BAB II

TINJAUAN PUSTAKA

2.1 Profil Unikom

Profil Unikom berisi penjelasan yang memaparkan tentang sejarah, logo, visi, misi, tujuan, dan struktur organisasi Unikom.

2.1.1 Sejarah Unikom

Universitas Komputer Indonesia (UNIKOM) secara resmi berdiri pada hari Selasa, tanggal 8 Agustus 2000 berdasarkan Surat Keputusan Menteri Pendidikan Nasional nomor 126/D/0/2000.

Awalnya dimulai pada bulan Juli tahun 1994 ketika didirikan Lembaga Pendidikan Komputer Indonesia Jerman, disingkat LPKIG, bertempat di jalan Dipati Ukur 102 Bandung. Dengan 1 ruang kelas berkapasitas 50 orang dan 1 laboratorium komputer dengan 25 unit komputer, Lembaga ini membuka program pendidikan 1 tahun dengan 5 program studi yaitu Ahli Komputer Aplikasi Bisnis, Ahli Komputer Keuangan & Perbankan, Ahli Komputer Akuntansi & Perpajakan, Ahli Komputer Manajemen & Pemasaran dan Sekretaris Eksekutif. Jumlah peserta pendidikan pada tahun pertama ini sebanyak 233 siswa.

Pada tahun kedua, 1995, dibuka jenjang pendidikan 3 tahun untuk memenuhi animo siswa tahun pertama yang ingin memperdalam ilmunya, disamping pemikiran jangka panjang pengembangan institusi. Pada tahun ini juga

dibuka program studi baru, meliputi : Ahli Komputer Teknik Informatika, Ahli Komputer Manajemen Informatika dan Sekretaris Eksekutif. Ruang kelas ditambah menjadi 2 buah dan laboratorium komputer menjadi 2 buah dengan jumlah siswa sebanyak 457 orang.

Pada tahun ketiga, 1996, dilakukan penambahan gedung kuliah baru bertempat di jalan Dipati Ukur 116 (gedung FISIP sekarang), sekaligus pemindahan pusat administrasi dan perkantoran. Digedung baru ini dilakukan penambahan 1(satu) Lab. Komputer, 5 (lima) Ruang Kuliah, Ruang Dosen dan Ruang Kemahasiswaan. Jumlah siswa dari tahun 1996 hingga tahun 1998 bertambah dari 632 orang menjadi 1184 orang.

Pada tahun kelima, 1998, dimulai pembangunan Kampus baru (Gedung Rektorat/Kampus-1 sekarang) berlantai 6 (enam) di jalan Dipati Ukur 114. Pembangunan Kampus baru ini dapat diselesaikan pada bulan Agustus 1999, sehingga pada awal perkuliahan bulan September 1999 telah dapat digunakan.

Mencermati dinamika peserta didik dan pengembangan Institusi kedepan, pada tanggal 24 Desember 1998 dibentuklah Yayasan Science dan Teknologi dan dilanjutkan dengan pengajuan pendirian STIMIK IGI dan STIE IGI ke DIKTI.

Pada bulan Juli 1999 STIE IGI diresmikan dengan keluarnya SK Mendiknas no. 119/D/O/1999 dengan 5 program studi : Akuntansi S1, Manajemen S1, Manajemen Pemasaran D3, Keuangan Perbankan D3 serta Akuntansi D3.

Pada bulan Agustus 1999 STIMIK IGI diresmikan dengan keluarnya SK Mendiknas no. 143/D/O/1999 dengan 5 program studi : Teknik Informatika S1,

Manajemen Informatika D3, Teknik Komputer D3, Komputerisasi Akuntansi D3 serta Teknik Informatika D3.

Agar Sistem Pendidikan lebih Efisien, Efektif, Produktif dengan Struktur Organisasi yang lebih baik, enam bulan kemudian dilakukan usulan ke DIKTI untuk melakukan Merger kedua Sekolah Tinggi diatas menjadi Universitas.

Pada hari Selasa, tgl. 8 Agustus 2000 keluarlah SK MENDIKNAS no. 126/D/O/2000 atas Universitas Komputer Indonesia yang disingkat dengan nama UNIKOM.Pada SK tersebut sekaligus diijinkan dibukanya 11 program studi baru: Teknik Komputer S1, Manajemen Informatika S1, Teknik Industri S1, Teknik Arsitektur S1, Perencanaan Wilayah dan Kota S1, Ilmu Hukum S1, Ilmu Komunikasi S1, Ilmu Pemerintahan S1, Desain Interior D3, Desain Komunikasi Visual S1 dan Desain Komunikasi Visual D3.

Sejak berdirinya pada tahun 2000, setiap tahunnya UNIKOM menerima ± 2.000 mahasiswa baru. Terakhir pada tahun 2009 yang lalu diterima sebanyak 3.108 mahasiswa baru. Hingga tahun akademik 2009/2010 terdapat 6 Fakultas dan 23 Program Studi di UNIKOM dengan jumlah mahasiswa sebanyak 15.000 orang yang berasal dari berbagai pelosok tanah air dan dari luar negeri yang sedang menempuh pendidikan di UNIKOM.

2.1.2 Visi, Misi, dan Tujuan Unikom

Berikut penjabaran dari visi, misi, dan tujuan Universitas Komputer Indonesia adalah:

1) Visi

Menjadi Universitas terdepan dibidang Teknologi Informasi & Komputer, berwawasan Global dan menjadi pusat Unggulan dibidang ilmu pengetahuan Teknologi dan seni yang mendukung pembangunan nasional serta berorientasi pada kepentingan masyarakat, bangsa dan Negara .

2) Misi

Menyelenggarakan Pendidikan tinggi kearah masyarakat Industri maju dengan sistem pendidikan yang kondusif, tenaga pengajar berkualitas dan program-program studi berbasis pada teknologi informasi & komputer dengan mengoptimalkan sumber daya yang ada, kualitas dan manajemen mutu berdasarkan prinsip *Quality Is Our Tradition*.

3) Tujuan

Menghasilkan Ilmuwan dan berpikiran tinggi maju dibidangnya masing-masing, mahir menggunakan teknologi informasi & komputer dalam bekerja serta beriman dan bertakwa kepada Tuhan Yang Maha Esa.

2.1.3 Logo Unikom

Logo Unikom (Gambar 2.1) memiliki makna dari setiap bagiannya, yaitu bingkai segi lima, lingkaran dalam segi lima tempat tulisan berwarna kuning, bulatan dalam berwarna biru, komputer, stasiun *relay*, satelit, cakrawala, dan buku.



Gambar 2.1 Logo Unikom

Adapun penjabaran makna dari setiap bagiannya adalah sebagai berikut:

a. Bingkai Segi Lima

Melambangkan UNIKOM berlandaskan falsafah negara yakni Pancasila dan Undang-Undang Dasar 1945.

b. Lingkaran dalam Segi Lima Tempat Tulisan Berwarna Kuning

Melambangkan motto UNIKOM menuju kejayaan yakni *Quality Is Our Tradition*.

c. Bulatan dalam Berwarna Biru

Melambangkan UNIKOM bertujuan menghasilkan ilmuwan unggul dan berpikiran maju yang Bertaqwa kepada Tuhan Yang Maha Esa.

d. Komputer

Melambangkan ciri utama UNIKOM yang memberikan pendidikan Teknologi Informasi dan Komputasi pada seluruh Jurusan yang ada dilingkungan Universitas Komputer Indonesia, menjadi Universitas Terdepan dibidang

Teknologi Informasi dan Komputer serta sebagai Universitas komputer pertama di Indonesia.

e. Stasiun Relay

Melambangkan UNIKOM menyelenggarakan Pendidikan Tinggi kearah masyarakat industri maju dengan sistem pendidikan yang kondusif dan tenaga pengajar berkualitas untuk menghasilkan lulusan-lulusan terbaik.

f. Satelit

Melambangkan UNIKOM berwawasan Global dan menjadi pusat unggulan dibidang IPTEK & seni yang mendukung Pembangunan Nasional serta berorientasi pada kepentingan masyarakat, bangsa dan negara.

g. Cakrawala

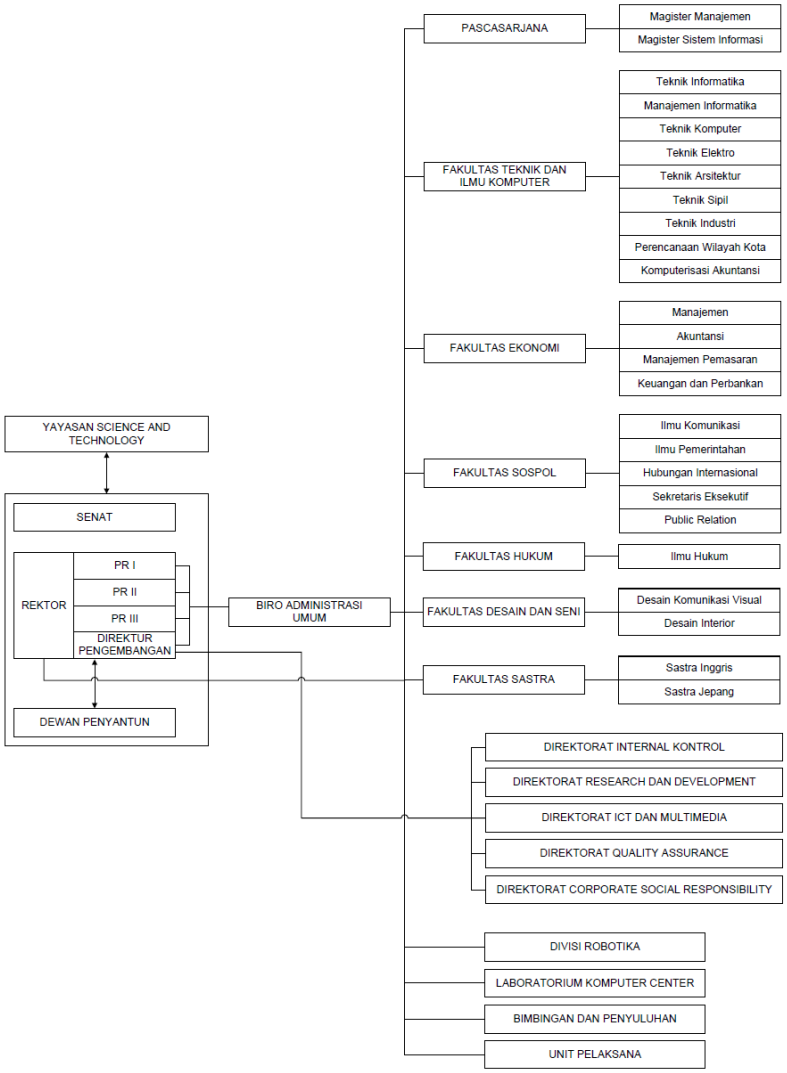
Melambangkan indahnya menggapai Cita-cita dan mengejar ilmu setinggi Langit.

h. Buku

Melambangkan sumber ilmu yang tiada habis-habisnya.

2.1.4 Struktur Organisasi Unikom

Unikom terdiri dari bagian atau divisi yang memiliki peranannya masing-masing.



Gambar 2.2 Struktur Organisasi UNIKOM

Keterangan :

PR I : Pembantu Rektor Bidang Akademik

PR II : Pembantu Rektor Bidang Keuangan, Administrasi, dan Kepegawaian

PR III : Pembantu Rektor Bidang Kemahasiswaan dan Kerjasama

2.2 Sistem

Pada dasarnya kata sistem berasal dari bahasa Yunani "*Systema*" yang berarti kesatuan yaitu kesuluruhan dari bagian-bagian yang mempunyai hubungan satu sama lain. Sistem adalah suatu kumpulan dari elemen-elemen baik berbentuk fisik maupun bukan fisik yang menunjukkan suatu hubungan diantaranya dan berinteraksi bersama-sama menuju suatu tujuan. Suatu sistem dapat terdiri dari beberapa subsistem yang saling berhubungan membentuk suatu kesatuan sehingga tujuan dari sistem dapat tercapai.

Berikut pengertian sistem menurut beberapa ahli:

1. Stephen A.Moscove dan Mark G.Simkin

   Suatu sistem adalah suatu kesatuan yang terdiri dari interaksi sub sistem yang berusaha untuk mencapai tujuan yang sama.

2. MJ. Alexander

   Suatu sistem adalah suatu group dari elemen-elemen baik berbentuk fisik maupun yang menunjukkan suatu kumpulan saling berhubungan diantaranya dan berinteraksi bersama-sama menuju satu atau lebih tujuan akhir dari sistem.

3. Richard F. Newschell

   Suatu sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan dikembangkan sesuai dengan skema yang berinteraksi untuk melaksanakan suatu kegiatan utama dalam bisnis.

Ciri – ciri dari sistem adalah sebagai berikut :

1. Adanya tujuan yang jelas.

2. Mempunyai strukutur yang jelas.
3. Terdiri dari satu kesatuan usaha dari bagian-bagian yang saling bergantung dan berinteraksi satu sama lain.

2.2.1 Karakteristik Sistem

Karakteristik sistem adalah sistem yang mempunyai komponen sistem, batasan sistem, linkungan luar sistem, penghubung, input, output, pengolah sistem dan sasaran sistem.

1. Komponen Sistem (*Components*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi yaitu saling bekerjasama membentuk satu kesatuan.Suatu sistem dapat mempunyai sistem yang lebih besar yang disebut Supra Sistem.

2. Batasan Sistem (*Boundary*)

   Batasan sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau lingkungan luarnya.Dimana suatu sistem dipandang sebagai satu kesatuan yang menunjukkan ruang lingkup dari sistem tersebut.

3. Lingkungan Luar Sistem (*Environment*)

   Lingkungan luar sistem adalah lingkungan diluar batas dari sistem yang mempengaruhi operasi sistem yang dapat bersifat menguntungkan atau merugikan.

4. Penghubung Sistem (*Interface*)

   Penghubung sistem merupakan penghubung antara satu subsistem dengan subsistem lainnya yang memungkinkan adannya sumber daya mengalir dari satu subsistem dengan subsistem yang lainnya.

5. Masukkan Sistem (*Input*)

   Masukkan sistem adalah energi yang dimasukkan ke dalam sistem yang dapat berupa masukkan perawatan (*Maintenance*) dan masukkan sinyal (*Signal Input*).

6. Keluaran Sistem (*Ouput*)

   Keluaran sistem adalah hasil dari energy yang diolah dan diklasifikasikan menjadi keluaran yang berguna dari sisa pembuangan. Keluaaran dapar berupa masukan dari subsistem yang lain atau kepada supra sistem.

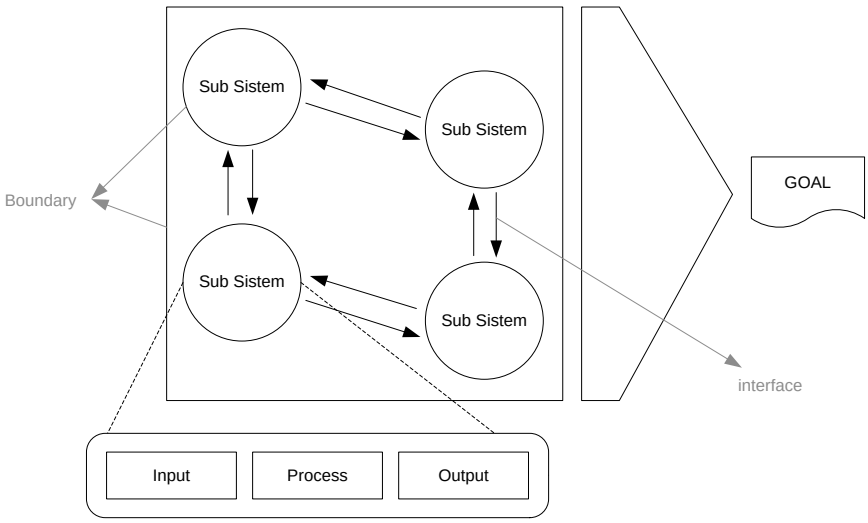
7. Pengolah Sistem (*Process*)

   Pengolah sistem akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem (*Goal*)

   Sasaran sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil jika mengenai sasaran atau tujuan.

   Berikut adalah diagram yang menggambarkan lingkungan sebuah sistem:



Gambar 2.3 Lingkungan Sistem

2.2.2 Data dan Informasi

Data adalah deskripsi tentang benda, kejadian, aktivitas, dan transaksi yang tidak mempunyai makna atau tidak berpengaruh secara langusng kepada pemakai. Data dapat berupa nilai yang terformat, teks, citra, *audio*, dan *video*. Sedangkan informasi merupakan hasil proses data-data yang beragam yang telah dibentuk sedemikian rupa sehingga sesuai dengan keinginan pengguna dan telah mengalami proses yang telah tersusun dengan baik dan benar sesuai kriteria yang diharapkan.

Sumber dari informasi adalah data. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata, tetapi data merupakan bentuk yang masih mentah dan belum dapat memberikan arti banyak bagi pemakai, sehingga perlu diolah lebih lanjut untuk menghasilkan informasi yang dibutuhkan. Dengan kata lain informasi adalah hasil dari pengolahan data.

Ciri-ciri informasi adalah sebagai berikut :

1. Data yang telah diolah.
2. Menjadi suatu bentuk yang lebih berguna dan berarti bagi yang menerima.
3. Menggambarkan suatu kejadian dan kesatuan nyata.
4. Digunakan untuk mengambil keputusan.

2.2.3 Kualitas Informasi

Istilah kualitas informasi dipakai untuk menyatakan informasi yang baik. Kualitas informasi dapat diukur dari tiga hal yaitu :

1. Akurat (*accurate*)

   Informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan serta jelas dalam menjelaskan maksudnya sehingga hasil dari informasi dapat dipergunakan sebagai acuan bagi pengguna informasi untuk mengambil keputusan..

2. Tepat Waktu (*Time Lines*)

   Informasi yang diterima oleh penerima tidak boleh terlambat. Suatu informasi harus sesuai dengan keadaan saat itu. Keterlambatan suatu informasi dapat berakibat fatal bagi suatu organisasi atau penggunanya. Hal ini dikarenakan informasi merupakan landasan dalam pengambilan keputusan.

3. Relevan (*relevance*)

   Informasi harus memiliki manfaat bagi penggunanya. Relevansi informasi untuk setiap orang berbeda-beda tergantung pengunannya.

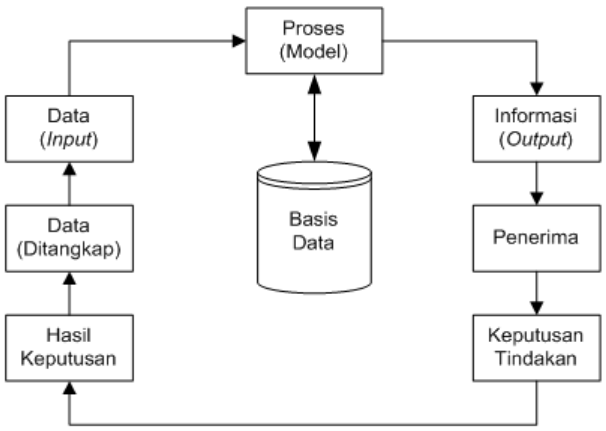
2.2.4 Nilai Informasi

Nilai Informasi (value of information) ditentukan oleh dua hal yaitu manfaat dan biaya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efekif dibandingkan dengan biaya mendapatkannya.

2.2.5 Siklus informasi

Siklus informasi menurut John Burch adalah gambaran secara umum mengenai proses terhadap data sehingga menjadi informasi yang bermanfaat bagi pengguna. Informasi yang menghasilkan informasi berikutnya. Demikian seterusnya proses pengolahan data menjadi informasi.

Data ditangkap sebagai *input*, diproses melalui suatu model membentuk *output* berupa informasi. Pemakai kemudian menerima informasi tersebut sebagai landasan untuk membuat suatu keputusan dan melakukan tindakan operasional yang akan membuat sejumlah data baru. Data baru tersebut selanjutnya menjadi input pada proses berikutnya, begitu seterusnya sehingga membentuk suatu siklus informasi (*information cycle*) atau disebut pula siklus pengolahan data (*processing cycle*).



Gambar 2.4 Siklus Informasi

2.2.6 Sistem Informasi

Sistem informasi adalah suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai suatu tujuan yaituvmenyajikan informasi. Pengertian lain dari sistem informasi adalah sekumpulan prosedur organisasi yang pada saat dilaksanakan dan memberikan informasi bagi pengambil keputusan dan/atau untuk mengendalikan informasi.

Sistem informasi merupakan suatu cara tertentu untuk menyediakan informasi yang dibutuhkan oleh organisasi untuk beroperasi dengan cara yang sukses dan menguntungkan. Dalam suatu organisasi,sistem informasi dapat dikatakan sebagai suatu sistem yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan informasi yang diperlukan untuk pengambilan keputusan.

2.2.7 Komponen Sistem Informasi

Sistem informasi terdiri dari komponen-komponen yang disebut dengan istilah blok bangunan (*building blok*) yaitu:

1. Perangkat Keras (*Hardware*), Mencakup piranti-piranti fisik yang terdapat dalam komputer seperti printer dan printer.
2. Perangkat Lunak (*Software*), yaitu sekumpulan instruksi yang memungkinkan perangkat keras dapat memproses data.
3. Orang (*Brainware*), yaitu pihak yang bertanggung jawab dalam pengembangan sistem informasi, pemrosesan, dan penggunaan keluaran sistem informasi.

4. Prosedur, yaitu aturan yang digunakan untuk mewujudkan pemrosesan data dan pembangkitan keluaran yang dikehendaki.
5. Jaringan Komputer, yaitu sistem penghubung yang memungkinkan sumber daya dapat dipakai secara bersama.
6. Basis Data (*Database*), yaitu sekumpulan tabel yang saling berelasi dengan penyimpanan data.

2.2.8 Tujuan Sistem Informasi

Sistem Informasi memiliki beberapa tujuan, yaitu:

1. Integrasi sistem, seperti:
    a. Menghubungkan sistem individu/kelompok
    b. Pengkolektifan data dan penyambungan secara otomatis
    c. Peningkatan koordinasi dan pencapaian sinergi
2. Efisiensi pengelolaan, seperti:
    a. Penggunaan basis data dalam upaya kesamaan pengadministrasian data
    b. Pengelolaan data berkaitan dengan karakteristik Informasi
    c. Penggunaan dan pengambilan Informasi
3. Dukungan keputusan untuk manajemen, seperti:
    a. Melengkapi Informasi guna kebutuhan proses pengambilan kebutuhan
    b. Akuisisi Informasi eksternal melalui jaringan komunikasi
    c. Ekstraksi dari Informasi internal yang terpadu.

2.2.9 Manfaat Sistem Informasi

Sistem Informasi memiliki beberapa manfaat, yaitu:

1. Menghemat tenaga kerja
2. Peningkatan efisiensi
3. Mempercepat proses
4. Perbaikan dokumentasi
5. Pencapaian standar
6. Perbaikan keputusan

2.3 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak (*software engineering*) merupakan pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomi, dipercaya dan bekerja secara efisien menggunakan mesin. Rekayasa perangkat lunak lebih fokus pada praktik pengembangan perangkat lunak dan mengirimkan perangkat lunak yang bermanfaat bagi pengguna. Rekayasa perangkat lunak fokus pada bagaimana membangun perangkat lunak yang memenuhi kriteria berikut:

1. Dapat terus dipelihara setelah perangkat lunak selesai dibuat seiiring berkembangnya teknologi dan lingkungan.
2. Dapat diandalkan dengan proses bisnis yang dijalankan dan perubahan yang terjadi.
3. Efisien dari segi sumber daya dan penggunaan.
4. Kemampuan untuk dipakai sesuai dengan kebutuhan.

Pekerjaan yang terkait dengan rekayasa perangkat lunak dapat dikategorikan menjadi tiga buah kategori umum tanpa melihat area dari aplikasi, ukuran proyek atau kompleksitas perangkat lunak yang akan dibangun. Yaitu:

a. Fase Pendefinisian (*Definition Phase*)

Fase pendefinisian fokus pada "*what*" yang artinya harus mencari tahu atau mengidentifikasi informasi apa yang harus diproses, seperti apa fungsi dan performansi yang diinginkan, seperti apa prilaku sistem yang diinginkan, apa kriteria validasi yang dibutuhkan untuk mendefinisikan sistem.

b. Fase Pengembangan (*Development Phase*)

Fase pengembangan fokus pada "*how*" yang artinya selama tahap pengembangan perangkat lunak seorang perekayasa perangkat lunak (software engineer) berusaha untuk mendefinisikan bagaimana data diinstrukturkan dan bagaimana fungsi-fungsi yang dibutuhkan diimplementasikan, bagaimana karakter antarmuka tampilan, bagaimana desanin ditranslasikan ke dalam bahasa pemrograman dan bagaimana pengujian dijalankan.

c. Fase Pendukung (*Support Phase*)

Fase pendukung fokus pada perubahan yang terasosiasi pada perbaikan kesalahan (*error*), adaptasi yang dibutuhkan pada lingkungan perangkat lunak yang terlibat dan perbaikan yang terjadi akibat perubahan kebutuhan pengguna. Fase pendukung terdiri dari empat tipe perubahan antara lain:

1. Koreksi (*Correction*), yaitu pemeliharaan dengan melakukan perbaikan terhadap kecacatan perangkat lunak.

2. Adaptasi (*Adaptation*), yaitu merupakan tahap untuk memodifikasi perangkat lunak guna mengakomodasi perubahan lingkungan luar dimana perangkat lunak dijalankan.
3. Perbaikan (*Enchancement*), pemeliharaan perfektif atau penyempurnaan melakukan eksekusi atau penambahan pada kebutuhan fungsional sebelumnya.
4. Pencegahan (*prevention*), pencegahan atau sering disebut juga dengan rekayasa ulang sistem (software reengineering) harus dikondisikan untuk mempu melayani kebutuhan yang diinginkan pemakainya.

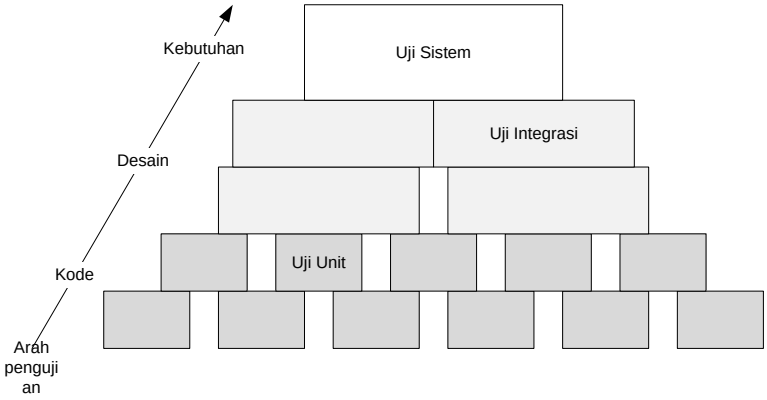
d. Pengujian

Pengujian adalah tahapan yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan dengan cara menemukan sebanyak-banyaknya kesalahan dalam selang waktu yang realistis.

Pengujian perangkat lunak sering dikaitkan dengan verifikasi (*verification*) dan validasi (*validation*). Verifikasi mengacu pada sekumpulan aktifitas yang berbeda yang menjamin bahwa perangkat lunak mengimplementasikan sebuah fungsi sfesifik dengan benar. Sedangkan mengacu pada sekumpulan aktifitas yang berbeda yang menjamin bahwa perangkat lunak yang dibangun dapat ditelusuri sesuai dengan kebutuhan pengguna.

1. Pengujian Verifikasi (*Verification*)

Pengujian verifikasi dilakukan mulai dari lingkup yang kecil naik ke lingkup yang besar seperti pada gambar berikut ini:



Gambar 2.5 Hirarki Pengujian Sistem

Pengujian diawali dari pengujian unit. Unit disini bisa berupa kumpulan kelas dan *package*. Setelah unit-unit selesai diuji maka dilakukan pengujian integrasi. Pengujian integrasi sebaiknya dilakukan secara bertahap, tidak dilakukan secara satu tahap langsung di akhir untuk menghindari kesulitan penelusuran jika terjadi kesalahan. Pengujian integrasi lebih kepada pengujian penggabungan dari dua atau lebih unit yang terdapat pada perangkat lunak. Setelah pengujian integrasi selanjutnya dilakukan pengujian sistem dimana unit-unit proses yang sudah diintegrasikan diuji dengan antarmuka yang sudah dibuat sehingga pengujian ini dimaksudkan untuk menguji sistem perangkat lunak secara keseluruhan.

2. Pengujian Validasi (*Validation*)

Pengujian validasi memiliki beberapa pendekatan, diantaranya sebagai berikut:

1. *Black-Box Testing* (Pengujian Kotak Hitam)

Pengujian kotak hitam adalah pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan

untuk mengetahui apakah fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak. Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah.

2. White-Box Testing (Pengujian Kotak Putih)

Pengujian kotak putih adalah pengujian perangkat lunak dari segi desain dan kode program. Pengujian kotak putih dilakukan dengan memeriksa lojik dari kode program. Pembuatan kasus uji dapat mengikuti standar pengujian pemrograman yang seharusnya.

2.4 Metodologi Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek dan pengujian.

2.4.1 Analisis Berorientasi Objek

Analisis berorientasi objek atau Object Oriented Analysis (OOA) adalah tahapan untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek, apakah kebutuhan yang ada dapat diimplementasikan menjadi sistem berorientasi objek.

Metode OOA saat ini yang paling banyak digunakan adalah dengan menggunakan UML (*Unifield Modeling Language*) yang akan dijelaskan pada sub bab UML.

2.4.2 Perancangan Berorientasi Objek

Perancangan berorientasi objek atau *Object Oriented Design* (OOD) adalah tahapan perantara untuk memetakan spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek ke perancangan permodelan agar lebih mudah diimplementasikan dengan pemrograman berorientasi objek.

Permodelan berorientasi objek biasanya dituangkan dalam dokumentasi perangkat lunak dengan menggunakan perangkat permodelan berorientasi objek, diantaranya adalah UML. Pada tahap ini biasanya kendala dan permasalahan pembangunan sistem dapat dikenali.

OOA dan OOD dalam proses yang berulang-ulang sering memiliki batasan yang samar sehingga kedua tahapan ini sering juga disebut OOAD (*Object Oriented Analysis dan Design*).

2.4.3 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek atau *Object Oriented Programming* (OOP) adalah sebuah cara membangun perangkat lunak dengan menggunakan beberapa objek yang akan saling berinteraksi satu sama lain dengan menggunakan bahasa pemrograman berorientasi objek seperti Java, C#, C++, Smalltalk, Eiffel dan PHP.

2.4.4 Konsep Dasar Berorientasi Objek

Berikut adalah beberapa konsep dasar tentang metodologi berorientasi objek:

1. Kelas (*Class*), yaitu kumpulan objek-objek dengan karakteristik yang sama.
2. Objek, yaitu abstraksi atau sesuatu yang mewakili dunia nyata seperti benda, manusia, tempat, kejadian dan hal-hal lain yang bersifat abstrak.
3. Metode, yaitu operasi pada sebuah kelas. Hampir sama dengan fungsi/prosedur pada metodologi struktural.
4. Atribut, yaitu variabel global yang dimiliki oleh sebuah kelas.
5. Abstraksi, yaitu prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.
6. Enkapsulasi, yaitu pembungkusan atribut data dan layanan yang dipunyai suatu objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.
7. Pewarisan, yaitu mekanisme yang memungkinkan suatu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.
8. Antarmuka (*interface*), sangat mirip dengan kelas tetapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi.
9. *Reusability*, yaitu pemanfaatan kembali objek yang sudah didefinisikan untuk suatu pemanfaatan pada permasalahan lainnya yang melibatkan objek tersebut.
10. Generalisasi dan spesialisasi, yaitu hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.
11. Komunikasi antarobjek, dilakukan lewat pesan (*message*) yang dikirim dari satu objek ke objek yang lainnya.

12. Polimorfisme, yaitu kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga ini akan menghemat baris program.
13. *Package*, yaitu sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokan kelas-kelas.

2.4.5 Keuntungan Metodologi Berorientasi Objek

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

1. Meningkatkan produktivitas, kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan pengembangan, sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan saat pengkodean.
3. Kemudahan pemeliharaan, dengan menggunakan model objek pola-pola yang cenderung tetap akan stabil dapat dipisahkan dengan pola-pola yang mungkin sering berubah-ubah.
4. Konsistensi, sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.
5. Meningkatkan kualitas perangkat lunak, pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya sehingga perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.5 Basis Data

Basis data terdiri atas dua kata yaitu basis dan data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, peserta, pembeli, pelanggan), barang, hewan, peristiwa konsep, keadaan dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya[13].

Basis data dapat didefinisikan dari sejumlah sudut pandang[3], seperti:

1. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
3. Kumpulan file atau tabel atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronik.

2.5.1 Basis Data Relasional

Basis data relational adalah basis data yang dilihat oleh pemakai sebagai sekumpulan tabel dua dimensi yang saling berelasi dan ternormalisasi dengan derajat yang beragam untuk mengambarkan suatu berkas data. Model ini menunjukan cara mengorganisasikan data secara fisik dalam memori sekunder, yang akan berdampak pada keseluruhan data yang terkait dalam sistem yang dibangun.

2.5.2 Normalisasi

Normalisasi merupakan tahapan perancangan dalam membangun basis data relasional yang tidak secara langsung berkaitan dengan model data, tetapi dengan menerapkan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal. Adapun langkah-langkah untuk menghasilkan struktur tabel yang normal, yaitu :

1. Bentuk *Unnormallized*

   Bentuk *Unnormallized* adalah bentuk tabel dengan mencantumkan semua *field* data yang ada.

2. Bentuk Normal Pertama atau 1 NF (*First Normal Form*)

   Bentuk normal pertama terpenuhi jika sebuah tabel tidak memiliki atribut bernilai banyak (*Multivalued Atribute*) atau lebih dari satu atribut dengan domain nilai yang sama.

3. Bentuk Normal Kedua atau 2 NF (*Second Normal Form*)

   Bentuk normal kedua terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam *key* primer memiliki ketergantungan fungsional pada *key* primer secara utuh.

4. Bentuk Normal Ketiga atau 3 NF (*Third Normal Form*)

   Bentuk normal ketiga terpenuhi jika dan hanya jika semua atribut bukan kunci. memiliki dependensi transitif terhadap kunci primer.

5. Boyce-Codd *Normal Form* (BCNF)

Suatu tabel dikatakan berada pada BCNF jika dan hanya jika semua ketergantungan fungsional dengan notasi X → Y, maka X harus merupakan *superkey* pada tabel tersebut.

6. Bentuk Normal Keempat atau 4 NF (*Fourth Normal Form*)

Suatu tabel dikatakan berada pada normal keempat jila tidak mengandung dua atribut atau lebih yang bernilai banyak.

7. Bentuk Normal Kelima atau 5 NF (*Fifth Normal Form*)

Bentuk normal kelima berkaitan dengan ketergantungan relasi antar tabel (*Join Dependency*).

2.5.3 Entity-Relationship (ER)

Model data entity-relationship (E-R) merupakan model data yang didasarkan pada perpsepsi bahwa dunia nyata merupakan sekumpulan dari jumlah objek (entitas) dasar dan relasi antar objek-objek data tersebut, dimana model ini menggunakan diagram hubungan antara entitas dengan entitas lain atau *relationship*.

Tipe *relationship* tergantung jumlah maksimum kardinalitas setiap entitas. Ada 4 tipe *relationship*, yaitu:

1. Satu-ke-satu (1:1)

   Tingkat hubungann satu-ke-satu, dinyatakan dengan satu kejadian pada entitas pertama, hanya mempunyai satu hubungan dengan satu kejadian pada entitas yang kedua dan sebaliknya.

2. Satu-ke-banyak (1:N)

   Tingkat hubungan satu-ke-banyak adalah untuk satu kejadian pada entitas yang pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas yang kedua.

3. Banyak-ke-satu (N:1)

   Untuk banyak kejadian pada entitas yang pertama hanya dapat mempunyai satu hubungan dengan kejadian pada entitas yang kedua.

4. Banyak-ke-banyak (N:N) atau (M:N)

   Tingkat hubungan banyak ke banyak terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan dengan kejadian pada entitas lainnya, baik dilihat dari sisi entitas yang pertama maupun dilihat dari sisi yang kedua.

2.5.4 Database Management System (DBMS)

Database Management System (DBMS) adalah perangkat lunak sistem yang dirancang untuk membuat, memelihara, mengontrol dan mengakses basis data dengan cara yang praktis dan efisien[13]. DBMS dapat digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda.

Pada umumnya DBMS menyediakan fitur-fitur sebagai berikut:

1. Independensi data program, karena basis data ditangani oleh DBMS, program dapat ditulis ulang sehingga tidak bergantung pada struktur data dalam basis data.

2. Kemanan, dimaksudkan untuk mencegah pengaksesan data oleh orang yang tidak berwenang.
3. Integritas, dimaksudkan untuk menjaga agar data selalu dalam keadaan yang valid dan konsisten.
4. Pemulihan (*recovery*), DBMS menyediakan mekanisme untuk mengembalikan basis data ke keadaan semula yang konsisten sekiranya terjadi gangguan perangkat keras atau perangkat lunak.
5. Katalog sistem
6. Perangkat produktivitas, DBMS menyediakan sejumlah perangkat produktifitas dan kemudahan bagi pemakai seperti pembangkit query dan pembangkit laporan.

2.5.5 Bahasa Basis Data

Sistem basis data menyediakan bahasa untuk mendefinisikan skema dan memanipulasi basis data untuk mengekspresikan *query database*. Dalam prakteknya, bahasa untuk mendefinisikan ddan manipulasi database bukan merupakan bahasa yang terpisah, melainkan bagian dari bahasa database seperti kebanyakan penggunaan sql.

a. Data Definition Language (DDL)

DDL adalah perintah-perintah yang biasa digunakan untuk mendefinisikan skema basis data dan juga sub skema. Hasil dari kompilasi pernyataan-pernyataan DDL disimpan dalam berkas spesial yang disebut katalog sistem. Katalog sistem ini memerlukan metadata, yaitu data yang menjelaskan objek-objek dalam basis data.

Pernyataan *sql* seperti *create table*, *drop table* merupakan contoh perintah yang termasuk dalam kategori DDL.

b. Data Manipulation Language (DML)

DML adalh perintah-perintah yang digunakan untuk melakukan hal seperti berikut:

1. Mengambil data pada basis data,
2. Menambah data pada basis data,
3. Mengubah data pada basis data,
4. Menghapus data pada basis data.

DML dapat dibagi menjadi dua kelompok, yaitu prosedural dan non-prosedural.

1. **DML prosedural** adalah perintah-perintah yang memungkinkan pemakai menentukan data apa saja yang diperlukan dan bagaimana cara mendapatkannya.
2. **DML non-prosedural** adalah perintah-perintah yang memungkinkan pemakai menentukan data apa saja yang diperlukan tanpa perlu menyebutkan cara mendapatkannya.

2.6 RFID (*Radio Frequency Identification*)

RFID (*Radio Frequency Identification*) adalah teknologi yang dapat mengidentifikasi objek dengan menggunakan transmisi gelombang radio.[8]

Teknologi RFID mampu mengidentifikasi beberapa objek secara simultan tanpa diperlukan kontak langsung (*contactless*). Teknologi ini dikembangkan sebagai pengganti atau penerus teknologi *barcode*. Implementasi RFID telah

banyak digunakan pada lingkungan manufaktur dan industri yang memerlukan akurasi dan kecepatan identifikasi objek dalam jumlah yang besar serta berada di area yang luas. RFID bekerja pada HF untuk aplikasi jarak dekat (*proximity*) dan bekerja pada UHF untuk aplikasi jarak jauh (*vicinity*).

Secara umum RFID mempunyai dua komponen, komponen pertama adalah benda elektronik yang didekatkan pada benda yang dikenali, komponen ini disebut *tag transponder*. Komponen kedua adalah alat yang dapat membaca *tag* (*reader*). Namun dua komponen ini saja tidak cukup untuk membuat sistem RFID berfungsi. Ada komponen lain yang diperlukan yaitu *software* atau aplikasi, fungsinya adalah sebagai elemen pengolah data hasil kerja dari kedua komponen RFID tadi.

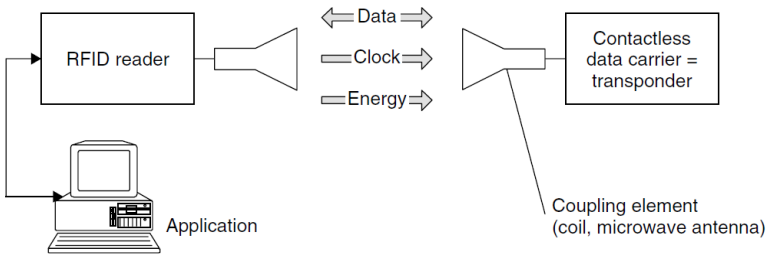
Kelebihan-kelebihan RFID adalah sebagai berikut :

1. Pengenalan dapat dilakukan tanpa kontak langsung dengan *hardware* (*contactless*),
2. Data yang terkandung dalam *tag* dapat ditulis ulang (*rewritable data*),
3. Transmisi data tidak harus tegak lurus dengan pembaca (*absence line of sight*),
4. Kapasitas data yang luas, mendukung banyak pembacaan data (*support for multiple tag reads*),
5. Fisik kokoh dan dapat melakukan tugas pintar (*smart task*).

2.6.1 Cara Kerja RFID

Berikut adalah gambaran singkat cara kerja sistem RFID : RFID *reader* selalu dalam kondisi siap untuk membaca kehadiran *transponder*. Ketika suatu

transponder langsung mengirimkan data yang dibawa ketika merespon kehadiran frekuensi radio dari RFID *reader*. RFID *reader* segera menerima data yang dikirimkan lalu melewatkan data itu ke aplikasi untuk dilakukan pengolahan. Lebih jelasnya perhatikan gambar berikut ini.



Gambar 2.6 Komponen Sistem RFID

2.6.2 Komponen RFID

1. *Tag*

Seperti yang telah dijelaskan sebelumnya, tag (*trasnponder*) adalah salah satu komponen utama dalam sistem RFID yang berfungsi mengirimkan data (*transmit*) yang dibawa ketika merespon kehadiran frekuensi radio dari *reader*.



Gambar 2.7 Bentuk Tag RFID [4]

Tag RFID dibentuk menjadi beragam bentuk fisik yang disesuaikan dengan kebutuhan diantaranya adalah bentuk gantungan kunci, kancing, kartu, stiker dan sebagainya.

2. *Reader*

Prinsip kerja RFID reader serupa dengan *tranceiver* radio, yaitu memancarkan dan menerima. Reader ini dalam kondisi siaga akan memancarkan gelombang elektromagnetik sesuai dengan daya jangkaunya. Ketika ada tag memasuki area jangkauannnya, tag akan mendapat daya dari gelombang elektromagnet *reader*.

Dari daya yang diperoleh, tag memancarkan data yang dibawa. Data pancaran tersebut akan diterima oleh reader. Selanjutnya data yang diterima tadi akan diteruskan pada aplikasi untuk diolah sesuai dengan rancangan sistem.

Dalam fungsinya, RFID reader dituntut untuk dapat melakukan dua tugas, yaitu berkomunikasi melalui gelombang radio dan membaca data yang dibawa tag kemudian diteruskan ke aplikasi.

2.7 UML

Pemodelan (*modeling*) adalah gambaran dari realita yang dituangkan dalam bentuk pemetaan dengan aturan tertentu. Permodelan dalam dunia pembangunan perangkat lunak digunakan dalam proses merancang perangkat lunak sebelum melakukan pengkodean (*coding*) tujuannya untuk mempermudah langkah berikutnya dari pembangunan perangkat lunak sehingga lebih terencana dan semua kebutuhan pengguna dapat terpenuhi dengan lengkap dan tepat, termasuk faktor-faktor seperti *scalability*, *robustness*, *security*, dan sebagainya.

Kesuksesan suatu pemodelan perangkat lunak ditentukan oleh tiga unsur, yang kemudian terkenal dengan sebuan segitiga sukses (*the triangle for success*). Ketiga unsur tersebut adalah metode pemodelan (*notation*), proses (*process*) dan *tool* yang digunakan. Memahami notasi pemodelan tanpa mengetahui cara pemakaian yang sebenarnya (proses) akan membuat sebuah proyek gagal. Dan pemahaman terhadap metode pemodelan dan proses disempurnakan dengan penggunaan *tool* yang tepat.

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan men-dokumentasikan sistem perangkat lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. UML kita dapat membuat model untuk semua jenis perangkat lunak, dimana aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi dan jaringan serta ditulis dalam bahasa pemrograman. Tetapi karena UML menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan perangkat lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural seperti C, Delphi atau VB.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram perangkat lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim

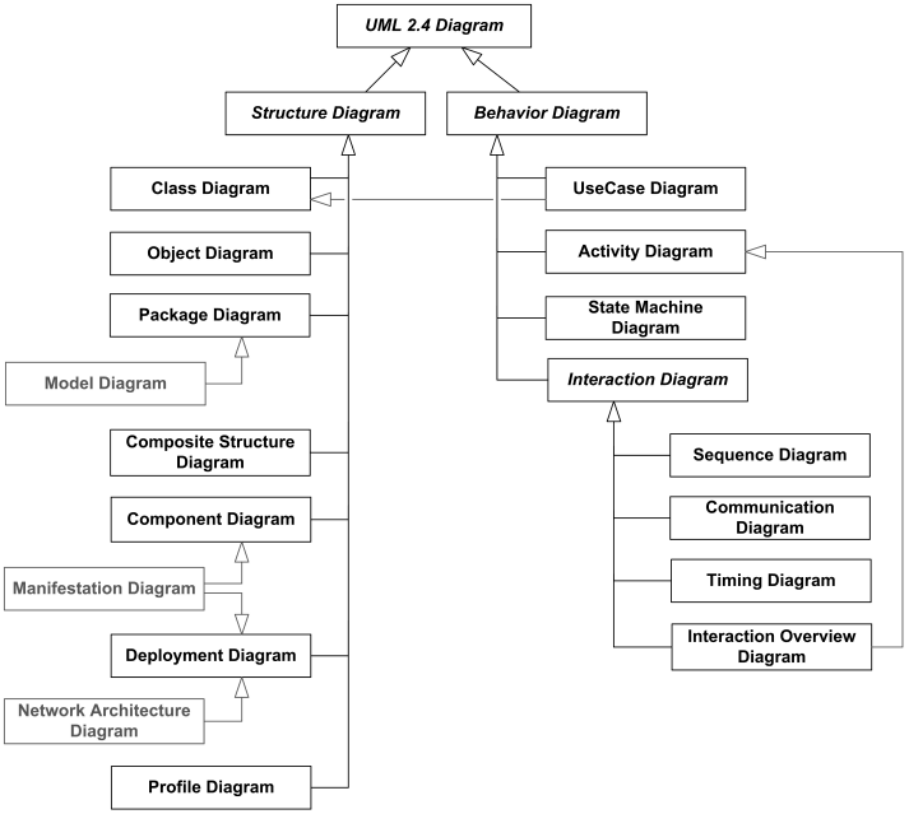
Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi perancangan berorientasi objek. Pada tahun 1995 dirilis draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG). Tahun 1997 UML versi 1.1 muncul, dan hingga saat ini telah ada versi 2.4.1 yang terdiri dari 4 macam spesifikasi, yaitu *Diagram Interchange Sfesification*, UML *Infratructure*, UML *Super Structure* dan *Object Constraint Language* (OCL). Seluruh spesifikasi tersebut dapat diakses di http://www.omg.org

2.7.1 Diagram UML

Pada UML versi 2.4 terdapat 2 kategori diagram sebagai berikut:

1. *Structure Diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behaviour Diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.



Gambar 2.8 Diagram UML 2.4

2.7.2 Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan perancangan berorientasi objek. *Class* menggambarkan keadaan (atribut) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda).

Sebuah Kelas memiliki struktur sebagai berikut:

1. Nama Kelas/Stereotype
2. Atribut, yaitu variabel-variabel yang dimiliki suatu kelas.
3. Metoda, yaitu fungsi-fungsi yang dimiliki oleh suatu kelas.

Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Diagram Class mempunyai 3 macam *relationalships* (hubungan), sebagai berikut :

1. *Association*

   Assosiation adalah hubungan antara bagian dari dua kelas. Terjadi *association* antara dua kelas jika salah satu bagian dari kelas mengetahui yang lainnya dalam melakukan suatu kegiatan. Di dalam diagram, sebuah *association* adalah penghubung yang menghubungkan dua kelas.

2. *Aggregation*

   Aggregation adalah *association* dimana salah satu kelasnya merupakan bagian dari suatu kumpulan. *Aggregation* memiliki titik pusat yang mencakup keseluruhan bagian.

3. *Generalization*

   Generalization Suatu hubungan turunan dengan mengasumsikan satu kelas merupakan suatu *Super Class* dari kelas yang lain. *Generalization* memiliki tingkatan yang berpusat pada *Super Class*.

2.7.3 Object Diagram

Object diagram menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada *object diagram* harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas harus terpakai objeknya.

2.7.4 Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen perangkat lunak, termasuk ketergantungan (*dependency*) di antaranya.

Komponen perangkat lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa class dan/atau package, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

2.7.5 Composite Structure Diagram

Composite structure diagram baru mulai ada pada UML 2.0. *Composite structure diagram* dapat digunakan untuk menggambarkan struktur dari bagian-bagian yang saling berhubung maupun mendeskripsikan struktur pada saat berjalan (*runtime*) dari *intance* yang saling terhubung. Contoh penggunaan diagram ini misalnya untuk menggambarkan deskripsi dari setiap bagian mesin yang saling terkait untuk menjalankan fungsi mesin tersebut, menjalankan aliran data router pada jaringan komputer, dan lain sebagainya.

2.7.6 Package Diagram

Untuk mengatur pengorganisasian diagram Class yang kompleks, dapat dilakukan pengelompokan kelas-kelas berupa package (paket). Package adalah kumpulan elemen-elemen logika UML. Gambar di bawah ini mengenai model bisnis dengan pengelompokan kelas-kelas dalam bentuk paket-paket.

2.7.7 Deployment Diagram

Deployment diagram menggambarkan detail bagaimana komponen di-deploy dalam infrastruktur sistem, di mana komponen akan terletak, bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisik sebuah node adalah server, workstation, atau perangkat keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar node (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

2.7.8 Use Case Diagram

Use case diagram merupakan permodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibangun. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibangun. *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkindan dapat dipahami. Ada dua hal utama pada *use case* yaitu aktor dan *use case.*

1. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem yang akan dibangun.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor

2.7.9 Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu, *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Seperti halnya *state*, standar UML menggunakan segi empat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

2.7.10 State Machine Diagram

State machine diagram menggambarkan transisi dan perubahan keadaan suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya *state machine diagram* menggambarkan *class* tertentu.

Dalam UML, *state* digambarkan berbentuk segi empat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

2.7.11 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa pesan yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah kejadian untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi metoda dari *class*. *Activation*

bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan simbol khusus untuk *objek boundary*, *controller* dan *persistent entity*.

2.7.12 Communication Diagram

Communication Diagram pada UML versi 2.x adalah penyederhanaan *Collaboration Diagram* pada UML versi 1.x. *Communication diagram* juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor *messages* dari level yang sama memiliki prefiks yang sama.

2.7.13 Timming Diagram

Timming diagram merupakan diagram yang fokus pada penggambaran terkait batasan waktu. *Timming diagram* digunakan untuk menggambarkan tingkah laku sistem dalam periode waktu tertentu. *Timming diagram* biasanya digunakan untuk mendeskripsikan operasi dari alat digital karena penggambaran secara visual akan lebih mudah dipahami daripada dengan kata-kata.

2.7.14 Interaction Overview Diagram

Interaction overview diagram mirip dengan activity diagram yang berfungsi untuk menggambarkansekumpulan urutan aktivitas. *Interaction overview diagram*

adalah bentuk aktivitas diagram yang setiap titik merepresentasikan diagram interaksi. Interaksi diagram dapat meliputi *sequence diagram, communication diagram, timming diagram* dan *interaction overview diagram*.

Hampir semua interaksi pada *Interaction overview diagram* sama dengan notasi pada *activity diagram*. Tambahan pada *interaction overview diagram* adalah *interaction occurence* dan *interaction element*.

1. *Interaction Occurence* atau kejadian interaksi adalah referensi yang digunakan untuk diagram interaksi yang ada
2. *Interaction Eelement* atau elemen interaksi mirip *interaction occurence* perbedaannya didalam interaction element menampilkan isi diagram yang direfrensikan secara langsung. Sedangkan *interaction occurence* hanya menampilkan nama diagram yang direfrensikan.

2.7.15 Profil Diagram

Profil diagram menggambarkan mekanisme ekstensi UML dengan mendefinisikan stereotype, tagged value dan constraint. Profil memungkinkan adaptasi dari model UML yang berbeda platform dan domain.

2.8 MySQL

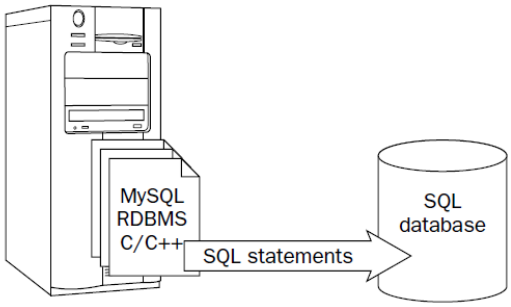
MySQL adalah sebuah produk *Relational Database Management System* (RDBMS). MySQL dapat mengakses data, memanipulasi data, melindungi data dari korupsi dan inkonsistensi serta mempertahankan metadata yang diperlukan untuk menentukan data yang disimpan dalam *database*. Perbedaan antara DBMS dengan RDBMS adalah bukan hanya dari mendukung atau tidaknya penyimpanan data dalam tabel, tetapi juga hubungan antartabel. MySQL menyediakan banyak

fitur yang mendukung sebuah lingkungan yang aman untuk menyimpan, memelihara, dan mengakses data. Berikut kelebihan-kelebihan yang ditemukan di MySQL:

1. Skalabilitas, MSQL dapat menangani *database* yang besar, yang telah dibuktikan implementasinya dalam organisasi seperti Yahoo, Google, Cisco, HP, NASA dan lain sebagainya.
2. Portabilitas, MySQL dapat berjalan pada berbagai macam sistem operasi termasuk Windows, Unix, Linux, Solaris dan Mac OS. Juga dapat berjalan pada arsitektur yang berbeda, mulai dari *low-end* PC sampai *high-end mainframe*.
3. Konektivitas, MySQL sepenuhnya mendukung jaringan dan dapat diakses dari mana saja di internet serta pengguna dapat mengakses *database* MySQL secara bersamaan. MySQL juga menyediakan berbabagai macam API (*Aplication Program interface*) untuk mendukung konektivitas aplikasi yang ditulis dalam bahasa C, C++, Perl, PHP, Java, Phyton, C# dan lain sebagainya.
4. Keamanan, MySQL mencakup seluruh keamanan yang kuat untuk mengontrol akses data dan juga mendukung *Secure Socket Layer* (SSL) *Protocol*
5. Kecepatan, MySQL dikembangkan dengan kecepatan.
6. Mudah digunakan, MySQL mudah untuk digunakan dan diimplementasikan.

7. *Open Source,* MySQLAB membuat kode MySQL tersedia untuk digunakan setiap orang. filosofi *open source* memungkinkan khalayak global untuk berpartisipasi dalam pengembangan

Sebuah DBMS/RDBMS tidak dapat lepas dari SQL (*Structured Query Language*). SQL merupakan sebuah bahasa yang digunakan untuk mengelola dan berinteraksi dengan data dalam *database* relasional. SQL adalah bahasa *database* yang paling universal digunakan, dan itu telah menjadi bahasa standar untuk manajemen *database*. SQL bekerja sama dengan sebuah RDBMS untuk mendefinisikan struktur dari *database*, menyimpan data di *database* tersebut, memanipulasi data, mengambil data, mengontrol akses ke data, dan menjamin integritas data.



Gambar 2.9 Interaksi SQL dengan MySQL RDBMS

Beberapa perintah dasar SQL yang sering dipergunakan pada MySQL:

1. *Create Database*, perintah yang digunakan untuk membuat database baru.

   Sintaks : CREATE DATABASE DATABASE_NAME
2. *Drop Database,* perintah yang digunakan untuk menghapus database.

   Sintaks : DROP TABEL TABEL_NAME
3. *Create Tabel,* perintah yang digunakan untuk membuat tabel baru.

Sintaks : Create Tabel tabel_name (create_definition)

4. *Describe*, perintah yang digunakan untuk mendeskripsikan tabel.

   Sintaks : Describe (Desc) tabel [colum]

5. *Alter Tabel*, perintah yang digunakan untuk memodifikasi tabel.

   Sintaks : Alter [Ignor] Tabel table_name

6. *Drop Tabel*, perintah yang digunakan untuk menghapus tabel.

   Sintaks : Drop Tabel tabel_name [tabel_name..]

7. *Delete*, perintah yang digunakan untuk menghapus record dri tabel.

   Sintaks : Delete From tabel_name Where Where_definiition

8. *Select*, perintah yang digunakan untuk query ke database.

   Sintaks : select * from tabel_name

MySQL mendukung beberapa API yang memungkinkan aplikasi yang ditulis dalam berbagai jenis bahasa pemrograman untuk berkomunikasi dengan database MySQL. Berikut beberapa API yang didukung oleh MySQL:

1. C API, adalah antarmuka pemrograman utama yang memungkinkan aplikasi C/C++ untuk menghubungkan ke MySQL. Sebagian besar aplikasi klien termasuk dalam distribusi MySQL yang ditulis dalam C dan bergantung pada API ini.
2. ODBC, MySQL mendukung Open Database Connectivity (ODBC) melalui MySQL Connector / ODBC. ODBC adalah konektivitas database standar yang memungkinkan berbagai jenis aplikasi untuk menghubungkan ke berbagai jenis database.

3. JDBC, MySQL mendukung Java Database Connectivity (JDBC) melalui MySQL Connector / JDBC. JDBC adalah konektivitas *database* standar yang memungkinkan Java untuk menghubungkan ke berbagai jenis database.
4. PHP API, yang sekarang termasuk dengan *preprocessor* PHP, memungkinkan *script* PHP pada halaman Web untuk berkomunikasi secara langsung dengan *database* MySQL. Koneksi *database* dan permintaan data (melalui pernyataan SQL) dikodekan langsung dalam *script* PHP.

2.9 .NET Framework dan Bahasa Pemrograman C#

2.9.1 NET Framework

Microsoft mengumumkan C# pada tahun 2000 bersamaan dengan pengumuman *platform* .NET Framework. .NET Framework adalah *platform* yang merupakan perwujudan teknologi modern. Framework .NET diciptakan untuk dapat memecahkan masalah yang banyak dihadapi dunia pemrograman secara lebih efisien. Metodologi perancangan berorientasi objek dan konsep software sebagai komponen tertanam dengan kuat di Framework.NET.

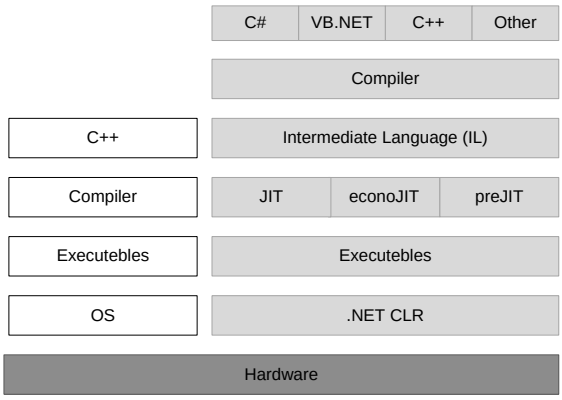
Dua bagian penting dari Framework .NET adalah *Common Language Runtime* (CLR) dan Class Library .NET.

1. *Common Language Runtime* (CLR)

CLR adalah bagian dari Framework .NET yang mengelola program-program .NET yang dijalankan. CLR inilah yang mengatur hal-hal seperti pengalokasian memori, pengecekan type, dan keamanan.

Dalam .NET, program tidak dikompilasi kedalam file executable (.exe) melainkan dikompilasi ke *Intermediate Language* (IL), oleh karena itu program-

program .NET dapat dijalankan di sistem operasi apa saja asalkan pada sistem operasi yang bersangkutan terpasang .NET Framework. Program .NET dengan CLR bisa dianalogikan seperti program Java dengan JVM. Cara kerjanya adalah ketika program .NET dijalankan, CLR akan memanggil JIT (*Just In Time Coompiler*) untuk mengompile IL dalam program tersebut agar menjadi kode mesin di sistem operasi yang bersangkutan. Keseluruhan program tersebut tidak dicompile oleh JIT sekaligus, melainkan sesuai kebutuhan. Hal inilah yang membuat program-program .NET dapat berjalan dengan kecepatan tinggi karena program tersebut dijalankan secara *native* pada sistem operasi yang bersangkutan. IL juga merupakan salah satu faktor yang memungkinkan tercapainya *cross language interperability*, yaitu kemampuan sebuah program untuk berinteraksi dengan program yang dibuat menggunakan bahasa lain.



Gambar 2.10 Struktur CLR

2. Class Library .NET

Di dalam Framework .NET terdapat banyak *class library* mulai dari *class* yang berisi fungsi-fungsi matematika, *class* yang berhubungan dengan keamanan,

class untuk membuat program dengan GUI (*Graphic User Interface*) sampai pada class yang berhubungan dengan input/output.

Adapun kelebihan *platform* Framework.NET dibandingkan dengan *platform* lain adalah sebagai berikut:

1. *Platform Independence*, program-program yang dibuat untuk berjalan di atas Framework .NET dapat dijalankan di sistem operasi apapun, asalkan di sistem yang bersangkutan terpasang implementasi Framework.NET tidak terkecuali untuk sistem operasi *mobile*.
2. *Language Independence* dan *Cross Language Interoperability*, *platform* .NET tidak terkait dengan bahasa apapun. Contoh-contoh bahasa yang bisa dipakai untuk pemrograman .NET adalah C#, C++, VB.NET, J#, F# dan sebagainya.
3. Kecepatan, walaupun bersifat *portable* program-program .NET dapat berjalan dengan kecepatan tinggi, bahkan dapat lebih cepat dari program-program *unmanaged*.
4. Keamanan, karena semua program .NET berjalan di bawah pengawasan CLR, maka keamanan menjadi lebih terjamin.
5. Produktifitas Tinggi, konsep pemrograman berbasis objek dan kekayaan *class library* .NET yang tertanam kuat memungkinkan pembuatan dan pengembangan program lebih mudah.
6. *Support* untuk Jaringan Internet, Framework.NET dirancang dengan internet "*in mind*". Interaksi komponen yang terletak di dua komputer yang berbeda dapat dilakukan secara mudah.

7. .NET adalah *Platform* yang Terbuka, siapapun boleh membuat bahasa dan *compiler* yang menghasilkan program .NET, bahkan membuat implementasi .NET itu sendiri. Framework .NET juga banyak menggunakan teknologi-tekonologi yang sudah dipakai secara luas seperti XML, Unicode, HTTP, dan TCP/IP.

2.9.2 Bahasa Pemrograman C#

C# adalah bahasa pemrograman berbasis objek yang didukung oleh Microsoft .NET Framework. C# dibuat dengan tujuan untuk menyediakan bahasa pemrograman yang sederhana, modern, berorientasi objek, berpusat pada internet dan memiliki kinerja tinggi[1]. C# adalah salah satu dari banyak bahasa yang bisa dipakai untuk pemrograman .NET. Kelebihan utama bahasa ini adalah sintaksnya yang mirip C++, Java dan VB namun lebih sederhana dan mudah. Untuk mencapai kemudahan inilah konsep-konsep sulit pada C++, Java dan VB disederhanakan dengan mencakup semua dukungan untuk pemrograman terstruktur, pemrograman berbasis komponen dan pemrograman berorientasi objek.

C# dikembangkan oleh sebuah tim yang dipimpin oleh Anders Hejlsbreg dan Scott Wiltamuth yang sebelumnya dikenal karena menciptakan Turbo Pascal dan merancang Borland Delphi.

Jantung dari setiap pemrograman berorientasi objek adalah adanya dukungan untuk mendefinisikan dan bekerja dengan kelas. Kelas mendefinisikan tipe baru yang memungkinkan programmer untuk memperpanjang bahasa. C# mengandung keyword untuk mendeklarasikan kelas baru, metode dan properti serta sebagai

bahasa yang berorientasi objek C# juga dapat mengimplementasikan enkapsulasi, pewarisan dan polimorfisme.

C# sebagai bahasa pemrograman untuk Framework .NET memiliki ruang lingkup penggunaan yang sangat luas seperti pembuatan perangkat lunak dengan *user interface* Windows maupun *console* dan karena Framework .NET memberikan fasilitas untuk berinteraksi dengan kode yang *unmanaged*, *library* seperti DirectX dan OpenGL dapat digunakan di C#. C# juga dapat digunakan untuk pemrograman *web site*, *web service*, membangun perangkat lunak Zune, perangkat lunak permainan (desktop dan XBOX), perangkat lunak *mobile* dan lain sebagainya.

a. Struktur Dasar Program C#

Berikut adalah contoh program C# beserta keterangan struktur bahasanya:

```
using system;                                          → import

                                                       → komentar

//program yang akan menampilkan hello world            → kurung kurawal
namespace ProgramHelloWorld                            → fungsi Main()
{                                                      → deklarasi variabel
        class program                                  → statement
        {
                static void Main(string[] args)
                {
                        string kalimat;
                        kalimat = "Hello World";
```

```
            console.write(kalimat);
        }
    }
}
```

Komponen-komponen yang hampir selalu ada pada kode program c#:

1. Fungsi Main(), berisi sekumpulan perintah yang akan dikerjakan ketika menjalankan program.
2. Deklarasi Variabel, perintah agar komputer menyediakan variabel yang akan dipakai.
3. Statement/Perintah, digunakan untuk memerintahkan komputer agar melakukan sesuatu.
4. Keyword using, digunakan untuk menyertakan fungsi-fungsi dari suatu *library*.
5. Komentar, digunakan untuk catatan/tanda agar kode yang dibuat dapat dimengerti.
6. Kurung Kurawal, digunakan untuk memberitahu komputer bahwa perintah yang ada didalamnya merupakan satu-kesatuan.

b. Tipe Data dan Variabel

C# memiliki dua jenis data, yaitu:

1. Tipe data dasar, seperti: integer, string, decimal, float, byte dan sebagainya.
2. Tipe data reference seperti *object* dan *delegate*.

Tabel 2.1 Tipe Data pada C#

Tipe	.NET	Ukuran (byte)	Keterangan
Byte	Byte	1	Dapat berisi integer 0 – 255
Char	Char	2	Berisi karakter Unicode
Bool	Boolean	1	Hanya berisi *true* atau *false*
Sbyte	Sbyte	1	Memiliki range -128 – 127
Short	Int16	2	Memiliki range -32768 – 32767
ushort	UInt16	2	Memiliki range 0 – 6535
Int	Int32	4	Memiliki range 2,147,438648 – 2,147,483,647
Uint	UInt32	4	Memiliki range 0 – 2,94,967,295
Float	Single	4	Bilangan pecahan dengan kepresisian hingga 7 digit
double	Double	8	Bilangan pecahan dengan kepresisian hingga 15 - 16 digit
decimal	Decimal	12	Bilangan dengan kepresisian hingga 28 - 29 digit
Long	Int64	8	Memiliki range -9,223,372,036,854,775,808 – 9,223,372,036,854,775,807
Ulong	Int64	8	Memiliki range 0 – 18,446,744,073,709,551,615
String	String	Alfanumerik	Berisi karakter alfanumerik atau gabungan dari beberapa karakter

Variabel adalah lokasi penyimpan dengan tipe data. Variabel dapat memiliki data yang ditugaskan kepadanya dan dapat diubah secara pemrograman.

Berikut salah satu sintaks untuk mendeklarasikan sebuah variabel dalam C#:

<tipe data> <variabel>;

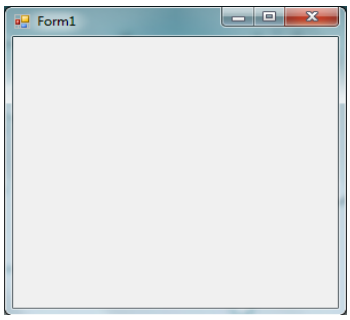
c. Membangun Aplikasi Windows

Sebelumnya perangkat lunak dibangun hanya berbasis *console* dengan antarmuka teks. Pada saat ini kebanyakan pengguna perangkat lunak menuntut

kehadiran perangkat lunak yang memiliki antarmuka pengguna grafis (GUI). GUI memungkinkan pengguna dapat berinteraksi lebih intim dengan perangkat lunak yang digunakannya. Namun di sisi lain, pengembangan perangkat lunak dengan GUI adalah hal yang kompleks.

Lingkungan .NET Framework menawarkan dukungan yang kuat untuk implementasi konsep *software reusability* dengan menyediakan tool lengkap untuk membangun Windows atau Aplikasi Web salahsatunya adalah Windows Form Framework. Windows Form menggunakan konsep formulir yang diambil dari lingkungan Visual Basic (VB) yang sukses dalam mendukung konsep pengembangan perangkat lunak RAD (*Rapid Application Development*) . C# adalah lingkungan pengembangan pertama yang mengadopsi alat RAD Visual Basic dengan karakteristik berorientasi objek dan memiliki kinerja tinggi seperti halnya keluaga bahasa C.

Berikut adalah gambar Windows Form pada C#:



Gambar 2.11 Windows Form

Visual Studio .NET menyediakan IDE (*Integrated Development Environment*) untuk bekerja dengan Windows Form. Yang akan dijelaskan pada sub bab berikutnya.

2.10 Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak yang dapat digunakan untuk melakukan pengembangan aplikasi, baik aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup kompiler, SDK, *Integrated Development Environment* (IDE), dan *Library*. Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, C#, VB.NET, J++ dan J#.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *unmanaged*/*native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed* code (dalam bentuk Microsoft Intermediate Language di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework).

Rilis terakhir Microsoft Visual Studio adalah Versi 2010 dan mendukung platform .NET Framework 4.0 dan versi terdahulunya. versi sebelumnya Visual Studio 2008 yang ditujukan untuk platform .NET Framework 3.5, Visual Studio 2005 ditujukan untuk platform .NET Framework 2.0 dan 3.0. Visual Studio 2003 ditujukan untuk .NET Framework 1.1, dan Visual Studio 2002 ditujukan untuk .NET Framework 1.0. Versi-versi tersebut di atas kini dikenal dengan sebutan Visual Studio .NET, karena membutuhkan Microsoft .NET Framework. Sementara itu, sebelum muncul Visual Studio .NET, terdapat Microsoft Visual Studio 6.0.

Beberapa keuntungan Visual Studio:

1. Dapat membuat aplikasi untuk beberapa platform termasuk Windows 7, Windows Server 2008 R2, SQL Server 2008, SharePoint, Office, Windows Azure, dan Windows Phone.
2. Lingkungan pemrograman dapat dipersonalisasi
3. Dukungan *Application Lifecycle Management* dapat memudahkan kolaborasi, melakukan *testing* ataupun *debugging*.

2.11 WampServer

WAMP (*Windows and the principal components of the package: Apache, MySQL and PHP*) merupakan paket *web server* terdiri dari *Apache*, *database* MySQL dan PHP (*Hypertext Preprocessor*) yang bekerja pada localhost dan dibuat secara independen untuk digunakan pada sistem operasi Windows.

Beberapa kelebihan WAMP:

1. Menyediakan layanan web server yang bekerja secara *offline* (bekerja pada localhost).
2. Menyediakan layanan database MySQL secara *offline*.
3. Menyediakan layanan akses terhadap bahasa scripting PHP secara *offline*.
4. Menyediakan antarmuka pengguna berbasis web untuk mengelola database MySQL dengan adanya fitur phpMyAdmin dan SQLiteManager.