BAB II
TINJAUAN PUSTAKA

2.1 Teori Umum

2.1.1 Pengertian Komputer

Hartono (2013:27), "Komputer adalah sebuah mesin yang dapat dikendalikan melalui perintah (*programmable machine*) yang dirancang untuk secara otomatis melakukan serangkaian urutan penghitungan (*arithmetic*) atau proses-proses yang diurutkan secara logis."

Menurut Sanders dalam Sutarman (2012:2), "Komputer adalah system elektronik untuk memanipulasi data yang cepat dan tepat serta dirancang dan diorganisasikan agar secara otomatis menerima dan menyimpan data *input*, memprosesnya, dan menghasilkan *output* di bawah pengawasan suatu langkah-langkah instruksi program yang tersimpan pada memori (*stored program*)."

2.1.2 Pengertian Perangkat Lunak (*Software*)

Sukamto dan Shalahuddin (2014:2), "Perangkat lunak (*software*) adalah program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaan (*user manual*).

2.1.3 Pengertian Program

Menurut Sutarman (2012:3), " Program adalah barisan perintah/ instruksi yang disusun sehingga dapat dipahami oleh komputer dan kemudian dijalankan sebagai barisan perhitungan numerik, dimana barisan perintah tersebut berhingga, berakhir, dan menghasilkan output."

2.1.4 Pengertian Data

Hartono (2013:15), "Data adalah hasil pengukuran dan pencatatan terhadap fakta tentang sesuatu, keadaan, tindakan, atau kejadian."

Asropuddin (2013:22) menjelaskan, "Data adalah kumpulan dari angka-angka maupun karakter-karakter yang tidak memiliki arti."

Sutabri (2012:25), “Data merupakan bentuk mentah yang belum dapat bercerita banyak sehingga perlu diolah lebih lanjut.”

Sutarman (2012:3), “Data adalah fakta dari sesuatu pernyataan yang berasal dari kenyataan, di mana pernyataan tersebuut merupakan hasil pengukuran dan pengamatan.”

2.1.5 Pengertian Basis Data (*Database*)

Hidayatullah (2014:137), “Basis data dapat didefinisikan sebagai himpunan kelompok data yang saling berhubugan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.”

Sukamto dan Shalahuddin (2014:43), “Basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan.”

2.2 Teori Judul

2.2.1 Pengertian Aplikasi

Asropuddin (2013:7), “Aplikasi dibuat untuk mengerjakan atau menyelesaikan masalah-masalah khusus.”

Menurut Sutabri (2012:147), “Aplikasi adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya.”

2.2.2 Pengertian Pengolahan Data

Sutarman (2012:4), “Pengolahan data adalah perhitungan/transformasi data menjadi informasi yang diolah secara elektronik dengan menggunakan komputer.”

Menurut Kristanto (2008:8), pengolahan data adalah waktu yang digunakan untuk menggambarkan perubahan bentuk data menjadi informasi yang memiliki kegunaan.

2.2.3 Pengertian Keuangan

Menurut Ridwan dan Inge, Keuangan merupakan ilmu dan seni dalam mengelola uang yang mempengaruhi kehidupan setiap orang dan setiap organisasi. Keuangan berhubungan dengan proses, lembaga, pasar, dan instrumen yang terlibat dalam transfer uang diantara individu maupun antara bisnis dan pemerintah.

2.2.4 Pengertian Kas dan Kas Kecil

Sugiono (2010:149), Kas merupakan alat pembayaran yang siap dan bebas dipergunakan untuk kegiatan perusahaan. Kas pada perusahaan dapat berupa kas yang ada di perusahaan itu sendiri serta kas yang ada di bank.

Kas kecil merupakan dana yang disiapkan untuk membayar kebutuhan yang segera dan dalam jumlah yang relatif kecil atau dengan kata lain pembayaran ini terlalu kecil bila dibayar dengan menggunakan cek atau giro. Dana kas kecil dipisahkan dari kas besar dan diserahkan pertanggungjawabannya kepada pemegang kas kecil. Mengenai besarnya kas kecil tidak ada standardisasi, namun dalam praktek sering didasarkan pada pengalaman empiris akan jumlah kebutuhan dan pengeluaran yang sifatnya relatif tetap dari masing-masing unit kerja dalam jangka waktu tertentu.

2.2.5 Pengertian Operasional

Menurut Widjono, Operasional ialah batasan pengertian yang dijadikan sebagai pedoman untuk melakukan sesuatu kegiatan ataupun pekerjaan.

2.2.6 Pengertian PT (Perseroan Terbatas)

Winarti dan Syahrizal (2012:1), “Perseroan Terbatas adalah badan hokum yang merupakan persekutuan modal yang dilakuka oleh minimal dua orang dengan tanggung jawab yang hanya berlaku pada perusahaan serta, tanpa melibatkan harta pribadi atau perseorangan yang ada di dalamnya (para pemegang saham), didirikan berdasarkan perjanjian, melakukan kegiatan usaha dengan modal dasar yang seluruhnya terbagi dalam saham dan memenuhi persyaratan

yang ditetapkan dalam Undang-Undang Republik Indonesia Nomor 40 Tahun 2007 tentang Perseroan Terbatas serta peraturan pelaksanaannya."

2.2.7 Pengertian Aplikasi Pengolahan Data Keuangan Operasional pada PT Ananda Energi Utama

Aplikasi Pengolahan Data Keuangan Operasional pada PT Ananda Energi Utama merupakan sebuah *software* yang dibangun untuk melakukan tugas pengolahan data keuangan bagian operasioanl pada PT Ananda Energi Utama yang mana bisa membantu staf keuangan untuk melakukan pencatatan uang keluar dan uang masuk yang digunakan untuk kebutuhan operasional yang besar atau pun yang kecil untuk kegiatan sehari-hari serta dapat menghasilkan laporan keuangan dengan cepat dan mudah.

2.3 Teori Khusus

2.3.1 Pemograman Berorientasi Objek

2.3.1.1 Pengertian Pemograman Berorientasi Objek

Menurut Sukamto dan Shalahuddin (2014:100), "Berorientasi Objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya."

Menurut Sutarman (2012:165), "Pemograman Berorientasi Objek adalah suatu program komputer dapat dipandang sebagai kumpulan dari unit tunggal atau objek yang dapat melakukan aksi atau tindakan satu sama lain."

2.3.1.2 Konsep Dasar Berorientasi Objek

Menurut Sukamto dan Salahuddin (2014:104), Konsep dasar yang harus dipahami tentang metodologi berorientasi objek:

a. Kelas (*class*)

   Secara teknis, kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodelogi berorientasi objek. Kelas secara fisik adalah berkas

atau *file* yang berisi kode program, di mana kode program merupakan semua hal yang terkait dengan nama kelas.

b. Objek (*Object*)

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya.

c. Metode (*method*)

Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metodeatau operasi dapat berasal dari *event*, aktivitas atau aksi keadaan, fungsi , atau kelakuan dunia nyata. Contoh metode atau operasi misalnya read, write, move, copy, dan sebagainya.

d. Atribut (*attribute*)

Atribut dari sebuah kelas adalah variable global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek.

e. Abstraksi (*abstraction*)

Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

f. Enkapsulasi (*encapsulation*)

Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

g. Pewarisan (*inheritance*)

Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

h. Antarmuka (*interface*)

Antarmuka sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi.

i. *Reusability*

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

j. Generalisasi dan Spesialisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.

k. Komunikasi Antar Objek

Komunikasi antar objek dilakuka lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.

l. Polimorfisme (*polymorphism*)

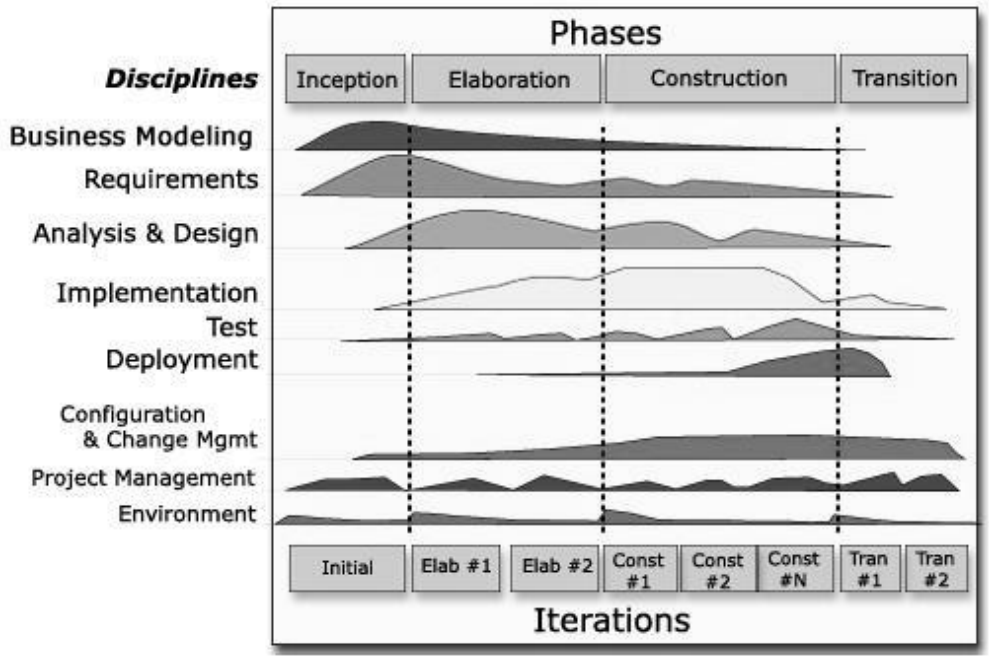
Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

m. *Package*

Package adalah sebuah container atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.3.2 Metodologi RUP (*Rational Unified Process*)

Menurut Suryana, *Rational Unified Process* (RUP) merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai *best practises* yang terdapat dalam industri pengembangan perangkat lunak. Ciri utama metode ini adalah menggunakan *use-case driven* dan pendekatan iteratif untuk siklus pengembangan perangkat lunak. Gambar dibawah menunjukkan secara keseluruhan arsitektur yang dimiliki RUP.



Gambar 2.1 Arsitektur *Rational Unified Process*

RUP menggunakan konsep *object oriented*, dengan aktifitas yang berfokus pada pengembangan model dengan menggunakan *Unified Model Language* (UML). Melalui gambar diatas dapat dilihat bahwa RUP memiliki, yaitu:

a. Dimensi pertama digambarkan secara horizontal. Dimensi ini mewakili aspek-aspek dinamis dari pengembangan perangkat lunak. Aspek ini dijabarkan dalam tahapan pengembangan atau fase. Setiap fase akan memiliki suatu *major milestone* yang menandakan akhir dari awal dari phase selanjutnya. Setiap phase dapat berdiri dari satu beberapa iterasi. Dimensi ini terdiri atas *Inception*, *Elaboration*, *Construction*, dan *Transition*.
b. Dimensi kedua digambarkan secara vertikal. Dimensi ini mewakili aspek-aspek statis dari proses pengembangan perangkat lunak yang dikelompokkan ke dalam beberapa disiplin. Proses pengembangan perangkat lunak yang dijelaskan kedalam beberapa disiplin terdiri dari empat elemen penting, yakni *who is doing*, *what*, *how* dan *when*. Dimensi ini terdiri atas *Business Modeling*, *Requirement, Analysis and Design, Implementation, Test, Deployment, Configuration* dan *Change Manegement, Project Management, Environtment.*

2.3.2.1 Penerapan Tahap Metodologi Pengembangan Perangkat Lunak dengan RUP

Dalam *Rational Unified Process* terdapat empat tahap pengembangan perangkat lunak yaitu:

a. *Inception*

Pada tahap ini pengembang mendefinisikan batasan kegiatan, melakukan analisis kebutuhan user, dan melakukan perancangan awal perangkat lunak (perancangan arsitektural dan *use case*). Pada akhir fase ini, prototipe perangkat lunak versi *Alpha* harus sudah dirilis

b. *Elaboration*

Pada tahap ini dilakukan perancangan perangkat lunak mulai dari menspesifikasikan fitur perangkat lunak hingga perilisan prototipe versi *Betha* dari perangkat lunak.

c. *Construction*

Pengimplementasian rancangan perangkat lunak yang telah dibuat dilakukan pada tahap ini. Pada akhir tahap ini, perangkat lunak versi akhir yang sudah disetujui administrator dirilis beserta dokumentasi perangkat lunak.

d. *Transition*

Instalasi , *deployment* dan sosialisasi perangkat lunak dilakukan pada tahap ini.

2.3.2.2 Aliran Kerja Utama RUP

Adapun aliran kerja utama pada metodologi RUP adalah sebagai berikut:

a. Pemodelan Bisnis (*Bussines Modeling*)

Mendeskripsikan struktur dan proses-proses bisnis organisasi.

b. Kebutuhan (*Requirement*)

Mendefinisikan kebutuhan perangkat lunak dengan menggunakan metode *use case.*

c. Analisis dan Perancangan (*Analysis and Design*)

Mendeskripsikan berbagai arsitektur perangkat lunak dari berbagai sudut pandang.

d. Implementasi (*Implementation)*

Menuliskan kode-kode program, menguji, dan mengintegrasikan unit-unit programnya.

e. Pengujian (*Test*)

Mendeskripsikan kasus uji, prosedur, dan alat ukur pengujian.

f. Deployment

Menangani konfigurasi sistem yang akan diserahkan.

2.3.2.3 Aliran Kerja Pendukung RUP

Adapun aliran kerja pendukung RUP adalah sebagai berikut:

a. Manajemen konfigurasi dan perubahan (*configuration and change management*) mengendalikan perubahan dan memelihara artifak-artifak proyek.

b. Manajemen proyek (*Project Management*)

Mendeskripsikan berbagai stategi pekerjaan dengan proses yang berulang.

c. Lingkungan (*Environment*)

Menangani infrastuktur yang dibutuhkan untuk mengembangkan sistem.

2.3.3 *Unified Modelling Language* (UML)

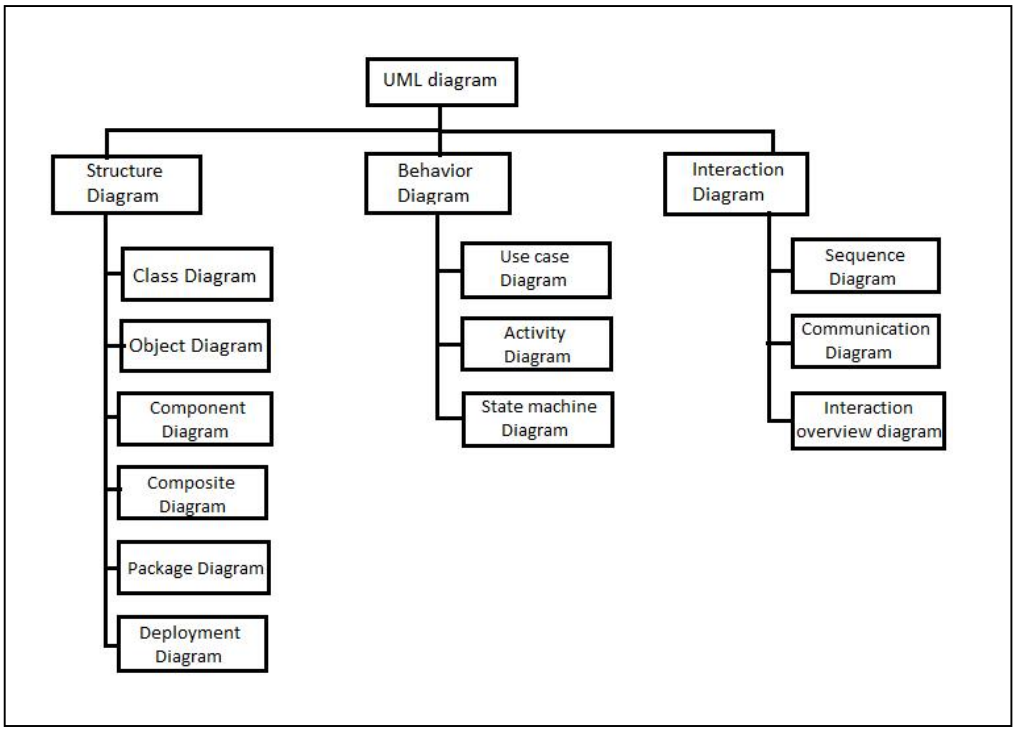
Menurut Sukamto dan Shalahuddin (2014:133), "*UML* (*Unified Modelling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek."

Menurut Widodo dan Herlawati (2011:6), "*UML* singkatan dari *Unified Modeling Language* yang berarti bahasa permodelan standar."

Asropuddin (2013:102), "UML singkatan dari "Unified Modeling Language" adalah bahasa pemrograman yang digunakan untuk perangkat lunak berorientasi objek."

2.3.3.1 Macam-macam Diagram UML

Menurut Sukamto dan Salahuddin (2014:140), UML terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar di bawah ini.



Gambar 2.2 Macam-macam Diagram UML

Berikut ini penjelasan singkat dari pembagian kategori tersebut.

a. Structure Diagram

Yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

b. Behavior Diagram

Yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

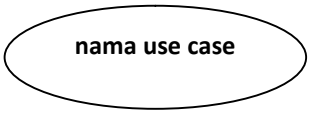
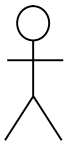
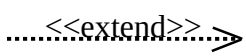
c. Interaction Diagram

Yaitu kumpulan diagram yang digunakan untuk menggambarkan interkasi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

2.3.4 *Use Case Diagram*

Menurut Sukamto dan Shalahuddin (2014:155), "*Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu."

Tabel 2.1 Simbol-simbol Diagram *Use Case*

No.	Simbol	Deskripsi
1.	*Use Case* 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*.
2.	Aktor / *Actor*  **nama aktor**	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
3.	Asosiasi / *Association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4.	Ekstensi / *Extend* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.

Lanjutan **Tabel 2.1** Simbol-simbol Diagram *Use Case*

No.	Simbol	Deskripsi
5.	Generalisasi / *Generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6.	Menggunakan / *Include* / *Uses* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. *Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan.

Sumber: Buku Karangan Rosa A.S dan M. Shalahuddin (2014:156)

2.3.5 *Class Diagram*

Widodo dan Herlawati (2011:3) menjelaskan, "*Class diagram* adalah penggamabaran satu set objek yang memiliki atribut dan *behavior* yang sama."

Menurut Sukamto dan Shalahuddin (2014:141), bahwa Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

a. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.

b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Tabel 2.2 Simbol-simbol Diagram Kelas

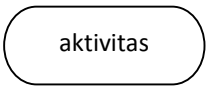
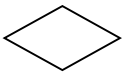
No.	Simbol	Deskripsi
1.	Kelas **nama_kelas** +atribut +operasi()	Kelas pada struktur sistem.
2.	Antarmuka / *Interface* **nama_interface**	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
3.	Asosiasi / *Association*	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
4.	Asosiasi berarah / *Directed association*	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
5.	Generalisasi	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus).
6.	Kebergantungan / *Dependency*	Relasi antarkelas dengan makna kebergantungan antarkelas.
7.		Relasi antarkelas dengan makna semua-bagian (*whole-part*).

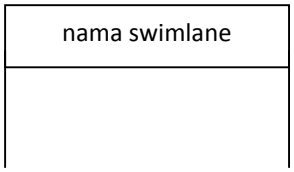
Sumber: Buku Karangan Rosa A.S dan M. Shalahuddin (2014:146)

2.3.6 *Activity Diagram*

Menurut Sukamto dan Shalahuddin (2014:161), Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses atau menu yang ada pada perangkat lunak. Diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Tabel 2.3 Simbol-simbol Diagram Aktivitas

No.	Simbol	Deskripsi
1.	Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan / *Decision* 	Asosiasi percabangan dimana jika ada ada pilihan aktivitas lebih dari satu.
4.	Penggabungan / *Join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.	Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

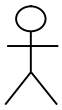
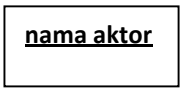
6.	Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

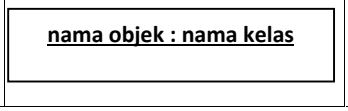
Sumber: Buku Karangan Rosa A.S dan Shalahuddin (2014:162)

2.3.7 *Sequence Diagram*

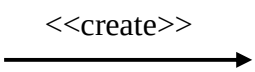
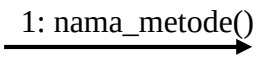
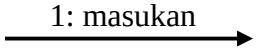
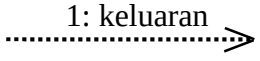
Menurut Sukamto dan Shalahuddin (2014:165), *Sequnce Diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirmkan dan diterima antar objek. Oleh karena itu, untuk menggambar *sequence diagram* harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat *sequence diagram* juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

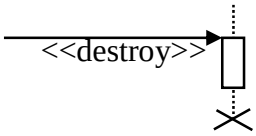
Tabel 2.4 Simbol-simbol *Sequnce Diagram*

No.	Simbol	Deskripsi
1.	Aktor  **nama actor** atau  tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.

2.	Garis hidup / *Lifeline*	Menyatakan kehidupan suatu objek.
3.	Objek 	Menyatakan objek yang berinteraksi pesan.

Lanjutan **Tabel 2.4** Simbol-simbol *Sequnce Diagram*

No.	Simbol	Deskripsi
4.	Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya.
5.	Pesan tipe *create* 	Menyatakan suatu objek membuat objek lain, arah panah mengarah pada objek yang dibuat.
6.	Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri.
7.	Pesan tipe *send* 	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
8.	Pesan tipe *return* 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.

9.	Pesan tipe *destroy*  	Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri,sebaiknya jika ada *create* maka ada *destroy*.

Sumber: Buku Karangan Rosa A.S dan Shalahuddin (2014:165)

2.4 Teori Program

2.4.1 Pengenalan Microsoft Visual Basic.NET

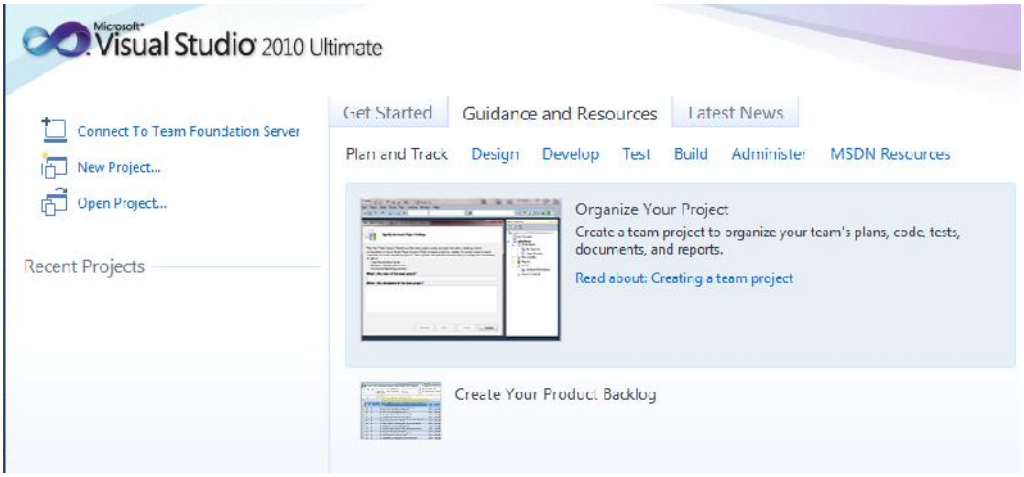
Hidayatullah (2014:5), Visual Basic.NET adalah Visual Basic yang direkayasa kembali untuk digunakan pada *platform* .NET sehingga aplikasi yang dibuat menggunakan Visual Basic.NET dapat berjalan pada system computer apa pun, dan dapat megambil data dari *server* dengan tipe apa pun asalkan terinstal .NET framework.

Berikut ini perkembangan Visual Basic.NET:

a. Visual Basic.NET 2002 (VB 7.0)
b. Visual Basic.NET 2003 (VB 7.1)
c. Visual Basic 2005 (VB.8.0)
d. Visual Basic 2008 (VB 9.0)
e. Visual Basic 2010 (VB 10.0)
f. Visual Basic 2012 (VB 11.0)
g. Visual Basic 2013

2.4.2 Mengenal IDE (*Integrated Development Environment*) Visual Basic 2010

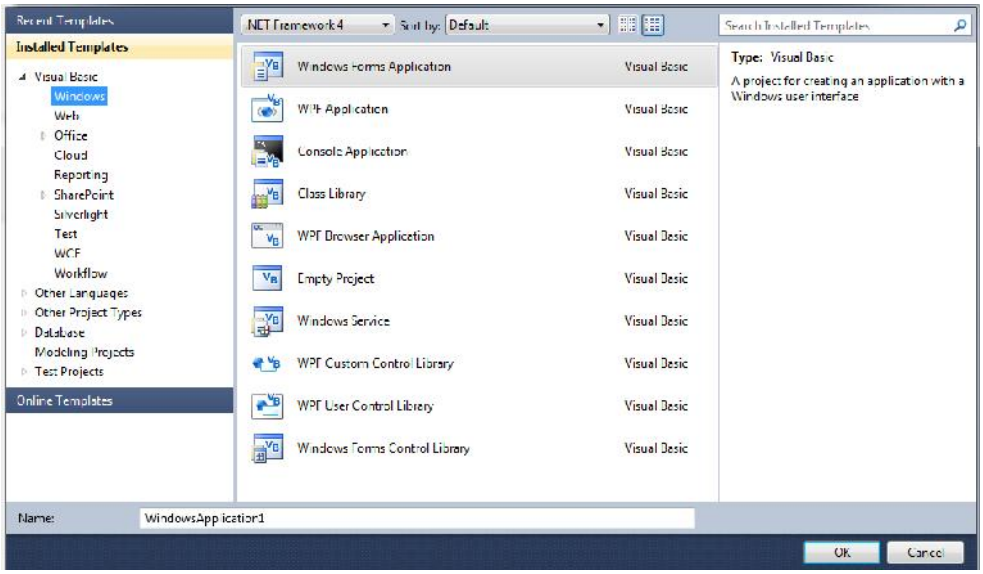
Pada waktu Visual Basic 2010 dijalankan, akan tampil sebuah Start Page seperti berikut:



Gambar 2.3 *start page* dari Visual Studio 2010

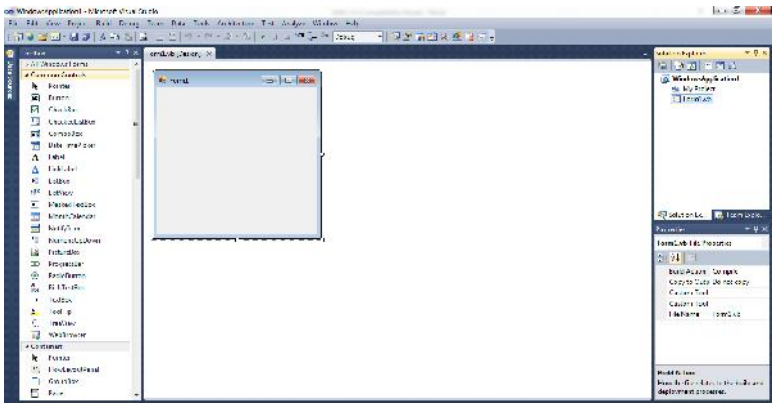
Untuk membuka proyek yang ada gunakan tombol **Open Project** atau langsung mengklik pada daftar proyek yang ditampilkan sedangkan untuk membuat sebuah proyek baru, klik tombol **New Project.**

Setelah itu akan muncul kotak dialog **New Project.** Pada kotak pilih **Other Languages > Visual Basic > Windows > Windows Forms Application**. Untuk memberi nama proyek dapat dilakukan pada bagian **Name**, tentukan posisi penyimpanan *file-file* proyek dan tentukan nama *solution*-nya dan tekan **OK**. Selanjutnya muncul **Visual Basic 2010 IDE** tempat untuk membangun aplikasi Visual Basic.NET.



Gambar 2.4 Kotak Dialog *New Project*

Pada **IDE Visual Studio 2010** untuk *Windows Application* secara *default* telah terdapat sebuah *form*. *Form* tersebut bernama Form1. Pada *form* inilah tempat meletakkan kontrol-kontrol atau komponen-komponen untuk membuat sebuah aplikasi Windows Form dan kontrol-kontrol dari aplikasi inilah yang biasanya disebut dengan GUI (Graphical User Interface). Jadi user akan berinteraksi dengan sebuah program aplikasi melalui GUI.



Gambar 2.5 IDE Visual Studio 2010

2.4.3 Tools pada Microsoft Visual Studio 2010

a. Menu Bar

Menu bar adalah bagian dari IDE yang terdiri atas perintah-perintah untuk mengatur IDE, mengedit kkode, dan mengeksekusi program.

Gambar 2.6 *Menu Bar*

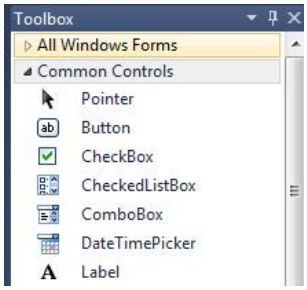
b. *Toolbar*

Toolbar fungsinya sama seperti menu. Bedanya pada *toolbar* pilihan-pilihan berbentuk icon.

Gambar 2.7 *Tool Bar*

c. *Toolbox*

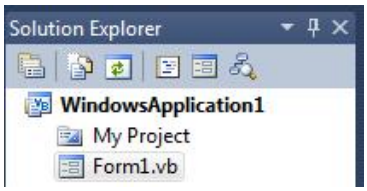
Toolbox adalah tempat di mana control-kontrol dan komponen-komponen diletakkan.



Gambar 2.8 *ToolBox*

d. *Solution Explorer*

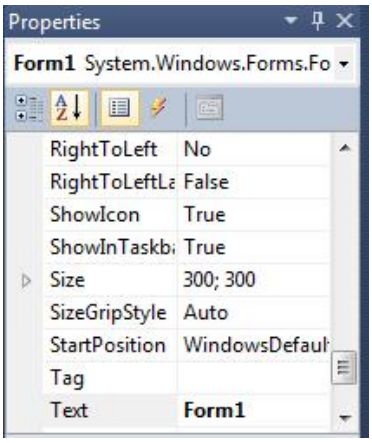
Solution Explorer memberikan tampilan daftar *file-file* dari proyek yang sedang dibuat.



Gambar 2.9 *Solution Explorer*

e. *Properties Window*

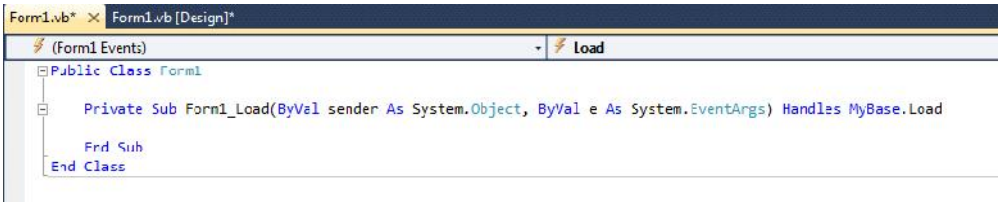
Properties Windows adalah tempat menyimpan *property* dari setiap objek kontrol dan komponen.



Gambar 2.10 *Properties Window*

f. *Code Editor*

Code Editor adalah tempat di mana kita meletakkan atau menuliskan kode program aplikasi kita.



Gambar 2.11 *Code Editor*

2.4.4 Tipe Data dalam VB.NET

Tabel 2.5 Tipe data dalam VB.NET

Tipe Data	Tipe Struktur dari Common Language Runtime	Jumlah Alokasi Memori	Range Nilai
Boolean	System.Boolean	Tergantung platform	True or False.
Byte	System.Byte	1 byte	0 sampai 255 (*unsigned).*
Char	System.Char	2 bytes	0 sampai 65535 (*unsigned).*
Date	System.DateTime	8 bytes	0:00:00 January 1, 0001 sampai 11:59:59 PM December 31,9999.
Decimal	System.Decimal	16 bytes	0 sampai +/- 79,228,162,514,264,337,593,543,950,335 dengan tidak ada titik desimal; 0 sampai +/- 7.9228162514264337593543950335 dengan 28 angka decimal; angka bukan nol paling kecil adalah 0.0000000000000000000000000001 (+/-IE-28).
Double (double-precision floating-point)	System.Double	8 bytes	-1.79769313486231570E+308 sampai4.94065645841246544E-324 untuk nilai negatif; 4.94065645841246544E-324 sampai1.79769313486231570E+308 untuk nilai positif.

Integer	System.Int32	4 bytes	-2,147,483,648 sampai 2,147,483,647.
Long (long integer)	System.Int64	8 bytes	-9,223,372,036,854,775,808 sampai 9,223,372,036,854,775,807.
Object	System.Object (class)	4 bytes pada 32-bit platform 8 bytes pada 64-bit platform	Setiap tipe dapat disimpan dalam sebuah variable bertipe object.
Short	System.Int16	2 bytes	-32,768 sampai 32,767.

Lanjutan **Tabel 2.5** Tipe data dalam VB.NET

Tipe Data	Tipe Struktur dari Common Language Runtime	Jumlah Alokasi Memori	Range Nilai
Single (single-precision floating-point)	System.Single	4 bytes	-3.4028235E+38 sampai -1.401298E-45 untuk nilai negatif; 1.401298E-45 sampai 3.4028235E+38 untuk nilai positif.
String (variabl-length)	System.String (class)	Tergantung pada implementasi platform	0 sampai kira-kira 2 miliar karakter Unicode.
User-Defined Type (structure)	(diturunkan dari System.ValueType)	Tergantung pada platform	Setiap anggota struktur sesuai dengan tipe datanya dan tidak tergantung pada anggota yang lain.

Sumber: Buku Karangan Priyanto Hidayatullah (2014:81)

Mendeklarasikan Variabel

a. Deklarasi Variabel Lokal

```
Dim <nama_variabel> As <Tipe_data>
```

Contoh:

```
Dim index As Integer
```

b. Mengisi nilai kedalam sebuah

variable *<nama_variabel> = <nilai>*

Contoh:

index=100

Dim index As Integer =100

c. Deklarasi Konstanta

Const <nama_konstanta> As <tipe_data> = <nilai>

2.4.5 Operator

Operator adalah simbol yang berfungsi untuk mengoperasikan satu atau dua *operand*. *Operand* adalah sesuatu yang dioperasikan. Operator yang menangani satu operand disebut operator *unary*.

2.4.5.1 Operator Aritmatika

Operator aritmatika adalah operator untuk operasi matematika.

Tabel 2.6 Operator Aritmatika

Operator	Operasi	Keterangan
( )	Pengelompokkan	Tanda kurung buka dan kurung tutup digunakan untuk mengelompokkan operasi yang akan dikerjakan duluan.
^	Pemangkatan/ Eksponensial	Digunakan untuk operasi pemangkatan.
+ , -	Tanda/ Nilai	Biasanya tanda positif (+) tidak ditulis lagi.
*	Perkalian	
/	Pembagian	Pada operasi pembagian menggunakan hasil sampai dengan decimal
\	Pembagian integer	Pembagian yang hasilnya bilangan integer atau hanya hasil bulatnya yang

		digunakan.
Mod	Modulus	Hasilnya adalah sisa pembagian *integer*.
+	Penjumlahan	
-	Pengurangan	

Sumber: Buku Karangan Priyanto Hidayatullah (2014:98)

2.4.5.2 Operator Relasi (Pembanding)

Operator akan membandingkan di antara dua *operand*. Nilai keluarannya berupa *Boolean*.

Tabel 2.7 Operator Relasi (Pembanding)

Operator	Operasi
=	Sama dengan
< >	Tidak sama dengan
>	Lebih dari
<	Kurang dari
>=	Lebih dari sama dengan
<=	Kurang dari sama dengan

Sumber: Buku Karangan Priyanto Hidayatullah (2014:99)

2.4.5.3 Operator *Bitwise*

Guna dari operator *bitwise* adalah memanipulasi bit secara individu untuk membuat nilai dari tipe *integer* (*Byte, Short, Integer*, dan *Long*).

Tabel 2.8 Operator *Bitwise*

Operator	Kependekan dari	Artinya	Contoh	Hasil
And	Bitwise And	Baik sisi kanan mauoun kiri dari operator adalah 1	0 And 0 0 And 1 1 And 0 1 And 1	0 0 0 1
Or	Inklusif Bitwise Or	Salah satu atau kedua sisi dari	0 Or 0 0 Or 1	0 1

		operator adalah 1	1 Or 0 1 Or 1	1 1
Xor	Eksklusif Bitwise Or	Salah satu dari sisi kanan atau kiri dari operator adalah 1, tapi tidak keduanya.	0 Xor 0 0 Xor 1 1 Xor 0 1 Xor 1	0 1 1 0

Sumber: Buku Karangan Priyanto Hidayatullah (2014:99)

2.4.5.4 Operator Logika

Operator logika adalah operator yang membandingkan ekspresi *Boolean*.

Tabel 2.9 Operator Logika

Operator	Keterangan
And	Hasilnya *True* jika kedua *operand* bernilai *True*. Selain itu hasilnya *False.*
Or	Hasilnya *True* jika salah satu atau kedua *operand* bernilai *True*. Selain itu hasilnya *False*.
Xor	Hasilnya *True* jika hanya salah satu *operand* bernilai *True*. Selain itu hasilnya *False*.
AndAlso	Hasilnya *True* jika kedua *operand* bernilai *True*. Selain itu hasilnya *False*.
OrElse	Hasilnya *True* jika salah satu atau kedua *operand* bernilai *True*. Selain itu hasilnya *False*.
Not	Merupaka operator *unary*. Hasilnya *True* jika *operand False*. *False* operand *True*.

Sumber: Buku Karangan Priyanto Hidayatullah (2014:101)

2.4.6 Pengertian MySQL

Winarno (2014:2), "MySQL merupakan tipe data relasional yang artinya MySQL menyimpan datanya dalam bentuk table-tabel yang saling berhubungan."