BAB III

LANDASAN TEORI

Landasan teori adalah seperangkat konsep, definisi, dan proposisi yang disusun guna menyelesaikan masalah secara sistematis. Pada bab ini akan membahas landasan teori yang membahas tentang ilmu dan landasan pemikiran yang terkait dan mendukung dalam proyek sistem informasi ini.

3.1 Sistem

Menurut Jogiyanto (1989 : 23), sistem merupakan kumpulan dari elemen-elemen yang satu dengan yang lain berinteraksi dan bersama-sama beroperasi untuk mencapai tujuan tertentu. Sistem mempunyai peran yang sangat besar dalam menentukan berjalan tidaknya suatu lembaga atau perusahaan. Hal ini dikarenakan setiap perusahaan akan selalu berdasarkan pada suatu sistem dalam menjalankan aktifitas sehari-harinya.

Suatu sistem dapat dirumuskan sebagai suatu totalitas himpunan yang terdiri dari bagian-bagian yang mana antara satu dengan yang lainnya saling berinteraksi dan bersama-sama beroperasi guna mencapai suatu tujuan tertentu didalam suatu lingkungan. Bagian-bagian atau subsistem tersebut merupakan suatu kompleksitas tersendiri, tapi dalam kebersamaan mencapai suatu tujuan berlangsung secara harmonis dalam keteraturan yang pasti.

3.2 Sistem Informasi

Menurut Robert A. Leitch dan K. Roscoe Davis (1983:86), sistem informasi dapat didefinisikan sebagai suatu sistem di dalam suatu organisasi yang

mempertemukan kebutuhan pengolah transaksi atau informasi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.

Menurut Leman (1998:3-4) komponen sistem informasi terdiri dari *hardware, software,* data, manusia, dan prosedur. Kegiatan dalam sistem informasi mencakup :

1. Input, menggambarkan suatu kegiatan untuk menyediakan data untuk diproses.
2. Proses, menggambarkan bagaimana suatu data diproses untuk menghasilkan suatu informasi yang bernilai tambah.
3. Output, suatu kegiatan untuk menghasilkan laporan dari proses diatas tersebut.
4. Penyimpanan, suatu kegiatan untuk memelihara dan menyimpan data.
5. Control, suatu aktivitas untuk menjamin bahwa sistem informasi tersebut berjalan sesuai dengan yang diharapkan.

3.3 Analisa dan Perancangan Sistem

Analisa sistem merupakan tahap yang paling penting dari suatu pemrograman, karena merupakan tahap awal untuk melakukan evaluasi permasalahan yang terjadi serta kendala-kendala yang dihadapi dari sebuah sistem yang telah berjalan.

Analisa yang efektif akan memudahkan pekerjaan penyusunan rencana yang baik di tahap berikutnya. Sebaliknya, kesalahan yang terjadi pada tahap analisa ini akan menyebabkan kesulitan yang lebih besar, bahkan dapat menyebabkan gagalnya penyusunan sebuah sistem (Kendall,2003).

Untuk itu, diperlukan ketelitian dalam mengerjakan, sehingga tidak terdapat kesalahan dalam tahap selanjutnya, yaitu tahap perancangan sistem. Langkah-langkah yang diperlukan di dalam menganalisa sistem adalah:

1. Tahap perencanaan sistem
2. Tahap analisis sistem
3. Tahap perancangan sistem
4. Tahap penerapan sistem
5. Membuat laporan dari hasil analisa

Pada tahap perencanaan, dilakukan identifikasi masalah serta diperlukan adanya analisa yang digunakan untuk menentukan faktor-faktor yang menjadi permasalahan dalam sistem yang telah ada atau digunakan.

Data-data yang baik yang berasal dari sumber-sumber internal seperti misalnya laporan-laporan, dokumen, observasi, maupun dari sumber-sumber di luar lingkungan sistem seperti pemakai sistem, dikumpulkan sebagai bahan pertimbangan analisa. Jika semua permasalahan telah di identifikasi, dilanjutkan dengan mempelajari dan memahami alur kerja dari sistem yang digunakan.

Kemudian diteruskan dengan menganalisa dan membandingkan sistem yang terbentuk dengan sistem sebelumnya. Dengan adanya perubahan tersebut, maka langkah selanjutnya adalah membuat laporan-laporan hasil analisa sebelumnya dan sistem yang akan diterapkan. Perancangan sistem adalah proses menyusun atau mengembangkan sistem informasi yang baru. Dalam tahap ini, harus dipastikan bahwa semua persyaratan untuk menghasilkan informasi dapat terpenuhi.

Hasil sistem yang dirancang harus sesuai dengan kebutuhan pemakai, karena rancangan tersebut meliputi perancangan mulai dari sistem yang umum hingga diperoleh sistem yang lebih spesifik. Dari hasil rancangan sistem tersebut, dibentuk pula rancangan *database* disertai dengan struktur file antara sistem yang satu dengan yang lain. Selain itu, dibentuk pula rancangan input dan output sistem, misalnya menentukan berbagai bentuk input data dan isi laporan.

Apabila di dalam perancangan sistem terdapat kesalahan, maka kita perlu melihat kembali analisa dari sistem yang telah dibuat. Sehingga dapat di ambil kesimpulan bahwa analisa sistem mempunyai hubungan erat dengan perancangan sebuah sistem.

3.4 Visual Basic. Net 2005

Visual Basic. Net 2005 merupakan bahasa pemrograman yang bersifat *event driven* dan menawarkan Integrated Development Environment (IDE) visual untuk membuat program aplikasi berbasis sistem operasi Microsoft Windows dengan menggunakan model pemrograman Common Object Model (COM). Visual Basic merupakan turunan bahasa BASIC dan menawarkan pengembangan aplikasi komputer berbasis grafik dengan cepat, akses ke basis data menggunakan Data Access Objects (DAO), Remote Data Objects (RDO), atau ActiveX Data Objects (ADO), serta menawarkan pembuatan konsol ActiveX dan objek ActiveX. Beberapa bahasa skrip seperti Visual Basic for Applications (VBA) dan Visual Basic Scripting Edition (VBScript), mirip seperti halnya Visual Basic, tetapi cara kerjanya yang berbeda. Visual Basic.Net (VB.Net) merupakan pengembangan dari bahasa pemrograman Visual Basic sebelumnya yaitu Visual Basic 6. Beberapa keunggulan Visual Basic.Net dengan Visual Basic sebelumnya, yaitu:

1. Menyederhanakan *Deployment*

   Visual Basic.Net mengatasi masalah seputar *deployment* dari aplikasi berbasis Windows, yaitu "DLL HELL" dan registrasi COM (Component Object Model), sehingga dapat mempermudah *deployment* aplikasi yang berbasis Windows.

2. Menyederhanakan Pengembangan Perangkat Lunak

   Visual Basic.Net memiliki fitur *compiler* yang bekerja secara *real time* dan daftar *task* untuk penanganan kesalahan atau *bug* program sehingga pengembang dapat menangani secara langsung kesalahan program yang terjadi.

3. Mendukung Object Oriented Programming (OOP)

   Dalam Visual Basic.Net, dapat dibuat kode dalam *class* yang menggunakan secara penuh konstruksi berbasis objek. Class tersebut memiliki sifat *re-usable* atau dapat digunakan kembali. Visual Basic.Net memiliki fitur bahasa pemrograman berbasis objek termasuk implementasinya secara penuh, diantaranya sebagai contoh adalah konsep *inheritance* atau pewarisan, *encapsulation* atau pembungkusan, dan *polymorphism* atau banyak bentuk.

4. Mempermudah Migrasi dari Visual Basic 6 ke Visual Basic.Net 2005

   Interopability Common Object Model menyediakan komunikasi dua arah antara aplikasi Visual Basic 6 dengan Visual Basic.Net 2005. Wizard upgrade pada Visual Basic.Net 2005 memungkinkan pengembang dapat melakukan migrasi lebih dari 95% kode Visual Basic 6 menjadi kode Visual Basic.Net 2005.

Budiharto (2006:1) menyebutkan, "Visual Basic.Net 2005 adalah bahasa pemrograman terbaru yang memudahkan programmer Visual Basic 6 beralih ke

Visual Basic.Net 2005". Budiharto (2006:3-4) juga menyebutkan alasan penting lainnya untuk melakukan migrasi ke Visual Basic.Net 2005, yaitu:

1. Visual Basic.Net 2005 mengatasi semua masalah yang sulit di sekitar pengembangan aplikasi berbasis Windows dan mengurangi penggunaan aplikasi lainnya serta versi komponen, bahkan mewarisi sifat C++ dan berbau Java.
2. Windows form designer memungkinkan *developer* memperoleh aplikasi *desktop* dalam waktu yang singkat.
3. Bagi *developer*, Visual Basic.Net 2005 menyediakan model pemrograman data akses ActiveX Data Object (ADO) yang sudah dikenal dan diminati, ditambah dengan XML (Extensible Markup Language) baru yang berbasis Microsoft ADO.Net. Dengan ADO.Net, *developer* akan memperoleh komponen yang lebih baik, seperti control DataSet.
4. Visual Basic.Net 2005 menghasilkan Visual Basic.Net 2005 untuk web. Menggunakan form web yang baru memudahkan untuk membangun *thin-client* aplikasi berbasis web yang secara cerdas berjalan di *browser* dan *platform* manapun.
5. Mendukung pembangunan aplikasi *client-server*, terdistribusi, serta berupa aplikasi yang berbasis Windows serta web.
6. .Net Framework secara mendasar dibuat untuk dipasangkan pada Windows 2003 dengan keunggulan untuk memonitor kelalaian dari aplikasi yang sedang berjalan dan mengisolasi setiap aplikasi.

7. *Developer* dengan berbagai latar belakang bahasa pemrograman dapat dengan segera menguasai Visual Basic.Net 2005 karena kemudahan dan kemiripan kode yang ditawarkannya.
8. Integrasi dengan sistem yang telah ada sangat mudah, .Net Framework COM memungkinkan untuk dapat berinteraksi dan dengan dengan sistem yang sudah ada menggunakan XML Web Service. Visual Studio Upgrade Tool yang tersedia pada Visual Basic.Net 2005 dan Java Language Convention Assistant membantu menkonversi Visual Basic 6 dan Visual J++ agar berjalan pada .Net Framework.
9. Mendukung lebih dari 20 bahasa pemrograman, .Net Framework mendukung integrasi lebih dari 20 bahasa pemrograman yang tidak terbayang sebelumnya. Memungkinkan pengembang memilih bahasa pemrograman yang tepat sesuai latar belakang pemrogramnya.

3.5 Microsoft SQL Server 2005 Express

Microsoft SQL Server merupakan produk RDBMS (Relational Database Management System) yang dibuat oleh Microsoft. Microsoft SQL Server juga mendukung SQL (Structured Query Language) sebagai bahasa untuk memproses baris perintah ke dalam basis data. SQL ini telah digunakan secara umum pada semua produk *database* server yang ada di pasaran saat ini. Microsoft SQL Server banyak digunakan pada dunia bisnis, pendidikan, dan juga pemerintahan sebagai solusi *database* atau media penyimpanan data. Berbagai macam skala bisnis, dari bentuk bisnis kecil sampai bisnis skala *enterprise* dapat menggunakan Microsoft SQL Server sebagai pusat basis datanya. Microsoft SQL Server merupakan sebuah *database* relational yang dirancang untuk mendukung aplikasi dengan

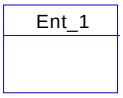
arsitektur *client-server*, dimana *database* terdapat pada komputer pusat yang disebut dengan *server*, dan informasi digunakan bersama-sama oleh beberapa *user* yang menjalankan aplikasi di dalam komputer lokalnya yang disebut dengan *client*. Arsitektur semacam ini memberikan integritas data yang cukup tinggi, karena semua *user* bekerja dengan informasi yang sama. Arsitektur *client-server* dapat mengurangi lalu lintas jaringan karena prosesnya hanya berjalan dengan permintaan data yang diperlukan oleh *user*.

Microsoft SQL Server 2005 Express dibagi kedalam beberapa komponen logis, seperti misalnya *table*, *view*, dan elemen-elemen lain yang dapat dilihat oleh *user* dengan menambahkan *add-on* dari aplikasi dengan nama *database management system*. Elemen-elemen ini secara fisik disimpan di dalam dua atau lebih *file* di dalam *disk*. Format *file* atau lokasi dimana elemen *logic* ini ditulis, tidak diketahui oleh *user system*. Apabila suatu *database* telah dibuat, *user* bisa memiliki akses yang telah diberikan kepadanya. Hal ini membuat Microsoft SQL Server 2005 Express dapat menyimpan beberapa *database* dan membatasi akses ke masing-masing *database* kepada *user* tertentu.

3.6 Entity Relationship Diagram

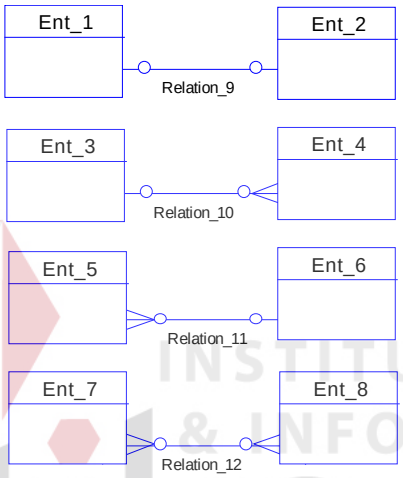
Entity Relationship Diagram, atau yang lebih dikenal dengan nama ERD, digunakan untuk mengimplementasikan, menentukan, dan mendokumentasikan kebutuhan-kebutuhan untuk sistem pemrosesan *database*. ERD menyediakan bentuk untuk menunjukkan struktur keseluruhan kebutuhan data dari pemakai. Adapun elemen-elemen yang terdapat pada ERD, adalah sebagai berikut:

1. *Entity* atau entitas, digambarkan dalam bentuk persegi seperti pada gambar berikut:



Gambar 3.1 Entity atau Entitas

2. *Relation* atau relasi merupakan penghubung antara entitas dengan entitas. Terdapat beberapa jenis relasi yang dapat digunakan, seperti *one-to-one*, *one-to-many*, *many-to-one*, dan *many-to-many*. Bentuk alur relasi secara detil dapat dilihat pada gambar berikut:



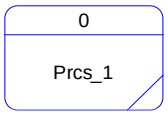
Gambar 3.2 Relation of Entity

3.7 Data Flow Diagram

Menurut Andri Kristanto (2004), Data Flow Diagram (DFD) adalah suatu model logika data atau proses yang dibuat untuk menggambarkan darimana asal data dan kemana tujuan data yang keluar dari sisem, dimana data tersebut disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan, dan proses yang dikenakan pada data tersebut.

Data Flow Diagram merupakan suatu metode pengembangan sistem yang terstruktur (*structure analysis* and *design*). Penggunaan notasi dalam *data flow diagram* sangat membantu untuk memahami suatu sistem pada semua tingkat

kompleksitas. Pada tahap analisi, penggunaan notasi ini dapat membantu dalam berkomunikasi dengan pemakai sistem untuk memahami sistem secara logika. Di dalam *data flow diagram*, terdapat empat simbol yang digunakan yaitu *process*, *external entity*, *data store*, dan *data flow*. Simbol *process* digunakan untuk melakukan suatu perubahan berdasarkan data yang diinputkan dan menghasilkan data dari perubahan tersebut. Simbol *process* dapat digambarkan sebagai bentuk berikut:



Gambar 3.3 Process

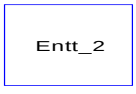
Pada bentuk gambar *process*, bagian atas berisi nomor untuk identitas proses. Suatu proses dengan nomor 0 (nol atau kosong) menandakan bahwa proses tersebut adalah sebuah context diagram. Diagram ini merupakan level tertinggi dari DFD yang menggambarkan hubungan sistem dengan lingkungan luarnya. Pembuatan context diagram dapat dilakukan dengan terlebih dahulu menentukan nama sistemnya, menentukan batasan dari sistem, dan menentukan terminator yang diterima atau diberikan daripada sistem untuk kemudian dilakukan penggambaran.

Nomor 1, 2, 3, dan seterusnya menandakan bahwa proses tersebut diartikan sebagai proses level-0 (nol) yang merupakan hasil turunan atau decompose dari proses context diagram. Proses level-0 membahas sistem secara lebih mendetil, baik dipandang dari segi kegiatan dari sebuah bagian, alur data yang ada, maupun *database* yang digunakan didalamnya. Pembuatannya dapat dilakukan dengan cara menentukan proses utama yang ada dalam sistem, menentukan alur data yang

diterima dan diberikan masing-masing proses daripada sistem sambil memperhatikan konsep keseimbangan (alur data yang masuk atau keluar dari suatu level harus sama dengan alur data yang masuk dan keluar pada level berikutnya), memunculkan data store sebagai sumber maupun tujuan data (optional), menggambarkan diagram level-0, menghindari perpotongan arus data, dan melakukan pemberian nomor pada proses utama (nomor tidak menunjukkan urutan proses).

Nomor 1.1, 1.2, 2.1, 2.2, dan seterusnya merupakan sebuah proses turunan atau *decompose* dari proses level-0 yang disebut sebagai proses level-1 (satu). Proses level-1 menggambarkan detil kerja dari sebuah bagian dalam sebuah sistem. Penggambarannya dilakukan dengan cara menentukan proses yang lebih kecil (sub-proses) dari proses utama yang ada di level-0, menentukan apa yang diterima atau diberikan masing-masing sub-proses daripada sistem dan tetap memperhatikan konsep keseimbangan, memunculkan *data store* sebagai sumber maupun tujuan alur data (*optional*), menggambar DFD level-1, dan berusaha untuk menghindari perpotongan arus data. Hasil turunan akhir disebut sebagai the *lowest level*, dimana hasil akhir ini tergantung dari kompleksitas sistem yang ada.

External entity disimbolkan dengan bentuk persegi yang digunakan untuk menggambarkan pelaku-pelaku sistem yang terkait, dapat berupa orang-orang, organisasi maupun instansi. *External entity* dapat memberikan masukan kepada *process* dan mendapatkan keluaran dari *process*. *External entity* digambarkan dalam bentuk sebagai berikut:

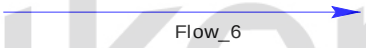


Gambar 3.4 External Entity

Data *store* digunakan sebagai media penyimpanan suatu data yang dapat berupa *file* atau *database*, arsip atau catatan manual, lemari *file*, dan tabel-tabel dalam *database*. Penamaan data *store* harus sesuai dengan betuk data yang tersimpan pada data *store* tersebut, misalnya tabel pelanggan, tabel detil penjualan, tabel detil pembelian, dan lain-lain. Data store digambarkan dalam bentuk simbol sebagai berikut:

Gambar 3.5 Data Store

Data *flow* merupakan penghubung antara *external entity* dengan *process* dan *process* dengan data *store*. Data *flow* menunjukkan aliran data dari satu titik ke titik lainnya dengan tanda anak panah mengarah ke tujuan data. Penamaan data *flow* harus menggunakan kata benda, karena di dalam data *flow* mengandung sekumpulan data. Data *flow* digambarkan dengan bentuk simbol sebagai berikut:



Gambar 3.6 Data Flow

3.8 System Flow

System flow adalah bagan yang menunjukkan arus pekerjaan secara menyeluruh dari suatu sistem dimana bagan ini menjelaskan urutan prosedur-prosedur yang ada dalam sistem dan biasanya dalam membuat *system flow* sebaiknya ditentukan pula fungsi-fungsi yang melaksanakan atau bertanggung jawab terhadap sub-sistem yang ada (Jogiyanto, 1998).

Terdapat berbagai macam bentuk simbol yang digunakan untuk merancang sebuah desain dari sistem, diantaranya adalah *terminator*, *manual operation*,

document, *process*, *database*, *manual input*, *decision*, *off-line storage*, *on-page reference*, dan *off-page reference*.

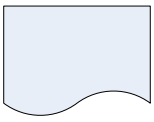
Terminator merupakan bentuk simbol yang digunakan sebagai tanda dimulainya jalan proses sistem ataupun tanda akhir dari sebuah pengerjaan suatu sistem. Bentuk dari *terminator* adalah sebagai berikut:

Gambar 3.7 Terminator

Manual operation digunakan untuk menggambarkan sebuah proses kerja yang dilakukan tanpa menggunakan komputer sebagai medianya (menggunakan proses manual). Bentuk simbolnya adalah:

Gambar 3.8 Manual Operation

Document merupakan simbol dari dokumen yang berupa kertas laporan, surat-surat, memo, maupun arsip-arsip secara fisik. Bentuk dari *document* di gambarkan dalam simbol berikut:

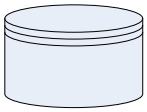


Gambar 3.9 Document

Process adalah sebuah bentuk kerja sistem yang dilakukan secara terkomputerisasi. *Process* disimbolkan dengan gambar:

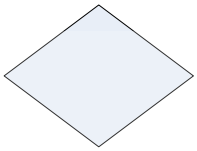
Gambar 3.10 Process

Database digunakan sebagai media penyimpanan data yang bersifat terkomputerisasi. Simbol dari *database* adalah sebagai berikut:



Gambar 3.11 Database

Decision merupakan operator logika yang digunakan sebagai penentu keputusan dari suatu permintaan atau proses dengan dua nilai, benar dan salah. Operator logika ini digambarkan sebagai berikut:

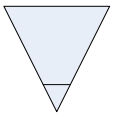


Gambar 3.12 Decision

Manual input digunakan untuk melakukan proses input ke dalam *database* melalui *keyboard*. *Manual input* digambarkan dengan simbol:

Gambar 3.13 Manual Input

Off-line storage merupakan bentuk media penyimpanan yang berbeda dengan *database*, dimana media penyimpanan ini menyimpan dokumen secara manual atau lebih dikenal dengan nama arsip. *Off-line storage* digambarkan dengan simbol:

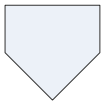


Gambar 3.14 Off-Line Storage

On-page reference digunakan sebagai simbol untuk menghubungkan bagan desain sebuah sistem apabila hubungan arus data yang ada terlalu jauh dalam permasalah letaknya. Bentuk simbol *On-page reference* adalah sebagai berikut:

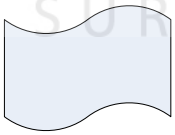
Gambar 3.15 On-Page Reference

Off-page reference memiliki sifat yang sedikit berbeda dengan *On-page reference*, karena simbol ini hanya digunakan apabila arus data yang ada dilanjutkan ke halaman yang berbeda. Bentuk simbolnya adalah:



Gambar 3.16 Off-Page Reference

Paper tape merupakan sebuah simbol yang umumnya menggantikan bentuk penggambaran jenis pembayaran yang digunakan (misal: uang) dalam transaksi yang ada pada sistem yang dirancang. Bentuk dari *paper tape* adalah dengan simbol:



Gambar 3.17 Paper Tape