BAB II

LANDASAN TEORI

II.1 Pengertian Perancangan Aplikasi

Perancangan aplikasi adalah penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi.(Burch, John & Gary Grudnitski, page 461)

Perancangan aplikasi menentukan bagaimana suatu aplikasi akan menyelesaikan apa yang mesti diselesaikan. Tahap ini termasuk mengkonfigurasi dari komponen-komponen perangkat lunak dan perangkat keras dari suatu aplikasi sehingga setelah dilakukan instalasi akan benar- benar memuaskan rancang bangun yang telah ditetapkan pada akhir tahap analisis aplikasi.(M Scott, George, 1986:518)

Dengan demikian dapat diartikan bahwa suatu perancangan aplikasi adalah:

1. Pendefinisian dari kebutuhan-kebutuhan fungsional.
2. Persiapan untuk rancang bangun sebuah implementasi.
3. Menggambarkan bagaimana suatu aplikasi dibentuk.
4. Dapat berupa penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi.
5. Di dalamnya termasuk menyangkut konfigurasi dari komponen-komponen perangkat lunak dan perangkat keras dari suatu aplikasi.

II.1.1. Tujuan Perancangan Aplikasi

Tujuan utama perancangan aplikasi ada dua, yaitu :

1. Untuk memenuhi kebutuhan para pemakai aplikasi
2. Untuk memberikan gambaran dan rancang bangun yang jelas dan lengkap kepada pemrogram komputer dan ahli-ahli teknik lainnya yang terlibat.(Whitten, Jeffry L, Lonnie D Benley, Thomas I.M.Ho, 1986 : 373)

Untuk mencapai kedua tujuan di atas, seorang analis aplikasi harus dapat memenuhi sasaran-sasaran berikut ini :

1. Perancangan aplikasi harus berguna, mudah dipahami dan nantinya mudah digunakan. Ini berarti bahwa data harus mudah ditangkap, metode-metode harus mudah diterapkan dan informasi juga harus mudah dihasilkan serta mudah dipahami dan digunakan.
2. Perancangan aplikasi harus dapat mendukung tujuan utama perusahaan sesuai dengan yang didefinisikan pada tahap perencanaan aplikasi yang dilanjutkan pada tahap analisis aplikasi.(Whitten, Jerry L, 1986, page 373-374).

II.2. Konsep Dasar Program

Sebelum lebih dalam mengupas tugas akhir ini, alangkah baiknya apabila diketahui terlebih dahulu definisi serta uraian singkat mengenai konsep dasar dari program yang berhubungan dengan tugas akhir.

II.2.1. Program

Menurut Sutarman (2009:3) dalam bukunya pengantar teknologi informasi mengemukakan bahwa “program adalah barisan perintah atau instruksi yang

disusun sehingga dapat dipahami oleh komputer dan kemudian dijalankan sebagai barisan perhitungan *numerik*, dimana barisan perintah tersebut berhingga, berakhir, dan menghasilkan *output*".

Beberapa langkah yang harus diperhatikan oleh pemrogram sebelum mengimplementasikan suatu permasalahan dalam program menurut Kristanto (2009:4) adalah:

1. Mendefinisikan masalah
2. Mencari solusi untuk masalah
3. Memilih teknik pemecahan masalah dan algoritma
4. Menulis program
5. Melakukan *testing* dan *debuging*
6. Melakukan dokumentasi
7. Melakukan pemeliharaan

II.2.2. Bahasa Pemrograman

Bahasa pemrogramman menurut Sutarman (2009:159) adalah "*software* yang dipakai oleh para *programer* (pembuat program atau *software*) untuk membuat atau menuliskan perintah-perintah atau program tertentu".

Beberapa contoh *software* bahasa pemrograman antara lain:

1. Borland Delphi
2. Visual Basic
3. C++
4. Assembler
5. PHP , ASP, Phyton, Perl, J2ME (Bahasa pemrograman berbasis WEB)

6. Melakukan dokumentasi

7. Melakukan pemeliharaan

II.3. *Unified Modeling Language* (*UML)*

Adi Nugroho (2010:6), *Unified Modeling Language* (*UML)* adalah bahasa pemodelan untuk aplikasi atau perangkat lunak yang berparadigma "berorientasi objek". Pemodelan *(Modeling)* sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajaran dan dipahami. Dalam hal ini sasaran model sesungguhnya adalah abstraksi segala sesuatu yang ada diplanet bumi menjadi gambaran-gambaran umum yang lebih mudah dipahami dan dipelajari. Adapun tujuan pemodelan (dalam rangka pengembangan sistem/perangkat lunak aplikasi) sebagai sarana analisis, pemahaman visualisasi dan komunikasi antar anggota tim pengembang.

II.3.1. Pengenalan UML

Adi Nugroho (2010), *UML* sebagai sebuah bahasa yang memberikan *vocabulary* dan tatanan penulisan kata-kata dalam *'MS Word*' untuk kegunaan komunikasi. Sebuah bahasa model adalah sebuah bahasa yang mempunyai *vocabulary* dan konsep tatanan/aturan penulisan serta secara fisik mempresentasikan dari sebuah aplikasi. Seperti halnya *UML* adalah sebuah bahasa *standard* untuk pengembangan sebuah *software* yang dapat menyampaikan bagaimana membuat dan membentuk model-model, tetapi tidak menyampaikan

apa dan kapan model yang seharusnya dibuat yang merupakan salah satu proses implementasi pengembangan *software.*

UML tidak hanya merupakan sebuah bahasa pemograman *visual* saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman, seperti *JAVA*, *C++*, *Visual Basic*, atau bahkan dihubungkan secara langsung ke dalam sebuah *object-oriented database*.

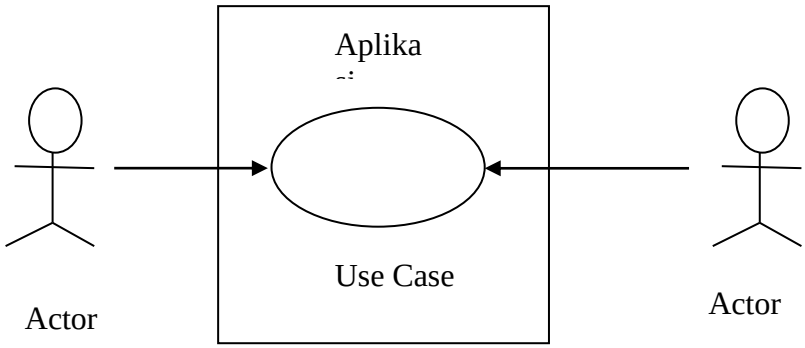
Begitu juga mengenai pendokumentasian dapat dilakukan seperti; *requirements, arsitektur, design, source code, project plan, tests,* dan *prototypes*. Untuk dapat memahami *UML* membutuhkan bentuk konsep dari sebuah bahasa model, dan mempelajari 3 (tiga) elemen utama dari *UML* seperti *building block*, aturan-aturan yang menyatakan bagaimana *building block* diletakkan secara bersamaan, dan beberapa mekanisme umum (*common*).

Alasan mengapa *UML* digunakan adalah, pertama, *scalability* dimana objek lebih mudah dipakai untuk menggambarkan aplikasi yang besar dan komplek. Kedua, *dynamic modeling*, dapat dipakai untuk pemodelan aplikasi dinamis dan *real time.* Sebagaimana dalam tulisan pertama, penulis menjelaskan konsep mengenai obyek, *OOA&D (Obyek Oriented Analyst/ Design)* dan pengenalan *UML*, maka dalam tulisan kedua ini lebih ditekankan pada cara bagaimana *UML* digunakan dalam merancang sebuah pengembangan software yang disertai gambar atau contoh dari sebuah aplikasi.

Adapun jenis-jenis dari tipe diagram *UML* adalah sebagai berikut :

1. *Use Case Diagram*

Use Case adalah deskripsi fungsi dari sebuah aplikasi dari *perspektif* pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah aplikasi dengan dengan aplikasinya sendiri melalui sebuah cerita bagaimana sebuah aplikasi dipakai. Model *use case* adalah bagian dari *requirement*. *Use case* adalah alat bantu terbaik guna menstimulasi pengguna potensial untuk mengatakan tentang suatu aplikasi dari sudut pandangnya. Dengan demikian diharapkan akan bisa dibangun suatu aplikasi yang bisa membantu pengguna, perlu diingat bahwa *use case* mewakili pandangan diluar aplikasi. Diagram *Use Case* menunjukkan 3 aspek dari aplikasi yaitu : *actor, use case dan aplikasi/sub aplikasi boundary. Actor* mewakili peran orang, *aplikasi* yang lain atau alat ketika berkomunikasi dengan *use case*. Berikut gambar notasi *use case :*

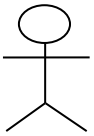


Gambar II.2. Notasi *Use Case* Diagram

(Sumber: Adi Nugroho 2010)

Berikut ini gambar *table* simbol-simbol dari *Use Case* Diagram adalah:

Tabel II.3. Simbol-simbol *Use Case Diagram*

Nama Komponen	**Deskripsi**	**Gambar**
Use Case	Menunjukkan proses yang terjadi pada system	
Actor	Menunjukkan user yang akan menggunakan aplikasi.	
Association	Sebuah garis yang berfungsi menghubungkan *Actor* dengan *Use Case*.	
Extend	Perluasan dari *Use Case* lain jika kondisi atau syarat terpenuhi.	<<extends>> 
Include	Menjelaskan bahwa *Use Case* termasuk dalam *Use Case* lain.	<<include>> 

(Sumber: Adi Nugroho 2010)

2. *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class diagram* menggambarkan struktur dan deskripsi *class, package*

dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

a. Nama (dan *stereotype*)
b. Atribut
c. Metoda

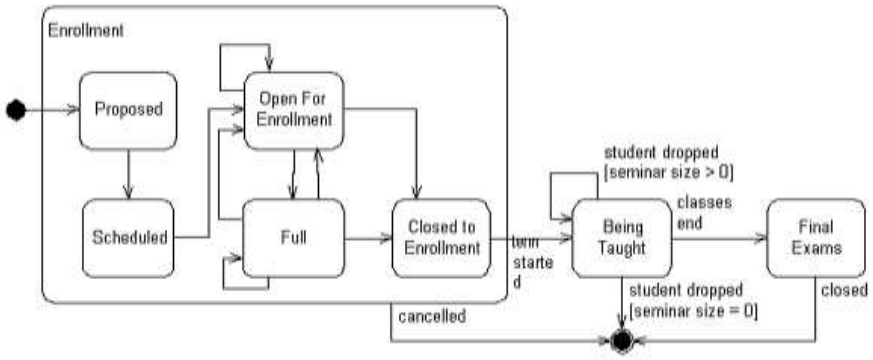
Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
c. *Public*, dapat dipanggil oleh siapa saja

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

3. *Statechart Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada aplikasi sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).



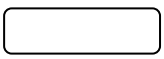
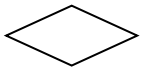
Gambar II.3. *Statechart Diagram*

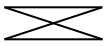
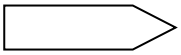
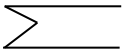
(Sumber: Adi Nugroho 2010)

4. *Activity Diagram*

Activity Diagram adalah teknik *untuk* mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa. Berikut adalah simbol yang ada pada *activity diagram.*

Tabel II.4. Simbol Yang Ada Pada *Activity Diagram*

SIMBOL	KETERANGAN
	Titik Awal
	Titik Akhir
	*Activity*
	Pilihan untuk pengambilan keputusan.
	*Fork* : digunakan untuk menunjukkan kegiatan yang dilakukan secara cenario atau untuk menggabungkan dua kegiatan cenario menjadi satu.

	*Rake* ; menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran Akhir (*Flow Final*)

(Sumber: Adi Nugroho 2010)

5. *Sequence Diagram*

   Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar aplikasi (termasuk pengguna, *display* , dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai *respons* dari sebuah *event* untuk menghasilkan *output* tertentu.

6. *Collaboration Diagram*

   Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram* , tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*

7. *Component Diagram*

   Component Diagram mengandung *component, interface* dan *relationship.* Notasi *component Diagram* dapat dilihat seperti gambar di bawah ini :

Gambar II.4. Notasi *Component Diagram*

(Sumber: Adi Nugroho 2010)

8. *Deployment Diagram*

   Deployment Diagram menunjukkan tata letak sebuah aplikasi secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*. Aplikasi terdiri dari node-node dimana setiap node diwakili untuk sebuah kubus. Garis yang menghubungkan antara 2 kubus menunjukkan hubungan di antara kedua node tersebut. Tipe node bisa berupa *device* yang berwujud *hardware* dan bisa juga *processor*.

II.4. Pengertian Basis Data (*Database*)

Basis data merupakan kumpulan dari data-data yang saling terkait dan saling berhubungan satu dengan yang lainya. Basis data adalah kumpulan-kumpulan *file* yang saling berkaitan.

Menurut Kusrini (2010:2), pengertian Basis Data adalah kumpulan data yang saling berelasi. Data sendiri merupakan fakta mengenai obyek, orang, dan lain-lain. Data dinyatakan dengan nilai (angka, deretan karakter, atau simbol).

Basis data dapat didefinisikan dalam berbagai susut pandang seperti berikut:

1. Himpunan kelompok data yang saling berhubungan yang diorganisasi sedemikian rupa sehingga kelak dapat dimanfaatkan dengan cepat dan mudah.

2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa tanpa pengulangan (*redundancy*) yang tidak perlu, untuk memenuhi kebutuhan.
3. Kumpulan *file*/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpan elektronik.

Beberapa contoh *software Database* antara lain:

1. *Microsoft Access*

2. *SQL Server*

3. *Oracle*

4. *MySQL*, dll.

II.4.1. Tujuan Basis Data

Menurut Kusrini (2010:2), Basis data bertujuan untuk mengatur data sehingga diperoleh kemudahan, ketepatan dan kecepatan dalam pengambilan kembali. Untuk mencapat tujuan, syarat basisdata yang baik adalah sebagai berikut :

a. Tidak adanya *redudansi* dan *inkonsistensi* data

   Redudansis terjadi jika suatu informasi disimpan dibeberapa tempat. Misalnya ada data mahasiswa yang memuat nim, nama, alamat dan atribut lainya, sementara kita punya data lain tentang data KHS mahasiswa yang isinya terdapat NIM, nama, mata kuliah dan nilai. Pada kedua data tersebut kita temukan atribut nama.

b. Kesulitan pengaksesan data

Basis data memiliki fasilitas untuk melakukan pencarian informasi dengan menggunakan *query* ataupun dari tool yang melibatkan tabelnya. Dengan fasilitas ini, bisa segera langsung melihat data dari *software DBMN* nya.

c. Multiple user

Basis data memungkinkan penggunaan data secara bersama-sama oleh banyak pengguna pada saat yang bersamaan atau pada saat yang berbeda. Dengan meletakkan basis data pada bagian *server* yang bisa diakses dari banyak *client*, sudah menyediakan akses kesemua pengguna dari komputer *client* ke sumber informasi yaitu basis data.

II.4.2. Manfaat/Kelebihan Basis Data

Menurut Kusrini (2010:5), Banyak manfaat yang diperoleh dengan menggunakan basis data, Manfaat/Kelebihan Basis Data dan kelebihan basis data diantaranya adalah :

a. Kecepatan dan kemudahan

Dengan menggunakan basis data pengambilan informasi dapat dilakukan dengan cepat dan mudah. Basis data memilki kemampuan dalam mengelompokkan, mengurutkan bahkan perhitungan dengan metematika. Dengan perancangan yang benar maka penyajian informasi dapat dilakukan dengan cepat dan mudah.

b. Kebersamaan pemakai (*sharability)*

Sebuah basis data dapat digunakan oleh banyak user dan banyak aplikasi. Untuk data yang diperlukan oleh banyak bagian/orang, tidak perlu dilakukan

pencacatan dimasing-masing bagian/orang, tetapi cukup dengan satu basis data untuk dipakai bersama.

c. Pemusatan kontrol data

Karena cukup satu basis data untuk banyak keperluan, pengontrolan terhadap data juga cukup dilakukan disatu tempat saja.

d. Efesiensi ruang penyimpanan

Dengan pemakaian bersama, tidak perlu menyediakan tempat penyimpanan diberbagai tempat tetapi cukup satu saja, sehingga ini dapat menghemat ruang penyimpanan yang dimiliki oleh sebuah organisasi.

e. Keakuratan (*Accuracy)*

Penerapan secara tepat acuan tipe data, domain data, keunikan data, hubungan antar data, dan lain-lain, dapat menekan ketidakakuratan dalam pemasukan/penyimpanan data.

f. Ketersediaan *(Availability)*

Dengan basis data, semua data dapat di*backup*, memilah-milah data mana yang masih diperlukan yang perlu disimpan ke tempat lain. Hal ini mengingat pertumbuhan transaksi sebuah organisasi dari lain waktu ke waktu membutuhkan penyimpanan yang semakin besar.

g. Keamanan *(Security)*

Kebanyakan *DBMS* dilengkapi dengan fasilitas manajemen pengguna. Pengguna diberi hak akses yang berbeda-beda sesuai dengan kepentingan dan posisinya. Basis data bisa diberikan *password* untuk membatasi orang yang diaksesnya.

h. Kemudahan dalam pembuatan program aplikasi baru

Penggunaan basis data merupakan bagian dari perkembangan teknologi. Dengan adanya basis data pembuatan aplikasi bisa memanfaatkan kemampuan dari *DBMS*. Sehingga membuat aplikasi tidak perlu mengurusi penyimpanan data, tetapi cukup mengatur *interface* untuk pengguna.

i. Pemakaian secara langsung

Basis data memiliki fasilitas yang lengkap untuk melihat datanya secara langsung dengan *tools* yang disediakan oleh *DBMS*.

j. Kebebasan data

Perubahan dapat dilakukan pada *level DBMS* tanpa harus membongkar kembali program aplikasinya.

k. User View

Basis data menyediakan pandangan yang berbeda-beda untuk tiap-tiap penguna.

II.4.3. Operasi Dasar Database

Menurut Kusrini (2010:9), Beberapa operasi dasar basis data yaitu :

a. Pembuatan basis data
b. Penghapusan basis data
c. Pembuatan *file/tabel*
d. Penghapusan file/tabel
e. Pengubahan *tabel*
f. Penambahan/pengisian
g. Pengambilan data

h. Penghapusan data

II.4.4. Normalisasi

Menurut Samiaji Sarosa (2010:5), Normalisasi adalah teknik yang dirancang untuk merancang tabel basis data relasional untuk meminimalkan duplikasi data dan menghindarkan basis data tersebut anomali. Suatu basis data dikatakan tidak normal jika terjadi 3 (tiga) anomali berikut :

a. *Insertion Anomaly*

   Anomali yang terjadi jika ada data yang tidak bisa disisipkan kedalam *table*.

b. *Update/Modification anomaly*

   Anomali yang terjadi jika ada perubahan pada suatu *item* data maka harus mengubah lebih dari satu baris data.

Langkah-langkah normalisasi sampai pada bentuk *3NF* adalah sebagai berikut :

a. First Nornal Form (1NF)

   Untuk menjadi *1NF* suatu *table* harus memenuhi dua syarat. Syarat pertama tidak ada kelompok data atau *field* yang berulang. Syarat kedua harus ada *primary key (PK)* atau kunci unik, atau kunci yang membedakan satu baris dengan baris yang lain dalam satu *table*. Pada dasarnya sebuah *table* selamat tidak ada kolom yang sama merupakan bentuk *table* dengan *1NF*.

b. Second Nornal Form (2NF)

   Untuk menjadi *2NF* suatu table harus berada dalam kondisi *1NF* dan tidak memilik *partial dependencies. Partial dependencies* adalah suatu kondisi jika atribut *non* kunci (*Non PK*) tergantung sebagian tetapi bukan seluruhnya.

c. Third Nornal Form (3NF)

Untuk menjadi *3NF* suatu *table* harus berada dalam kondisi *2NF* dan tidak memilik *transitive dependencies. Transitive dependencies* adalah suatu kondisi dengan adanya ketergantunga fungsional antara 2 atau lebih atribut *non* kunci (*Non PK*).

II.5. *Microsoft Visual Basic*

Microsoft Visual Basic adalah *software* program untuk membuat program berbasis *Windows*. *Visual basic* memakai bahasa pemrograman *BASIC* (*Beginners All-Purpose Symbolic Instruction Code)* yang merupakan salah satu bahasa pemrograman tingkat tinggi yang sederhana dan mudah untuk dipelajari (Ronald, 2010). *Visual Basic* merupakan salah satu bahasa pemrograman berorientasi objek yang mudah dipelajari.Selain menawarkan kemudahan ,*visual basic* juga cukup andal untuk digunakan dalam pembuatan berbagai aplikasi, terutama aplikasi *database.*

Visual basic merupakan bahasa pemrograman *event drive,* dimana program akan menunggu sampai ada respon dari *user*/pemakai program aplikasi yang dapat berupa kejadian atau *event,* misalnya ketika *user* mengklik tombol atau menekan *enter*.

II.5.1. Sekilas Sejarah *Visual Basic*

Menurut Wahana Komputer (2010:1), Sebelum adanya *Visual Basic*, terlebih dahulu muncul bahasa pemrograman *BASIC* pada awal tahun 1960-an di *Darthmouth College*, Amerika. Bahasa *BASIC* ini merupakan bahasa pemrograman yang mudah dipahami, sehingga para programmer pada masa itu

mempelajari bahasa ini sebagai bahasa pemrograman pertamanya. Hingga pada tahun 1982 *IBM* memperkenalkan *PC* pertamanya dan *Microsoft* pun membuat aplikasi operasi *MS-DOS* untuk digunakan pada *PC* ini.

VB 1.0 dikenalkan pada tahun 1991, pendekatan yg dilakukan untuk menghubungkan bahasa pemrograman dengan *GUI* berasal dari *prototype* yg dikembang oleh "Alan Cooper" yg di sebut *TRIPOD*, Kemudian *Microsoft* mengontrak copper dan asosiasinya utk mengembangkan *tripod* agar dapat digunakan di *windows 3.0* dibawah nama kode *Ruby*.

Tidak lupa *Microsoft* menyertakan pula bahasa *BASIC* di dalam aplikasioperasi *MS-DOS* ini yang dikenal dengan nama *QuickBasic* (*QBASIC*). Tetapi seiring berjalannya waktu, dan munculnya aplikasi operasi *Windows* pada tahun 1990-an dan menggantikan *MS-DOS*, tetapi karena antusias pengguna terhadap bahasa *QBASIC*. *Microsoft* kemudian memperkenalkan bahasa *Basic* khusus *Windows* yang dikenal dengan *Microsoft Visual Basic*. Hingga saat ini, *Visual Basic* sudah hadir dalam 10 versi.

II.5.2. Karakteristik *Visual Basic*

Menurut Wahana Komputer (2010:6), Karakteristik *Visual Basic* adalah sebagai berikut :

1. *Visual Basic is simple*

   Visual Basic menjadi populer karena kemudahan desain *form* secara *visual* dan adanya kemampuan untuk menggunakan komponen-komponen *ActiveX* yang dibuat oleh pihak lain.

2. *Visual Basic is object oriented*

   Pemrograman berorientasi objek adalah pendekatan atau metodologi perancangan aplikasi berdasarkan objek. Metode ini menggantikan metode *procedural* yang telah lama dilakukan. Dalam pemrograman berorientasi objek semua hal dapat dianggab objek.

3. *Visual Basic is distributed*

   Distributed Computing adalah metode komputerisasi dengan menggunakan dan beberapa komputer yang dihubungkan dengan jaringan untuk mengelola tugas-tugas tertentu. *Visual Basic* telah memiliki kemampuan *Networking* yang bagus, yang menjadikan menulis program *networking* seperti mengirim dan menerima data dari sebuah *file*.

4. *Visual Basic is interpreted*

   Karakteristik yang satu ini penting untuk dimengerti oleh pengguna *Visual Basic* yang baru pertama kali mengenal *Visual Basic*. *Visual Basic* adalah bahasa pemrograman yang menggunakan *interpreter* atau penerjemah supaya dapat menjalankan program

5. *Visual Basic is robust*

   Robust artinya dapat diandalkan *Visual Basic* merupakan bahasa pemrograman yang dapat diandalkan.

6. *Visual Basic is Secure*

   Sebuah bahasa pemrograman internet, *Visual Basic* digunakan pada lingkungan *networking* dan terdistribusi.

7. *Visual Basic is Architecture Neutral*

   Program yang dihasilkan oleh *Visual Basic* tidak tergantung oleh arsitektur komputer tertentu.

8. *Visual Basic is Portable*

   Program *Visual Basic* dapat dibawa kemana-mana dan dijalankan dimana-mana.

9. *Visual Basic is Performance*

   Kinerja dan performa pemrograman Bahasa *Visual Basic* ini sering mendapatkan kritikan atau dianggap lambat beberapa *developer*.

II.5.3. Komponen *Microsoft Visual Studio* 2010

Komponen dasar pada *Microsoft Visual Studio 2010* terdiri dari *Menu Bar, Toolbar Standar, Toolbox, Form Design, Solution Explorer, Properties* dan *Error List.* .

1. Menu Bar

Merupakan suatu menu yang terdiri dari 11 menu utama, masing-masing memiliki sub menu dan perintah lengkap dengan *shortcut key.*

2. Toolbar Standar

Merupakan suatu baris menu yang mempunyai fungsi yang sama pada setiap *tool standard* pada umumnya, seperti fungsi untuk menyimpan, mengkopi, menambah project baru, mengatur tampilan program dan masih banyak lagi.

3. Form Design

Merupakan suatu lembar *form* yang berfungsi untuk merancang tampilan aplikasi

secara visual dengan menempatkan kontrol-kontrol yang dipergunakan.

4. *Toolbox*

Merupakan suatu jendela yang berfungsi untuk menampung komponen-komponen standard.

5. *Solution Explorer*

Merupakan suatu jendela yang berfungsi untuk menampilkan object yang digunakan untuk membuat aplikasi seperti: *form*, *class* dan object lainnya.

6. *Properties Windows*

Merupakan suatu jendela yang berfungsi untuk mengatur nilai properties dari masing-masing komponen yang akan digunakan.

7. *Error List*

Merupakan suatu jendela yang berfungsi untuk menampilkan setiap kesalahan dari pembuatan kode program suatu aplikasi.

II.6. *SQL Server*

Menurut Wahana Komputer (2010:5), *SQL Server* adalah terobosan baru dari *microsoft* dalam bidang *database*. *SQL Server* adalah sebuah *DBMS* (*Database Management System*) yang dibuat oleh *microsoft* untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti *IBM* dan *Oracle*. *SQL Server* dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa *SQL Server* membawa beberapa terobosan dalam bidang pengolahan dan

penyimpanan data, beberapa fase penting dalam pengembangan *SQL Server* adalah sebagai berikut :

a. *SQL Server* dirilis pertama kali secara internal Pada tahun 1988, *Microsoft* mengeluarkan versi pertama dari *SQL Server*. Pada saat itu masih didesain untuk *platform OS/2* dan *didevelop* bersama antara *Microsoft* dengan *Sybase*
b. Versi *Windows* dirilis pada 1993 untuk *windows* 93.
c. Versi 6.0 dirilis *Microsoft* pada tahun 1995 dari *SQL Server*.
d. Versi 6.5 dirilis *Microsoft* Pada tahun 1996, merilis *SQL Server*
e. Versi *SQL Server 6.5 Enterprise Edition* dirilis Pada tahun 1997.
f. Versi *SQL Server* 7.0 dirilis *Microsoft* pada tahun 1998 dan *database engine*-nya ditulis ulang agar lebih optimal.
g. Akhirnya tahun 2000 *Microsoft* mengeluarkan *SQL Server* 2000 yang merupakan versi yang banyak digunakan

SQL server adalah *RDBMS (Relational Database Management System)* yang dapat menangani data yang bervolume besar. Meskipun begitu, tidak menuntut *resource* yang besar. *SQL server* termasuk *database* yang paling popular diantara *database-database* lainnya.

II.6.1 Kelebihan dan Keuntungan Memakai *SQL Server*

Menurut Wahana Komputer (2010:6), Dalam dunia programming ada beberapa *database* yang sering digunakan antara lain *SQL Server, Mysql, Oracle, PostgreSQL, MSql*, dan lain-lain. Ketika dibandingkan antara *SQL Server* dengan

aplikasi manajemen *database* yang lain, *SQL Server* memiliki beberapa kelebihan dan keuntungan dibandingkan *database* lain diantaranya adalah :

a. Banyak ahli berpendapat bawah *SQL Server* merupakan *server* tercepat
b. *SQL Server* merupakan aplikasi manajemen *database* yang *open source* yaitu *software* ini bersifat *free* atau bebas digunakan oleh perseorangan atau instansi tanpa harus membeli atau membayar kepada pembuatnya.
c. *SQL Server* mempunyai ferporma yang tinggi namun simpel
d. *SQL Server* dapat diakses melalaui protocol *ODBC (Open Database Connectivity)* buatan *Microsoft*. Ini menyebabkan *Mysql* dapat diakses oleh banyak *software*.
e. Semua *client* dapat mengakses *Server* dalam satu waktu tanpa harus menunggu yang lain mengakses *database.*
f. *SQL Server* dapat diakses dari semua tempat di internet dengan hak akses tertentu
g. *SQL Server* merupakan *database* yang mampu menyimpan data berkapasitas besar, sampai berukuran *gigabyte.*
h. *SQL Server* dapat berjalan diberbagai *operating aplikasi* seperti *Linux, Windows, Solaris* dan lain-lain.
i. lain.