si itu harus tersedia atau ada pada saat i*nforma*si itu dibutuhkan, tidak besok atau tidak beberapa jam lagi.

3. Relevan

   Artinya i*nforma*si yang diberikan harus sesuai dengan yang dibutuhkan. Kalau kebutuhan i*nforma*si ini untuk suatu organisasi maka i*nforma*si tersebut harus sesuai dengan kebutuhan i*nforma*si dalam organisasi tersebut.

4. Lengkap

   Artinya i*nforma*si harus diberikan secara lengkap. Misalnya i*nforma*si tentang penjualan tidak ada bulannya atau tidak ada data fakturnya.

I*nforma*si sangatlah berharga karena i*nforma*si dapat menunjukan sumber daya lainnya seperti segala sesuatu yang dapat kita lihat dan kita raba. Kemampuan i*nforma*si dalam menunjukan sumber daya lainnya dan apa yang terjadi dengan sumber daya tersebut sangatlah penting untuk suatu sistem.

Dari sumber di atas dapat disimpulkan bahwa i*nforma*si merupakan hasil dari pengolahan data yang telah memiliki arti sehingga i*nforma*si tersebut dapat digunakan oleh pihak-pihak yang membutuhkan dalam mengambil suatu kebijaksanaan.

Menurut Azhar Susanto (2006 : 40) **:** Data adalah fakta atau apapun yang dapat digunakan sebagai *input* dalam menghasilkan i*nforma*si. Data Bisa berupa bahan untuk diskusi, pengambilan keputusan, perhitungan, atau pengukuran.

2.3 Konsep Dasar Sistem Informasi

I*nforma*si merupakan hal yang sangat penting bagi suatu manajemen dalam mengambil keputusan. I*nforma*si tersebut dapat diperoleh dari suatu sistem infomasi atau disebut juga dengan *processing system* atau *information processing system* atau *information generating system*. Berikut adalah uraian mengenai sistem i*nforma*si

2.3.1 Definisi Sistem Informasi

Sistem i*nform*asi adalah "suatu sistem didalam suatu organisasi yang merupakan kombinasi dari orang-orang, fasilitas, teknologi, media, prosedur-prosedur dan pengendalian yang ditunjukkan untuk mendapatkan jalur komunikasi penting, memproses tipe transaksi rutin tertentu, memberi sinyal kepada manajemen dan lainnya terhadap kejadian-kejadian internal atau external yang penting dan menyediakan suatu dasar i*nform*asi untuk pengambilan keputusan yang cerdik". Jogiyanto(2005: 11).

Menurut Jogiyanto (2005 : 12), faktor-faktor yang menentukan kehandalan dari sebuah sistem i*nform*asi dapat dikatakan baik, apabila memenuhi kriteria-kriteria sebagai berikut;

1. Keunggulan (*usefulness*)

   Keunggulan berarti suatu sistem harus dapat menghasilkan i*nform*asi yang tepat dan relevan untuk mengambil keputusan manajemen dan personil operasi dalam organisasi.

2. Ekonomis

   Kemampuan sistem yang mempengaruhi sistem harus bernilai manfaat minimal sebesar biayanya.

3. Kehandalan *(Realibility)*

   Keluaran dari sistem harus mempunyai tingkat ketelitian tinggi dan sistem tersebut harus beroperasi secara efektif.

4. Pelayanan *(Custumer Service)*

   Suatu sistem memberikan pelayanan yang baik dan efisien kepada para pengguna sistem pada saat berhubungan dengan organisasi.

5. Kapasitas *(Capacity)*

   Setiap sistem harus mempunyai kapasitas yang memadai untuk menangani setiap periode sesuai kebutuhan.

6. Sederhana dalam kemudahan

   Sistem tersebut lebih sederhana (umum) sehingga struktur dan operasinya dapat dengan mudah dimengerti dan prosedur mudah diikuti.

7. Fleksibel *(Fleksibility)*

   Sistem in*form*asi ini harus dapat digunakan dalam kondisi yang bagaimana yang diinginkan oleh organisasi tersebut atau pengguna tertentu.

2.3.2 Komponen Sistem Informasi

Menurut Jogiyanto (2005 : 1), sistem in*form*asi terdiri dari enam komponen yang disebut blok bagunan *(building block)* yang saling berinteraksi antara satu sama lain dengan membentuk satu kesatuan demi mencapai sasarannya. Komponen tersebut antara lain;

1. Blok masukan *(input block)*

   Input mewakili data yang masuk kedalam sistem in*form*asi. *Input* disini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan yang dapat berupa dokumen-dokumen dasar.

2. Blok Model *(model block)*

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

3. Blok Keluaran *(output block)*

   Produk dari sistem in*form*asi adalah keluaran yang merupakan in*form*asi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.

4. Blok Teknologi *(technology block)*

   Teknologi digunakan untuk menerima *input*, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem keseluruhan.

5. Blok Basis Data *(database block)*

   Basis data merupakan kumpulan dari data yang saling berhubungan satu sama lainnya. Organisasi basis data yang baik berguna untuk efisiensi kapasitas penyimpanan. Basis data dapat diakses atau dimanipulasi dengan

menggunakan perangkat lunak paket yang disebut dengan DBMS *(Database Management System)*.

6. Blok Kendali *(block control)*

   Supaya sistem in*form*asi dapat berjalan sesuai dengan yang diinginkan, maka perlu diterapkan pengendalian-pengendalian didalamnya. Beberapa pengendalian perlu dirancang dan diterapkan untuk menyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat langsung diatasi.

2.4 JAVA

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995.

Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM).

Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan *dependensi* implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa *platform* sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

2.4.1 Sejarah Perkembangan JAVA

Bahasa pemrograman Java terlahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot *Duke* yang dibuat oleh Joe Palrang.

Pertemuan proyek berlangsung di sebuah gedung perkantoran *Sand Hill Road* di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program *Java Oak* pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (*Star Seven*).

Setelah era *Star Seven* selesai, sebuah anak perusahaan Tv kabel tertarik ditambah beberapa orang dari proyek The Green Project. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto.

Perusahaan baru ini bertambah maju: jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan perambah (*browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai *Web Runner*, terinsipirasi dari film 1980-an, *Blade Runner*. Pada perkembangan rilis pertama, *Web Runner* berganti nama menjadi *Hot* Java.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 Mei 1995.

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric

Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "Bapak Java", James Gosling. Nama Oak ini tidak dipakai untuk versi *release* Java karena sebuah perangkat lunak lain sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling. Konon kopi ini berasal dari Pulau Jawa. Jadi nama bahasa pemrograman Java tidak lain berasal dari kata Jawa (bahasa Inggris untuk Jawa adalah *Java*).

2.4.2 Kelebihan

Setiap bahasa pemrograman pasti memiliki kelebihan yang diunggulkan. Adapun kelebihan dari bahasa pemrograman JAVA adalah sebagai berikut:

1. ***Multiplatform***. Kelebihan utama dari Java ialah dapat dijalankan di beberapa *platform* / sistem operasi komputer, sesuai dengan prinsip *tulis sekali, jalankan di mana saja*. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / *bytecode*) sekali lalu hasilnya dapat dijalankan di atas beberapa *platform* tanpa peubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas *operating system* Linux tetapi dijalankan dengan baik di atas Microsoft Windows. *Platform* yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mac OS dan Sun Solaris. Penyebanya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs Java) untuk meninterpretasikan *bytecode* tersebut.
2. **OOP** (*Object Oriented Programming* - Pemrogram Berorientasi Objek)
3. **Perpustakaan Kelas Yang Lengkap**, Java terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan

perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

4. **Bergaya C++**, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.
5. **Pengumpulan sampah** otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

2.4.3 Kekurangan

Selain memiliki kelebihan yang diunggulkan sebagai bahasa pemrograman, JAVA juga memiliki beberapa kekurangan, yaitu:

1. **Tulis sekali, jalankan di mana saja** - Masih ada beberapa hal yang tidak kompatibel antara *platform* satu dengan *platform* lain. Untuk J2SE, misalnya *SWT-AWT bridge* yang sampai sekarang tidak berfungsi pada Mac OS X.
2. **Mudah didekompilasi**. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET *Platform*. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/di*reverse-engineer*.
3. **Penggunaan memori yang banyak**. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi

sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena *trend* memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.4.4 Integrated Development Environment

adalah program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak.

Sebuah IDE, atau secara bebas dapat diterjemahkan sebagai Lingkungan Pengembangan Terpadu, setidaknya memiliki fasilitas:

1. *Editor*, yaitu fasilitas untuk menuliskan kode sumber dari perangkat lunak.
2. *Compiler*, yaitu fasilitas untuk mengecek sintaks dari kode sumber kemudian mengubah dalam bentuk binari yang sesuai dengan bahasa mesin.
3. *Linker*, yaitu fasilitas untuk menyatukan data binari yang beberapa kode sumber yang dihasilkan *compiler* sehingga data-data binari tersebut menjadi satu kesatuan dan menjadi suatu program komputer yang siap dieksekusi.
4. *Debuger*, yaitu fasilitas untuk mengetes jalannya program, untuk mencari *bug*/kesalahan yang terdapat dalam program.

Sampai tahap tertentu IDE modern dapat membantu memberikan saran yang mempercepat penulisan. Pada saat penulisan kode, IDE juga dapat menunjukan bagian-bagian yang jelas mengandung kesalahan atau keraguan.

2.5 Metode Pengembangan Sistem

Terdapat beberapa metode dalam melakukan pengembangan sistem, namun dalam hal ini hanya akan membahas satu metode yang mendekati dalam melakukan pengembangan sistem. Adapun metode yang mendekati adalah Metode pengembangan dengan model *prototyping*.

Metode pengembangan dengan model *prototyping* yang merupakan model yang paling sering digunakan pada dunia *rill*. Karena metode ini secara keseluruhan akan mengacu kepada kepuasan pengguna. Bisa dikatakan metode ini merupakan metode *waterfall* yang dilakukan secara berulang-ulang. Keuntungan dari metode ini adalah adanya komunikasi yang baik antar pengembang sistem dan pengguna sistem, pengembang dapat bekerja lebih baik untuk memenuhi kebutuhan pelanggan, pengguna berperan aktif dalam pengembangan sistem, menghemat waktu dalam pengembangan sistem dan penerapan lebih mudah karena pengguna akan mengetahui fungsional yang diharapkan.

2.6 Metode Pendekatan Sistem

Pendekatan sistem dapat diartikan sebagai titik tolak atau sudut pandang kita terhadap proses sebuah sistem yang merujuk pada pandangan tentang terjadinya suatu proses yang bersifat masih sangat umum dalam mewadahi, mengispirasi, dan melatari metode pendekatan dalam cakupan teoritis tertentu.

Didalam melakukan penelitian ini, penulis menggunakan metode pendekatan sistem berorientasi objek, namun ada baiknya apabila kita melakukan perbandingan terlebih dahulu antara pendekatan terstruktur dan pendekatan berorientasi objek.

Menurut Adi Nugroho (2005 : 14) ”pendekatan terstruktur adalah aktifitas menuliskan daftar perintah yang harus diikuti oleh komputer, kemudian mengorganisasi perintah-perintah tersebut kedalam kelompok-kelompok yang dinamakan fungsi/prosedur. Yang umumnya menggunakan diagram aliran *(flow-chart)* untuk mengorganisasi aliran perintah dari satu aksi ke aksi lainnya”.

Sedangkan pendekatan berorientasi objek adalah suatu cara berpikir atau berlogika menghadapi masalah-masalah yang akan dicoba-atasi dengan bantuan

komputer, mencoba melihat permasalahan lewat pengamatan dunia nyata dimana setia objek adalah entitas tunggal yang memiliki kombinasi struktur dan fungsi tertentu.

Adapun perbandingan dari sisi lain kedua metode ini adalah ;

Pendekatan Terstruktur;

1. Data global, pada program yang berukuran sangat besar sangat sulit dilacak. Jika ingin merevisi struktur data global maka kita harus merevisi setiap prosedur/fungsi yang menggunakan data global tersebut. Ini akan sangat memakan waktu dan biaya.
2. Pendekatan terstruktur tidak menggambarkan dunia nyata *(real word)* dengan baik, ini karena fungsi-fungsi berorientasi pada aksi dan tidak berhubungan langsung dengan permasalah.
3. Program yang dibuat dengan pendekatan terstruktur lebih mirip dengan apa yang dibayangkan oleh pengembang dibandingkan dengan harapan dan keinginan pengguna.

Pendekatan Berorientasi Objek

1. Pendekatan berorientasi objek menawarkan pendekatan lainnya yang lebih ekonomis, yaitu dengan membuat objek-objek mandiri, yang sedapat mungkin terlepas satu sama lain dan bekerja sama dengan mengirim pesan *(message)* satu sama lain.
2. Selain menggorganisasi data-data, model objek juga mengorganisasi prosedur-prosedur dan fungsi-fungsi yang berguna untuk memanipulasi data yang terorganisasi.

Program yang dibuat dengan pendekatan berorientasi objek disesuaikan terlebih dahulu dengan kebutuhan pengguna.

2.6.1 Model Pendekatan Berorientasi Objek

Pemodelan objek adalah suatu metoda untuk menggambarkan suatu sistem yang memperlihatkan semua objek yang ada pada sistem. Model objek juga

memperlihatkan hubungan objek terhadap objek-objek lain, serta menampilkan atribut serta operasi yang menjadi ciri suatu kelas tertentu untuk kepentingan pengembangan suatu sistem in*form*asi dan/atau perangkat lunak tertentu.

Pemodelan objek juga merupakan sarana komunikasi antar anggota pengembang dengan calon pengguna untuk menangkap keinginan/kebutuhan pengguna serta harapan calon pengguna *(user's needs and expectation),* dan pengguna lebih mengetahui fungsional sebuah sistem sesuai apa dengan yang diharapkan. Fokus utama pemodelan berorientasi objek adalah dengan melihat objek yang saling berhubungan. Adapun tahapan-tahapan metodelogi berorientasi objek adalah sebagai berikut :

1. Perencanaan

   Tahapan perencanaan *(planning)* adalah menyangkut studi tentang kebutuhan pengguna *(user's specification),* studi kelayakan baik secara teknis maupun secara teknologi serta penjadwalan pengembangan suatu proyek sistem in*form*asi dan perangkat lunak.

2. Analisis

   Tahapan analisi *(analysis),* yaitu tahap dimana kita berusaha mengenali segenap permasalahan yang muncul pada penguna dengan mendekomposisi dan merealisasikan *use case* diagram lebih lanjut, mengenali komponen-komponen sistem/perangkat lunak, objek-objek, hubungan antarobjek dan sebagainya.

3. Perancangan

   Tahapan perancangan *(design)*, yaitu tahapan dimana kita mencoba mencari solusi permasalahan yang didapat dari analisis

4. Implementasi

   Tahapan implementasi, yaitu tahapan dimana kita mengimplementasikan perancangan sistem kesituasi yang nyata, dimana ditahapan ini, kita telah

memulai memilih perangkat keras dan penyusunan perangkat lunak aplikasi *(coding)*.

5. Pengujian

   Tahapan pengujian *(testing)*, yaitu tahapan untuk menentukan apakah sistem/perangkat lunak yang kita buat sudah sesuai dengan kebutuhan pengguna atau belum. Jika belum, proses selanjutnya adalah bersifat iteratif, yaitu kembali ketahap-tahap sebelumnya.

6. Pemeliharaan

   Tahapan pemeliharaan, yaitu tahapan dimana kita mulai melakukan pengoperasian sistem, dan jika diperlukan melakukan perbaikan-perbaikan kecil, kemudian jika waktu penggunaan sistem habis, maka akan masuk lagi pada tahapan perancangan.

2.6.2 *Tools* Yang Digunakan Dalam Pendekatan Berorientasi Objek

Didalam melakukan pendekatan berorientasi objek memiliki *tools* percangan yang dikenal dengan UML *(Unified Modeling Languange)*. Menurut Martin Fowler (2004 : 1) UML adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendiskripsian dan desain perangkat lunak, khususnya sistem yang dibangun dengan menggunakan pemorgraman berorientasi objek.

UML diperkenalkan oleh Ivar Jacobson (yang sebelumnya terkenal dengan konsep OOSE- *Object Oriented Software Engineering*), James Rumbaugh (yang sebelumnya terkenal dengan konsep OMT – *Object Modeling Technique*) serta Grady Booch (yang sebelumnya terkenal dengan konsep notasi *Booch* yang populer digunakan sebagi salah satu metodelogi analisis dan perancangan berorientasi objek).

Sepanjang tahun 1996, Rumbaugh, Booch, dan Jacobson memaparkan konsorsium UML yang dinamakan OMG (*Object Management Group*). OMG merupakan konsorsium yang berusaha membuat standarisasi metoda pengembangan perangkat lunak berorientasi objek.

Sebagai salah satu bahas pemodelan sistem/ perangkat lunak, menurut Adi Nugroho (2005 : 22), ada 3 ciri khas yang terdapat didalam UML, antara lain;

1. Merupakan *Use-Case Driven Language*, yang maksudnya adalah sasaran dari pemodelan adalah memandu para pengembang dalam menciptakan perangkat lunak dan/atau sistem i*nform*asi yang memenuhi segala harapan serta kebutuhan calon pengguna secara efisien.
2. Merupakan *Architecture-Centric Language*, yang maksudnya adalah para pengembang melakukan pengendalian kegiatan pengembangan sistem i*nform*asi/perangkat lunak dengan melakukan organisasi dari sistem/perangkat lunak, elemen-elemen serta *interface* yang tercakup didalam sistem/perangkat lunak yang termasuk didalamnya adalah bagaimana elemen-elemen berperilaku serta berkolaborasi sesuai dengan yang diharapkan oleh calon pengguna.
3. Merupakan *Interative and Incremental Language* yang dimaksud adalah pengembang perangkat lunak/sistem i*nform*asi tidaklah bersifat linear. Pengembang selalu melakukan pengecekan apakah sistem/perangkat lunak yang dibangun sudah sesuai dengan *use case* serta arsitektur yang telah disepakati. Jika belum, maka pengembang bisa kembali ke langkah sebelumnya dengan melakukan perbaikan-perbaikan yang diperlukan sehingga pada langkah selanjutnya, pengembang bisa mendapat pemahaman secara lebih baik.

Adapun konsep dasar dalam dalam UML untuk mendiskripsikan OOP (*Object Oriented Programin*) dengan menggunakan beberap diagram, antara lain ;

a. Use Case Diagram

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan dalam pembuatan *Use Case diagram* adalah "apa" yang diperbuat sistem dan bukan "bagaimana". *Use case* terdiri dari sekumpulan *Scenario* yang dilakukan oleh aktor. *Use case* menggambarkan sebuah pekerjaan tertentu, misalnya sistem meng- *created* sebuah daftar transaksi,

dan sebagainya. Aktor adalah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use Case diagram memfasilitasi komunikasi antar pengembang sistem/perangkat lunak dengan calon pengguna. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di- *include* akan dipanggil setiap kali *use case* yang meng- *include* dieksekusi secara normal. Sebuah *use case* dapat di- *include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common* . Sebuah *use case* juga dapat meng- *extend use case* lain dengan *behaviour* -nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

b. Activity Diagram

Activity diagram menggambarkan aliran aktivitas dalam sistem yang dirancang, yang setiap aliran berawal dari sebuah *decision* yang mungkin terjadi dan bagaimana mereka berakhir. *Activity diagram* merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya. *Activity diagram* tidak

menggambarkan *behavior internal* sebuah sistem dan interaksi antar subsistem secara fisik, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari *level* atas secara umum. Sebuah aktivitas dapat direalisasikan dari sebuah *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem dalam melakukan aktivitas. Diagram *activity* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab dalam melakukan aktivitas.

c. Sequence Diagram

Sequence Diagram menggambarkan perilaku pada sebuah *Scenario*. Diagram ini menunjukkan sejumlah contoh objek dan *message* (pesan) yang

diletakkan diantara objek-objek didalam *use case*. *Sequence Diagram* terdiri antar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Perilaku yang digambarkan adalah rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan sebuah *output*. Diawali dari apa yang meng- *trigger* aktivitas tersebut, proses dan peubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor memiliki *lifeline* vertikal. *Message* yang digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

d. Collaboration Diagram

Collaboration diagram adalah bentuk lain dari *sequence diagram* Bila *Sequence Diagram* diorganisir menurut waktu, maka *collaboration diagram* diorganisir menurut ruang/*space*. *Collaboration diagram* merupakan asosiasi diantara objek panah didekat garis asosiasi menunjukkan *message*, sedangkan *content message* ditujukan dengan label. Dengan *collaboration diagram* memungkinkan untuk memodelkan pengiriman sebuah *message* kebanyak objek pada *class* yang sama, demikian juga halnya dengan menunjukkan adanya objek aktif yang mengendalikan aliran *message*.

e. Class diagram

Class diagram menggambarkan himpunan kelas-kelas, *interface-interface*, kolaborasi-kolaborasi serta relasi-relasi. *Class digram* juga merupakan sebuah spesifikasi yang jika akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class diagram* memiliki tiga area pokok, yaitu :

1. Nama
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

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

Class dapat merupaka implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diintansikan, tetapi harus diimplementasikan dulu dari sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Didalam *class diagram*, terdapat hubungan antar *class*, antara lain;

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksitensi *class* lain.
2. Agregasi, yaitu hubungan yang menyatakan bagian (”terdiri dari”).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis juga dapat digambarkan dengan menggunakan *sequence diagram*.

f. Component Diagram

Component diagram menggambarkan organisasi serta kebergantungan pada komponen-komponen yang telah ada sebelumnya. *Component diagram* ini berhubungan dengan *class diagram* dimana komponen secara tipikal dipetakan kedalam satu atau lebih *class-class*, *interface-interface* serta kolaborasi-kolaborasi.

g. *Deployment Diagram*

Deployment diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infastruktur sistem, dimana komponen akan terletak (pada mesin, server atau perangkat keras lainnya), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server dan hal-hal lainnya yang bersifat fisikal sebuah *node* adalah server *workstation*, atau perangkat keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya.

2.7 Peralatan yang digunakan

Untuk membangun sistem ini penulis menggunakan beberapa peralatan penunnjang. Peralatan itu adalah sebagai berikut.

2.7.1 NetBeans

NetBeans merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra (dan terus bertambah!). Sun Microsystems mendirikan proyek kode terbuka NetBeans pada bulan Juni 2000 dan terus menjadi sponsor utama.

Saat ini terdapat dua produk : NetBeans IDE dan NetBeans *Platform*.

The NetBeans IDE adalah sebuah lingkungan pengembangan - sebuah kakas untuk pemrogram menulis, mengompilasi, mencari kesalahan dan menyebarkan program. Netbeans IDE ditulis dalam Java - namun dapat mendukung bahasa pemrograman lain. Terdapat banyak modul untuk memperluas Netbeans IDE. Netbeans IDE adalah sebuah produk bebas dengan tanpa batasan bagaimana digunakan.

Tersedia juga NetBeans *Platform*; sebuah fondasi yang modular dan dapat diperluas yang dapat digunakan sebagai perangkat lunak dasar untuk membuat aplikasi *desktop* yang besar. Mitra ISV menyediakan *plug-in* bernilai tambah yang dapat dengan mudah diintegrasikan ke dalam *Platform* dan dapat juga digunakan untuk membuat kakas dan solusi sendiri.

Kedua produk adalah kode terbuka (*open source*) dan bebas (*free*) untuk penggunaan komersial dan *non* komersial. Kode sumber tersedia untuk guna ulang dengan lisensi *Common Development* and *Distribution License* (CDDL).

2.7.2 MySQL

Menurut Bunafit Nugroho (2004 : 1) MySQL adalah sebuah program *database* server yang mampu menerima dan mengirimkan datanya dengan sangat cepat, multi *user* serta menggunakan perintah standar SQL

1. Keunggulan MySQL

Menurut Bunafit Nugroho (2004 : 3) ada beberapa keunggulan MySQL yang antara lain adalah ;

a. MySQL sebagai *Database Management System* (DBMS).

b. MySQL sebagai *Relation Database Management System* (RDBMS).

c. MySQL bersifat *OpenSource*, artinya program ini bersifat *free*.

d. MySQL merupakan sebuah *database* server, jadi dengan menggunakan *databse* ini kita dapat menghubungkan ke media internet sehingga dapat diakses dari jauh.

e. MySQL merupakan sebuah *database client*. Selain menjadi server yang melayani permintaan, MySQL juga dapat melakukan *query* yang mengakses *database* pada server.

f. MySQL mampu menerima *query* yang bertumpuk dalam satu permintaan atau disebuh *multi-threading*.

g. MySQL merupakan sebuah *database* yang mampu menyimpan data berkapasitas sangat besar hingga berukuran *gigabyte* sekalipun.

h. MySQL didukung oleh *driver* ODBC, artinya *database* MySQL dapat diakses menggunakan aplikasi apa saja termasuk berupa visual seperti delphi maupun visual basic.

i. MySQL adalah *database* menggunakan enskripsi *password*. Jadi *database* ini cukup aman karena memiliki *password* untuk mengaksesnya.

j. MySQL merupakan server *database* yang multi *user*, artinya *database* ini tidak hanya digunakan oleh sepihak orang akan tetapi merupakana *database* ruang dapat digunakan oleh banyak pengguna.

k. MySQL dapat menciptakan lebih dari 16 kunci per tabel, dan dalam satu kunci memungkinkan berisi belasan *field*.

l. MySQL mendukung *field* yang dijadikan sebagai kunci primer dan kunci uniq.

m. MySQL memiliki kecepatan dalam pembuatan tabel maupun peng-*update*-an tabel.