BAB II

DASAR TEORI

2.1 Telekomunikasi

Komunikasi adalah suatu proses penyampaian pesan (ide, gagasan) dari suatu pihak kepada pihak lain agar terjadi saling mempengaruhi dianatara keduanya. Pada umumnya, komunikasi di lakukan dengan menggunakan kata-kata (lisan) yang dapat di mengerti oleh kedua pihak. Apabila tidak ada bahasa verbal yang dapat dimengerti oleh keduanya, komunikasi masih dapat dilakukan dengan menggunakan gerak-gerik badan, menunjukkan sikap tertentu, misalnya tersenyum, menggelengkan kepala, mengangkat bahu. Cara seperti ini disebut komunikasi dengan bahasa nonverbal.

Di bidang komunikasi, kemajuan teknologi di indonesia telah mengalami kemajuan yang sangat pesat begitu juga dengan alat-alat telekomunikasi, seiring dengan kemajuan tersebut para praktisi Teknologi Informatikapun di tuntut untuk bisa membangun aplikasi yang bisa di gunakan dalam dunia telekomunikasi

2.2 Telepon Seluler (Ponsel)

Adalah perangkat telekomunikasi elektronik yang mempunyai kemampuan dasar yang sama dengan telepon fixed line konvensional, namun dapat dibawa ke mana-mana (portabel, mobile) dan tidak perlu disambungkan dengan jaringan telepon menggunakan kabel (nirkabel; wireless). Saat ini Indonesia mempunyai dua jaringan telepon nirkabel yaitu sistem *Global System For Mobile Telecommunications* (GSM) dan sistem *Code Division Multiple Access* (CDMA). Selain berfungsi untuk melakukan dan menerima

panggilan telepon, ponsel umumnya juga mempunyai fungsi pengiriman dan penerimaan pesan singkat *Short Message Service* (SMS). Mengikuti perkembangan teknologi digital, kini ponsel juga dilengkapi dengan berbagai pilihan fitur, seperti bisa menangkap siaran radio dan televisi, perangkat lunak pemutar audio dan video, kamera digital, game, dan layanan internet. Ada pula penyedia jasa telepon genggam di beberapa negara yang menyediakan layanan generasi ketiga dengan menambahkan jasa videophone, sebagai alat pembayaran, maupun untuk televisi online pada telepon genggam mereka. Sekarang, telepon genggam menjadi gadget yang multifungsi. Selain fitur-fitur tersebut, ponsel sekarang sudah ditanamkan fitur komputer. Jadi pada ponsel tersebut, orang bisa mengubah fungsi ponsel tersebut menjadi mini komputer. Di dunia bisnis, fitur ini sangat membantu bagi para pebisnis untuk melakukan semua pekerjaan di satu tempat dan membuat pekerjaan tersebut diselesaikan dalam waktu yang singkat. Mini komputer tersebut sering di sebut PDA adalah sebuah alat elektronik yang berbasis komputer dan berbentuk kecil serta dapat dibawa kemana-mana. PDA banyak digunakan sebagai pengorganisir pribadi pada awalnya, tetapi karena perkembangannya, kemudian bertambah banyak fungsi kegunaannya, seperti kalkulator, penunjuk jam dan waktu, permainan komputer, pengakses internet, penerima dan pengirim surat elektronik (e-mail), penerima radio, perekam video, dan pencatat memo. Selain dari itu, PDA bisa di gunakan untuk buku alamat dan menyimpan alamat, membaca buku elektronik, dan masih banyak lagi fungsi yang lain. Bahkan versi PDA yang lebih canggih dapat digunakan sebagai telepon genggam, akses internet, intranet, atau extranet lewat jaringan Wireless. Salah satu ciri khas PDA yang paling utama adalah fasilitas layar sentuh.

2.3 Layanan Internet Ponsel (WAP, GPRS, 3G)

Wireless Apliccation Protokol (WAP) adalah standar internasional terbuka untuk aplikasi yang menggunakan komunikasi nirkabel. Tujuan utamanya untuk membangun aplikasi yang dapat mengakses internet dari telepon genggam atau PDA. Ada beberapa versi WAP antara lain WAP 1.2.1 dan 2.0. WAP 1.2.1 hanya dapat menampilkan laman sederhana saja dibandingkan dengan WAP 2.0 yang mendukung bahasa xhtml dan gambar. WAP di buat pertama kali sebagai protokol komunikasi bergerak yang tidak bergantung pada sistem tertentu. WAP dirancang sebagai bagian dari sistem di masa depan sama halnya dengan Bluetooh dan GPRS. WAP merupakan protokol komunikasi bergerak yang terdiri dari beberapa layer dan dapat dijalankan pada sistem jaringan yang berbeda.

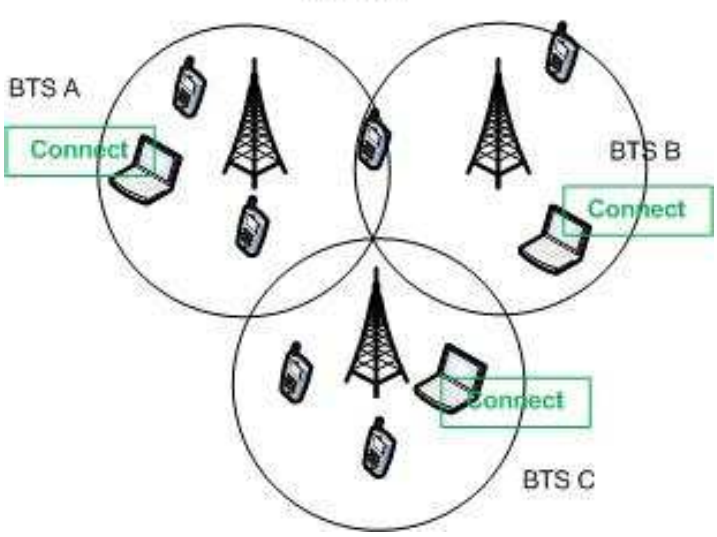
GPRS (*General Packet Radio Service)* merupakan sistem transmisi berbasis paket untuk GSM yang menggunakan prinsip "*tunneling*". Ia menawarkan laju data yang lebih tinggi. Laju datanya secara kasar sampai 60 kbps di bandingkan dengan 9.6 kbps yang dapat disediakan oleh rangkaian tersakelar GSM. Kanal-kanal radio ganda dapatdialokasikan bagi seorang pengguna dan kanal yang sama dapat pula digunakan secara berbagi (sharing) di antara beberapa pengguna sehingga menjadi sangat efisien.

Teknologi GPRS merupakan salah satu pendukung untuk mengakses internet secara cepat dan murah bahkan sangat membantu bagi pengguna dengan mobilitas kerja tinggi yang membutuhkan akses internet dimanapun dan kapanpun, namun banyak menghabiskan waktu diluar kantor. Teknologi GPRS merupakan jaringan selular generasi ke 2,5 (setelah GSM dan sebelum 3G) yang sudah populer dikalangan pengguna ponsel, notebook dan PDA.

GPRS merupakan teknologi baru yang memungkinkan para operator jaringan komunikasi bergerak menawarkan layanan data dengan laju bit yang lebih tinggi dengan tarif rendah, sehingga membuat layanan data menjadi menarik bagi massal. Para operator jaringan komunikasi bergerak di luar negeri kini melihat GPRS sebagia kunci untuk mengembangkan pasar komunikasi bergerak menjadi pesaing baru di lahan yang pernah menjadi milik jaringan kabel, yakni layanan internet. Kondisi ini di mungkinkan kerna penggunaan internet melalui jaringan kabel (telepon) dpat pula di lakukan melalui jaringan bergerak. Sebagai gambaran kecil, layanan bergerak yang kini menjadi sukses di pasar (bagi operator di manca negara) misalnya adalah, laporan cuaca, pemesanan makanan, berita olah raga sampai ke informasi seperti berita-berita penting harian. Dalam teorinya GPRS menjajikan kecepatan mulai dari 56 kbps sampai 115 kbps, sehingga memungkinkan akses internet, pengiriman data multimedia ke komputer, notebook dab handled computer. Namun, dalam implementasinya hal tersebut sangat tergantung pada faktor-faktor sebagai berikut:

1. Konfigurasi dan alokasi time slot pada level BTS
2. Software yang di gunakan
3. Dukungan fitur dan aplikasi ponsel yang di gunakan

Ini menjelaskan mengapa pada saat-saat tertentu dana di lokasi tertentu akses GPRS terasa lambat, bahkan lebih lambat dari akses CSD yang memiliki kecepatan 9,6 kbps. Gambar 2.1 di bawah ini merupakan gambar dari sebuah koneksi GPRS dalam suatu area.



Gambar 2.1 Jaringan GPRS yang terdiri dari beberapa BTS

Komponen-komponen utama jaringan GPRS adalah:

1. GGSN (Gateway GPRS Support Node)

   Adalah gerbang penghubung jaringan GPRS ke jaringan internet. Fungsi dari komponen ini adalah sebagai interface ke PDN (Public Data Network), information routing , network screening, user screening, address mapping.

2. SGSN (Serving GPRS Support Node)

   Adalah gerbang penghubung jaringan BSS/BTS ke jaringan GPRS. Kompnen ini berfungsi untuk mengantarkan paket data ke Mobile Station (MS), update pelanggan ke HLR, registrasi pelanggan baru.

3. PCU (Packet Control Unit)

   Adalah komponen di level BSS yang menghubungkan terminal ke jaringan GPRS

Berikut adalah cara kerja GPRS:

SGSN bertugas:

1. Mengirim paket ke Mobile Station (MS) dalam satu area.

2. Mengirim sejumlah pertanyaan ke HLR untuk memperoleh Profile data pelanggan GPRS (management Mobility).
3. Mendeteksi MS-GPRS yang baru dalam satu area servis yang menjadi tanggung jawabnya (location management).
4. SGSN di hubungkan ke BSS pada GSM dengan koneksi Frame Relay melalui PCU (Packet Control Unit) di dalam BSC.

GGSN bertugas:

1. Sebagai Interface ke jaringan IP eksternal seperti pada public internet atau mobile service provider.
2. Mengupdate informasi routing dari PDU (*Protokol Data Units*) ke SGSN.

GPRS menggunakan sistem komunikasi *Packet Switch* sebagai cara untuk mentransmisikan datanya. *Packet Switch* adalah sebuah sistem di mana data yang akan di transmisikan di bagi menjadi bagian-bagian kecil (paket) lalu di transmisikan dan di ubah kembali menjadi data semula. Sistem ini dapat mentransmisikan ribuan bahkan jutaan paket per detik. Transmisi di lakukan melalui PLMN (*Public Land Mobile Network*) dengan menggunakan IP backbone. Karena memungkinkan untuk pemakaian kanal transmisi secara bersamaan oleh pengguna lain maka biaya akses GPRS, secara teori lebih murah biaya akses CSD.

GPRS juga di desain untuk menyediakan layanan transfer packet data pada jaringan GSM dengan kecepatan yang lebih baik dari GSM. Kecapatan yang lebih baik ini di dapat dengan menggunakan *Coding Scheme* (CS) yang berbeda dari GSM.

2.4 HSCSD dan HSDPA

High Speed Circuit Switched Data (HSCSD) adalah sebuah perkembangan dari teknologi Circuit Switched Data (CSD) dan merupakan mekanisme dari data transmisi hanya untuk sistem selular yang menggunakan jaringan Global System for Mobile Communications (GSM). *High-Speed Circuit-Switched Data* (HSCSD) juga dapat diartikan sebagai sebuah spesifikasi untuk mentransfer data melalui jaringan *Global System for Mobile Communications* (GSM). HSCSD adalah sebuah service circuit switch yang digunakan untuk aplikasi dengan bandwidth yang lebih tinggi (14,4 kbps) seperti data stream, gambar bergerak, dan video telephony. Bandwidth data yang lebih besar ini didapatkan dengan menggabungkan 1 sampai dengan 8 time slot untuk satu subscriber (bundling of channels). HSCSD hanya digunakan untuk komunikasi point-to-point.

High-Speed Circuit-Switched Data (HSCSD) merupakan teknologi yang berkembang pada teknologi selular generasi ke 2,5 (2,5 Generation). *Enhanced Data rate for GSM Evolution* (EDGE) memungkinkan jaringan *Global System for Mobile Communications* (GSM) untuk dapat melaksanakan *Enhanced Circuit Switched Data* (ECSD) yang merupakan perkembangan dari *High-Speed Circuit-Switched Data* (HSCSD). *Enhanced Circuit Switched Data* (ECSD) menambah bandwidh dari tiap kali slot menjadi 48 kbit/s dan memberikan manfaat dari delapan kali slot, yang memberikan total kecepatan transmisi sebesar 384 kbit/s.

High-Speed Circuit-Switched Data / *Enhanced Circuit Switched Data* (ECSD) dan GPRS akhirnya tersingkirkan oleh *Universal Mobile Telecommunications System* (UMTS), yang memiliki teknologi packet-switched dengan kecepatan transmisi sebesar 2 Mbit/s.

Inovasi pertama pada *High-Speed Circuit-Switched Data* adalah menyediakan metode koreksi kesalahan (*error correction method*) yang berbeda, yang digunakan untuk mentransfer data. Koreksi kesalahan (*error correction*) yang digunakan pada GSM adalah rancangan untuk bekerja pada batasan dari ulasan dan pada kasus terburuk yang akan ditangani oleh GSM. Yang berarti bahwa sebagian besar kapasitas transmisi GSM dikendalikan dengan kode-kode koreksi kesalahan (*error correction codes*). *High-Speed Circuit-Switched Data* menyediakan tingkat-tingkat yang berbeda dari koreksi kesalahan (*error correction*) yang tepat yang dapat digunakan berdasarkan pada kualitas dari hubungan radio (*radio link*). Ini berarti bahwa pada kondisi terbaik 14,4 Kbit/s dapat dikeluarkan melalui kali slot tunggal yang di bawah kemampuan *Circuit Switched Data* (CSD).

Inovasi yang lain pada *High-Speed Circuit-Switched Data* adalah kemampuan untuk menggunakan multiple kali slot secara bersamaan. High-*Speed Circuit-Switched Data* menggunakan maksimum dari 4 kali slot, yang dapat memberikan sebuah peningkatan pada kecepatan transfer yang maksimum hingga 57,6 kbit/s atau sama dengan empat kali lipat dari 14,4 kbit/s. Walaupun dalam kondisi radio yang buruk dimana dibutuhkan koreksi kesalahan dengan level yang lebih tinggi, High-Speed Circuit-Switched Data masih dapat menyediakan kecepatan transfer sebesar empat kali lipat kecepatan yang lebih tinggi dari kecepatan transfer yang dimiliki Circuit Switched Data yaitu 38,4 kbit/s atau sama dengan empat kali lipat dari 9,6 kbit/s.

Perbandingan antara HSCSD dengan GPRS adalah sebagai berikut:

a. HSCSD memiliki keuntungan diatas GPRS dimana HSCSD menyokong garansi kualitas dari layanan karena mempersembahkan saluran komunikasi circuit-

switched. Ini membuat HSCSD menjadi protokol yang lebih baik untuk aplikasi timing-sensitive seperti mentransfer gambar atau video.

b. GPRS juga memiliki keuntungan diatas HSCSD untuk mentransfer data yang terbanyak karena HSCSD yang menggunakan circuit-switched memiliki efisiensi bandwidth yang kurang dan hubungan nirkable (*wireless link*) yang lebih mahal daripada GPRS yang menggunakan pergantian paket (*packet-switched*).
c. HSCSD tidak mengalami perkembangan seluas perkembangan GPRS. Walaupun demikian, HSCSD pada saat sekarang ini telah tersebar di lebih dari 27 negara.

High-Speed Downlink Packet Access (HSDPA) adalah sebuah protokol telepon genggam dan kadangkala disebut sebagai teknologi **3,5G.** HSDPA fase pertama berkapasitas 4,1 Mbps. Kemudian menyusul fase 2 berkapasitas 11 Mbps dan kapsitas maksimal *downlink peak data rate* hingga mencapai 14 Mbit/s. Teknologi ini dikembangkan dari WCDMA sama seperti EV-DO mengembangkan CDMA2000. HSDPA memberikan jalur evolusi untuk jaringan *Universal Mobile Telecommunications System* (UMTS) yang memungkinkan untuk penggunaan kapasitas data yang lebih besar (sampai 14,4 Mbit/detik arah turun).

HSDPA merupakan evolusi dari standar W-CDMA dan dirancang untuk meningkatkan kecepatan transfer data 5x lebih tinggi. HSDPA memdefinisikan sebuah saluran W-CDMa yang baru, yaitu *high-speed downlink shared channel* (HS-DSCH) yang cara operasinya berbeda dengan saluran W-CDMA yang ada sekarang. Hingga kini penggunaan teknologi HSDPA hanya pada komunikasi arah bawah menuju telepon genggam. Adapun kecepatan unduh data dari HSDPA ini berbeda-beda tergantung tempat di mana dia melakukan pengunduhan data, contohnya:

a. Di lingkungan perumahan teknologi ini dapat melakukan unduh data hingga berkecepatan 3,7 Mbps.
b. Dalam keadaan bergerak seseorang yang sedang berkendaraan di jalan tol berkecepatan 100 km/jam dapat mengakses internet berkecepatan 1,2 Mbps.
c. Di lingkungan perkantoran yang padat pengguna dapat menikmati streaming video dengan perkiraan kecepatan 300 Kbps.

Kelebihan dari HSDPA ini adalah mengurangi tertundanya pengunduhan data (*delay*) dan memberikan umpan balik yang lebih cepat saat pengguna menggunakan aplikasi interaktif seperti *mobile office* atau akses Internet kecepatan tinggi untuk penggunaan fasilitas permainan atau mengunduh audio dan video. Kelebihan lain HSDPA, meningkatkan kapasitas sistim tanpa memerlukan spektrum frekuensi tambahan. Hal ini menyebabkan berkurangnya biaya layanan mobile data secara signifikan.

2.5 Bahasa Pemrograman C#.net

Untuk memahami tentang bahasa C#, terlebih dulu harus mengerti apa itu Microsoft Framework.NET yang merupakan komponen yang tidak bisa di pisahkan dari bahasa C# itu sendiri.

2.5.1 Microsoft Framework.NET

Framework .NET adalah suatu komponen windows yang terintegrasi yang dibuat dengan tujuan untuk mensupport pengembangan berbagai macam jenis aplikasi serta untuk dapat mejalankan berbagai macam aplikasi generasi mendatang termasuk pengembangan aplikasi Web Services XML.

Framework .NET di design untuk dapat memenuhi beberapa tujuan berikut ini :

1. Untuk menyediakan environment kerja yang konsisten bagi bahasa pemrograman yang berorientasi objek (*object-oriented programming - OOP*) baik kode objek itu di simpan dan di eksekusi secara lokal, atau dieksekusi secara lokal tapi didistribusikan melalui internet atau di eksekusi secara remote.
2. Untuk menyediakan environment kerja di dalam mengeksekusi kode yang dapat meminimaliasi proses software *deployment* dan menghindari konflik penggunaan versi software yang di buat.
3. Untuk menyediakan environment kerja yang aman dalam hal pengeksekusian kode, termasuk kode yang dibuat oleh pihak ketiga (*third party).*
4. Untuk menyediakan environment kerja yang dapat mengurangi masalah pada persoalan performa dari kode atau dari lingkungan *interpreter* nya.
5. Membuat para developer lebih mudah mengembangkan berbagai macam jenis aplikasi yang lebih bervariasi, seperti aplikasi berbasis windows dan aplikasi berbasis web.
6. Membangun semua komunikasi yang ada di dalam standar industri untuk memastikan bahwa semua kode aplikasi yang berbasis Framework .NET dapat berintegrasi dengan berbagai macam kode aplikasi lain.

Sebagai salah satu sarana untuk dapat memenuhi tujuan di atas, maka dibuatlah berbagai macam bahasa pemrograman yang dapat digunakan dan dapat berjalan di atas platform Framework .NET seperti bahasa C#, VB.NET, J#, Perl.NET dan lain-lain. Masing-masing bahasa tersebut mempunyai kelebihan dan kekurangannya masing-masing, namun yang pasti, apapun bahasa pemrograman yang digunakan, semuanya akan

dapat saling berkomunikasi dan saling compatible satu dengan yang lainya dengan bantuan Framework .NET

Arsitektur Framework .NET

Framework .NET terdiri dari dua buah komponen utama, yaitu *Common Language Runtime (CLR)* dan *.NET Framework Class Library* atau kadang juga sering disebut dengan *Base Class Library (BCL).*

Common Language Runtime (CLR) adalah pondasi utama dari Framework .NET. CLR merupakan komponen yang bertanggung jawab terhadap berbagai macam hal, seperti bertanggung jawab untuk melakukan managemen memory, melakukan eksekusi kode, melakukan verifikasi terhadap keamanan kode, menentukan hak akses dari kode, melakukan kompilasi kode, dan berbagai layanan system lainnya. Dengan adanya fungsi CLR ini, maka aplikasi berbasis .NET biasa juga disebut dengan *managed code*, sedangkan aplikasi di luar itu biasa disebut dengan *un-managed code*.

Berikut ini beberapa hal yang disediakan CLR bagi para developer:

1. Dapat lebih menyederhakan proses pengembangan aplikasi.
2. Memungkinkan adanya variasi dan integrasi dari berbagai bahasa pemrograman yang ada di lingkunan Framework .NET
3. Keamanan dengan melakukan identing pada kode aplikasi.
4. Bersifat *Assembly* pada saat proses deployment / kompilasi
5. Melakukan versioning sebuah komponen yang bisa di daur ulang.
6. Memungkinkan penggunaan kembali kode, dengan adanya sifat inheritance.
7. Melakukan pengaturan / manajemen tentang *lifetime* sebuah objek.
8. Melakukan penganalisaan objek-objek secara otomatis.

CLR akan melakukan kompilasi kode-kode aplikasi menjadi bahasa assembly MSIL (*Microsoft Intermediate Language*). Proses kompilasi ini sendiri dilakukan oleh komponen yang bernama *Just In Time* (JIT). JIT hanya akan mengkompilasi metode-metode yang memang digunakan dalam aplikasi, dan hasil kompilasi ini sendiri di *chace* di dalam mesin dan akan dikompile kembali jika memang ada perubahan pada kode aplikasi.

.NET Framework Class Library atau sering juga disebut *Base Case Library* (BCL) adalah koleksi dari *reusable types* yang sangat terintegrasi secara melekat dengan CLR. *Class library* bersifat berorientasi terhadap objek yang akan menyediakan *types* dari fungsi-fungsi *managed code*. Hal ini tidak hanya berpengaruh kepada kemudahan dalam hal penggunaan, tetapi juga dapat mengurangi waktu yang diperlukan pada saat eksekusi. Dengan sifat tersebut, maka komponen pihak ketiga akan dengan mudah diaplikasikan ke dalam aplikasi yang dibuat.

Dengan adanya BCL ini, maka bisa menggunakan Framework .NET untuk membuat berbagai macam aplikasi, seperti Aplikasi *console,* Aplikasi berbasis windowd (*Windows Form*), Aplikasi ASP.NET (berbasis web), Aplikasi Web Services XML, Aplikasi berbasis *Windows Services,* Jika membuat sekumpulan Class untuk membuat aplikasi berbasis windows, maka Class-Class itu bisa di gunakan untuk jenis aplikasi lain, seperti aplikasi berbasis web (ASP.NET).

2.5.2 Pengenalan Bahasa C#

C# (dibaca “See-Sharp”) adalah bahasa pemrograman baru yang diciptakan oleh Microsoft (dikembangkan dibawah kepemimpinan Anders Hejlsberg yang notabene juga

telah menciptakan berbagai macam bahasa pemrograman termasuk Borland Turbo C++ dan Borland Delphi). Bahasa C# juga telah di standarisasi secara internasional oleh ECMA. Seperti halnya bahasa pemrograman yang lain, C# bisa digunakan untuk membangun berbagai macam jenis aplikasi, seperti aplikasi berbasis windows (desktop) dan aplikasi berbasis web serta aplikasi berbasis web services. Ada beberapa alasan mengapa memilih bahasa C#, yaitu:

Sederhana (simple)

C# menghilangkan beberapa hal yang bersifat kompleks yang terdapat dalam beberapa macam bahasa pemrograman seperti Java dan C++, termasuk diantaranya mengilangkan macro, templates, multiple inheritance dan virtual base classes. Hal-hal tersebut yang dapat menyebabkan kebingunan pada saat menggunakannya, dan juga berpotensial dapat menjadi masalah bagi para programmer C++. Jika anda pertama kali belajar bahasa C# sebagai bahasa pemrograman, maka hal-hal tersebut di atas tidak akan membuat waktu anda terbuang terlalu banyak untuk mempelajarinya.

C# bersifat sederhana, karena bahasa ini didasarkan kepada bahasa C dan C++. Jika anda familiar dengan C dan C++ atau bahkan Java, anda akan menemukan aspek-aspek yang begitu familiar, seperti statements, expression, operators, dan beberapa fungsi yang diadopsi langsung dari C dan C++, tetapi dengan berbagai perbaikan yang membuat bahasanya menjadi lebih sederhana.

Modern

Apa yang membuat C# menjadi suatu bahasa pemrograman yang modern? Jawabannya adalah adanya beberapa fitur seperti exception handling, garbage collection, extensible data types, dan code security (keamanan kode/bahasa pemrograman). Dengan

adanya fitur-fitur tersebut, menjadikan bahasa C# sebagai bahasa pemrograman yang modern.

Object-Oriented Language

Kunci dari bahasa pemrograman yang bersifat Object Oriented adalah encapsulation, inheritance, dan polymorphism. Secara sederhana, istilah-istilah tersebut bisa didefinisikan sebagai berikut:

encapsulation, dimana semua fungsi ditempatkan dalam satu paket (single package).

inheritance, adalah suatu cara yang terstruktur dari suatu kode-kode pemrograman dan fungsi untuk menjadi sebuat program baru dan berbentuk suatu paket.

polymorphism, adalah kemampuan untuk mengadaptasi apa yang diperlukan untuk dikerjakan.

Sifat-sifat tersebut di atas, telah di miliki oleh C# sehingga bahasa C# merupakan bahasa yang bersifat Object Oriented.

Powerfull dan fleksibel

C# bisa digunakan untuk membuat berbagai macam aplikasi, seperti aplikasi pengolah kata, grapik, spreadsheets, atau bahkan membuat kompiler untuk sebuah bahasa permrograman.

Efisien

C# adalah bahasa pemrograman yang menggunakan jumlah kata-kata yang tidak terlalu banyak. C# hanya berisi kata-kata yang biasa disebut dengan keywords. Keywords ini digunakan untuk menjelaskan berbagai macam informasi. Jika anda berpikiran bahwa bahasa pemrograman yang menggunakan sangat banyak kata-kata (keywords) akan lebih powerfull, maka jawabannya adalah “pemikiran itu tidak selalu benar”, karena hal itu

justru bisa menambah kerumitan para developer pada saat membuat suatu aplikasi. Pada gambar 2.2 di perlihatkan daftar keywords yang ada dalam bahasa C#:

```
abstract   as         base     bool      break
byte       case       catch    char      checked
class      const      continue decimal   default
delegate   do         double   else      enum
event      explicit   extern   false     finally
fixed      float      for      foreach
goto       if         implicit in        int
interface  internal   is       lock      long
namespace  new        null     object    operator
out        override   params   private   protected
public     readonly   ref      return    sbyte
sealed     short      sizeof   stackalloc
static     string     struct   switch    this
throw      true       try      typeof    uint
ulong      unchecked  unsafe   ushort    using
virtual     void      while
```

Gambar 2.2 Daftar Keywords dalam bahasa C#

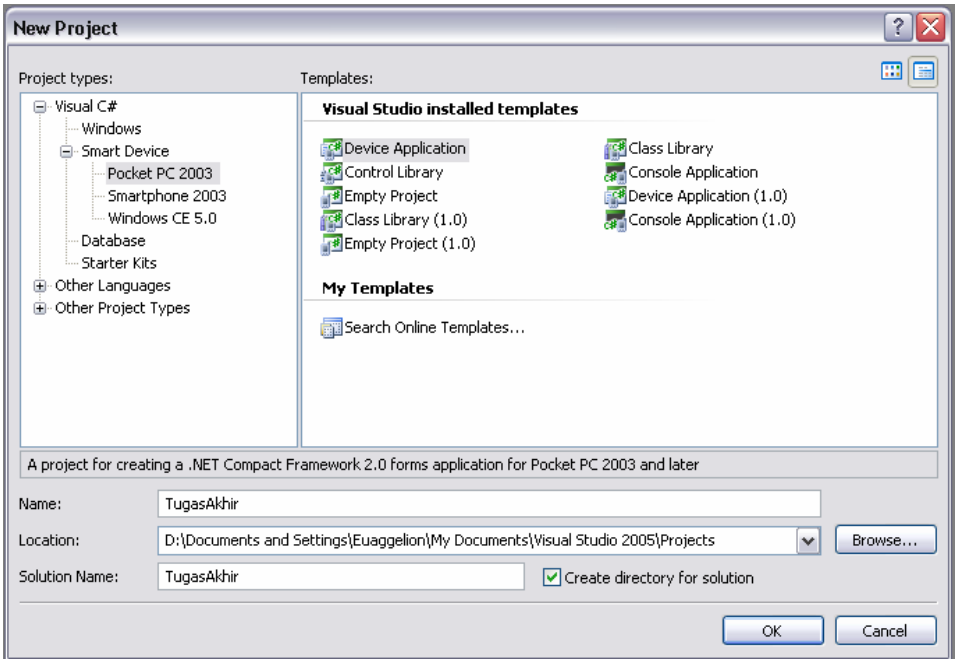
Modular

Kode C# ditulis dengan pembagian masing Class-Class (classes) yang terdiri dari beberapa routines yang disebut sebagai member methods. Class-Class dan metode-metode ini dapat digunakan kembali oleh program atau aplikasi lain. Hanya dengan memberikan informasi yang dibutuhkan oleh Class dan metode yang dimaksud, maka dapat di hasilkan suatu kode yang dapat digunakan oleh satu atau beberapa aplikasi dan program (reusable code)

C# akan menjadi populer

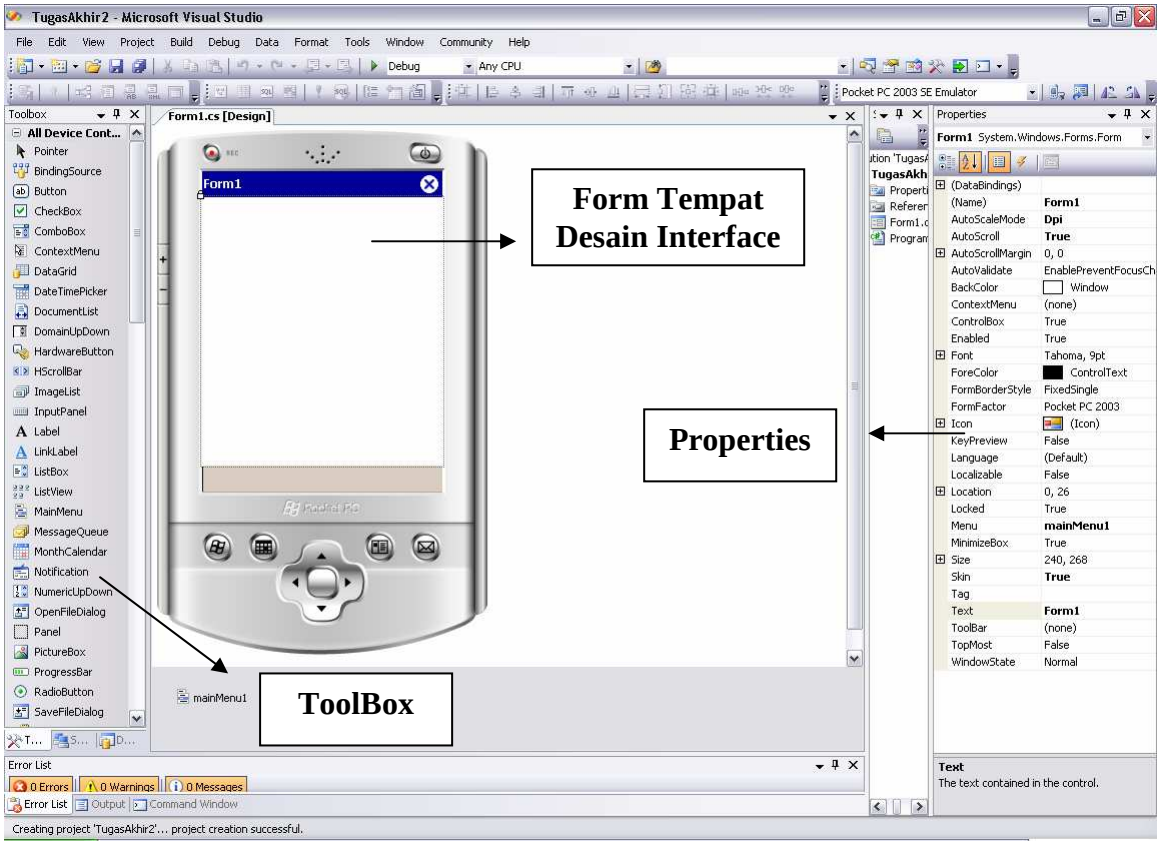
Dengan dukungan penuh dari Microsoft yang akan mengeluarkan produk-produk utamanya dengan dukungan Framework .NET, maka masa depan bahasa C# sebagai salah satu bahasa pemrograman yang ada di dalam lingkungan Framework .NET akan lebih baik. Dalam melakukan coding di gunakan editor Visual Studio.NET. Visual

Studio.NET merupakan editor yang paling ideal untuk membuat aplikasi yang berbasis Framework.NET, termasuk aplikasi dengan bahasa C#. Dan pengolahan data dilakukan menggunakan SQL yang bisa di koneksikan dengan Visual Studio.NET.



Gambar 2.3 Form untuk membuat Project baru

Gambar 2.3 di atas dan gambar 2.4 di bawah adalah gambar Form kerja dari Visual Studio 2005 yang menggunakan bahasa C# dengan program untuk aplikasi mobile.

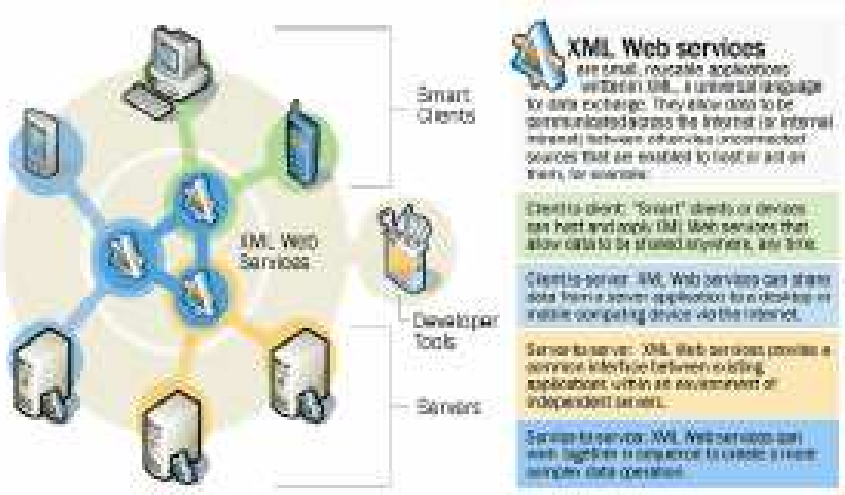


Gambar 2.4 Form kerja Visual Studio 2005 untuk aplikasi Mobile.

2.6 Web Service

Web Services merupakan salah satu bentuk implementasi dari arsitektur model aplikasi N-Tier yang berorientasi layanan. Perbedaan *Web Services* dengan pendekatan N-Tier lainnya adalah dari segi infrastruktur dan dokumen yang digunakan sebagai format pertukaran data. Dalam implementasinya, *Web Services* tidak mempunyai tampilan, karena *Web Services* termasuk dalam Business-Service tier. Artinya didalam *Web Services* hanya tersedia fungsi-fungsi yang nantinya dapat digunakan oleh aplikasi lainnya *Web Services* menggunakan XML sebagai format dokumen dalam melakukan pertukaran datanya. Karena XML merupakan suatu format dokumen yang berbasis teks, maka *Web Services* memungkinkan berlangsungnya komunikasi antar aplikasi yang berbeda dengan platform yang berbeda pula. *Web Services* dapat diimplementasikan dalam berbagai jenis platform dengan menggunakan bahasa pemrograman apa pun, dan bisa digunakan oleh berbagai aplikasi yang menggunakan bahasa pemrograman apapun dengan platform apapun juga. Selama aplikasi tersebut dapat berkomunikasi dengan *Web*

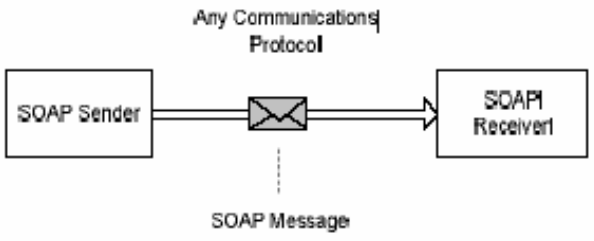
Services menggunakan protokol-protokol komunikasi. Termasuk HTTP, XML, SOAP, UDDI (Universal Description Discovery and Integration), dan WSDL (Web Services Description Language). Pada Gambar 2.4 berikut merupakan gambaran sederhana konsep *Web Services* serta keterhubungan antara *Web Services* dengan aplikasi Client maupun Server



Gambar 2.5 Webservice Concepts

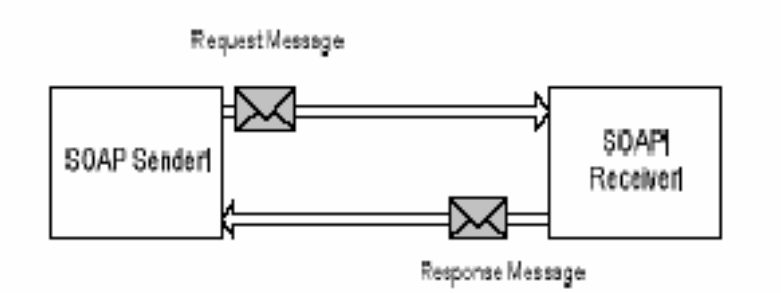
Simple Object Access Protocol (SOAP)

SOAP adalah sebuah protokol yang menyediakan sebuah cara standar untuk memaketkan pesan dalam lingkungan terdistribusi yang menggunakan XML, dan memfasilitasi komunikasi bergaya RPC (Remote Procedure Call) antara *Remote Client* dan *Server*. SOAP merupakan dokumen XML yang mengatur bagaimana request dan respons dari suatu web services akan bekerja. SOAP menggunakan teknologi XML untuk menggambarkan suatu kerangka *extensible messaging* yang menyediakan konstruksi pesan yang dapat dipertukarkan menggunakan beberapa protokol dasar seperti HTTP, FTP, SMTP atau yang lain.



Gambar 2.6 Sistem Pesan SOAP sederhana

Pesan yang dikirim antara SOAP *sender* dan SOAP *receiver* disebut SOAP *Message* atau SOAP *Envelope*.



Gambar 2.7 Model Pertukaran Pesan Request/Respone

Web Services Description Language (WSDL)

WSDL merupakan suatu dokumen XML yang menjelaskan method-method apa saja yang tersedia dalam suatu *web services*, parameter apa saja yang diperlukan untuk memanggil suatu method, dan apa hasil dari method yang akan dipanggil. Sintaks dokumen WSDL.

Web Service pada Platform.Net

Dengan menggunakan platform .Net, banyak sekali kemudahan yang dapat diperoleh, dimana user tidak lagi dipusingkan dengan detail proses untuk memformat dan mengirim data dengan menggunakan protokol SOAP. Selain itu dukungan *Common Language Runtime* memungkinkan untuk membangun *Web Service* dengan menggunakan berbagai macam bahasa yang disediakan seperti C++, VB, C#, J#.

2.7 Rekayasa Perangkat Lunak

Dalam suatu proses pengembangan aplikasi. Analisa dan rancangan telah menerapkan terminologi yang sangat tua. Pada saat masalah di telusuri dan spesifikasi di negosiasikan, dapat dikatakan bahwa saat ini berada dalam tahap perancangan. Merancang atau merekayasa perangkat lunak adalah menemukan suatu cara untuk menyelesaikan masalah. Dalam membangun perangkat lunak yang berbasis *Object*

Oriented salah satu tool/model yang di gunakan adalah pemodelan dengan UML (Unified Modeling Language).

Konsep Objek

Obyek dalam '*software analysis & design'* adalah sesuatu berupa konsep (*concept*), benda (*thing*), dan sesuatu yang membedakannya dengan lingkungannya. Secara sederhana obyek adalah mobil, manusia, *alarm* dan lainlainnya. Tapi obyek dapat pula merupakan sesuatu yang abstrak yang hidup didalam sistem seperti tabel, *database, event, system messages.* Obyek dikenali dari keadaannya dan juga operasinya. Sebagai contoh sebuah mobil dikenali dari warnanya, bentuknya, sedangkan manusia dari suaranya. Ciri-ciri ini yang akan membedakan obyek tersebut dari obyek lainnya. Alasan mengapa saat ini pendekatan dalam pengembangan software dengan *object-oriented*, pertama adalah *scalability* dimana obyek lebih mudah dipakai untuk menggambarkan sistem yang besar dan komplek. Kedua *dynamic modeling*, adalah dapat dipakai untuk permodelan sistem dinamis dan *real time*.

Teknik Dasar OOA/D *(Object-Oriented Analysis/Design)*

Dalam dunia pemodelan, metodologi implementasi obyek walaupun terikat kaidah-kaidah standar, namun teknik pemilihan obyek tidak terlepas pada subyektifitas software analyst & designer. Beberapa obyek akan diabaikan dan beberapa obyek menjadi perhatian untuk diimplementasikan di dalam sistem. Hal ini sah-sah saja karena kenyataan bahwa suatu permasalahan sudah tentu memiliki lebih dari satu solusi. Ada 3 (tiga) teknik/konsep dasar dalam OOA/D, yaitu pemodulan (*encapsulation*), penurunan (*inheritance)* dan *polymorphism.*

a. Pemodulan (*Encapsulation*)

Pada dunia nyata, seorang ibu rumah tangga menanak nasi dengan *menggunakan rice cooker*, ibu tersebut menggunakannya hanya dengan menekan tombol. Tanpa harus tahu bagaimana proses itu sebenarnya terjadi. Disini terdapat penyembunyian informasi milik *rice cooker*, sehingga tidak perlu diketahui seorang ibu. Dengan demikian menanak nasi oleh si ibu menjadi sesuatu yang menjadi dasar bagi konsep *information hiding*.

b. Penurunan (*Inheritance*)

Obyek-obyek memiliki banyak persamaan, namun ada sedikit perbedan. Contoh dengan beberapa buah mobil yang mempunyai kegunaan yang berbeda-beda. Ada mobil bak terbuka seperti truk, bak tertutup seperti sedan dan minibus. Walaupun demikian obyek-obyek ini memiliki kesamaan yaitu teridentifikasi sebagai obyek mobil, obyek ini dapat dikatakan sebagai obyek induk (*parent*). Sedangkan minibus dikatakan sebagai obyek anak (*child*), hal ini juga berarti semua operasi yang berlaku pada mobil berlaku juga pada minibus.

c. *Polymorphism*

Pada obyek mobil, walaupun minibus dan truk merupakan jenis obyek mobil yang sama, namun memiliki juga perbedaan. Misalnya suara truk lebih keras dari pada minibus, hal ini juga berlaku pada obyek anak (*child*) melakukan metoda yang sama dengan algoritma berbeda dari obyek induknya. Hal ini yang disebut *polymorphism*, teknik atau konsep dasar lainnya adalah ruang lingkup / pembatasan. Artinya setiap obyek mempunyai ruang lingkup kelas, atribut, dan metoda yang dibatasi.

Pengenalan UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML dapat di buat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET.

Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti 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).

Konsep Dasar UML

Dari berbagai penjelasan tentang konsep dasar UML dapat di rangkum menjadi sebuah skema yang di gambarkan pada Gambar 2.7 berikut ini.

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Gambar 2.8 Konsep dasar UML

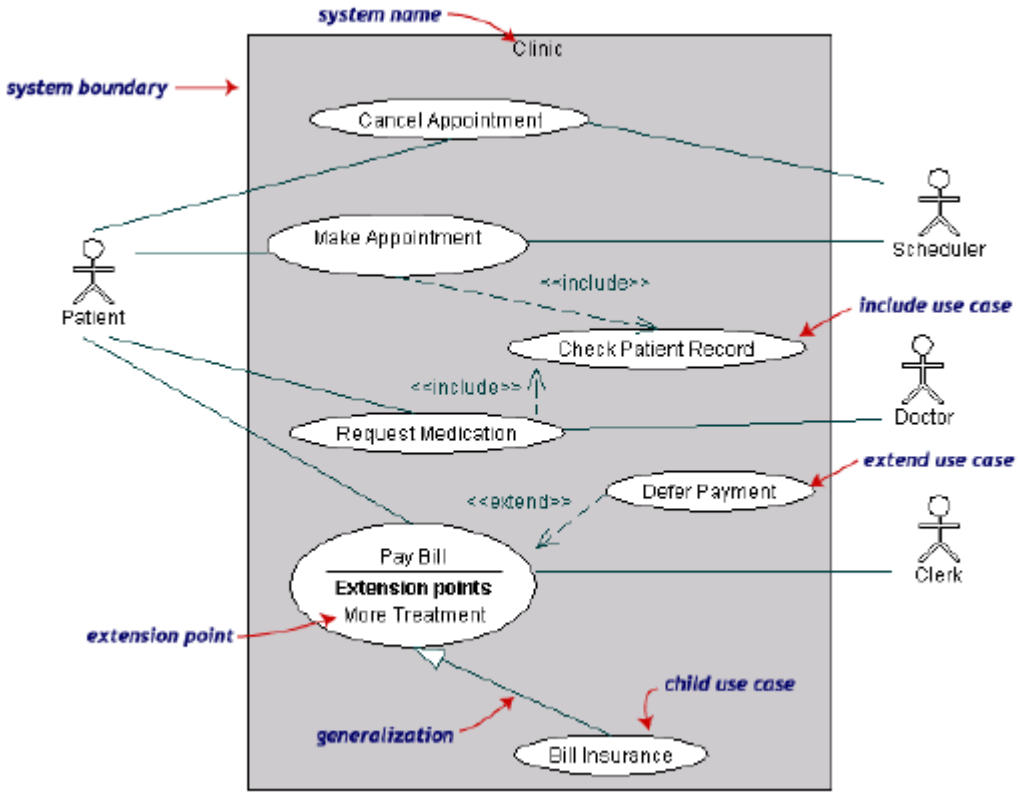
Seperti tercantum dalam gambar 2.8 di atas UML mendefinisikan diagram-diagram sebagai berikut:

1. Use Case diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah

daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. Penggambaran Use case diagram dapat di lihat dalam gambar 2.9 di bawah ini.



Gambar 2.9 Contoh Use Case diagram

2. Class diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

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

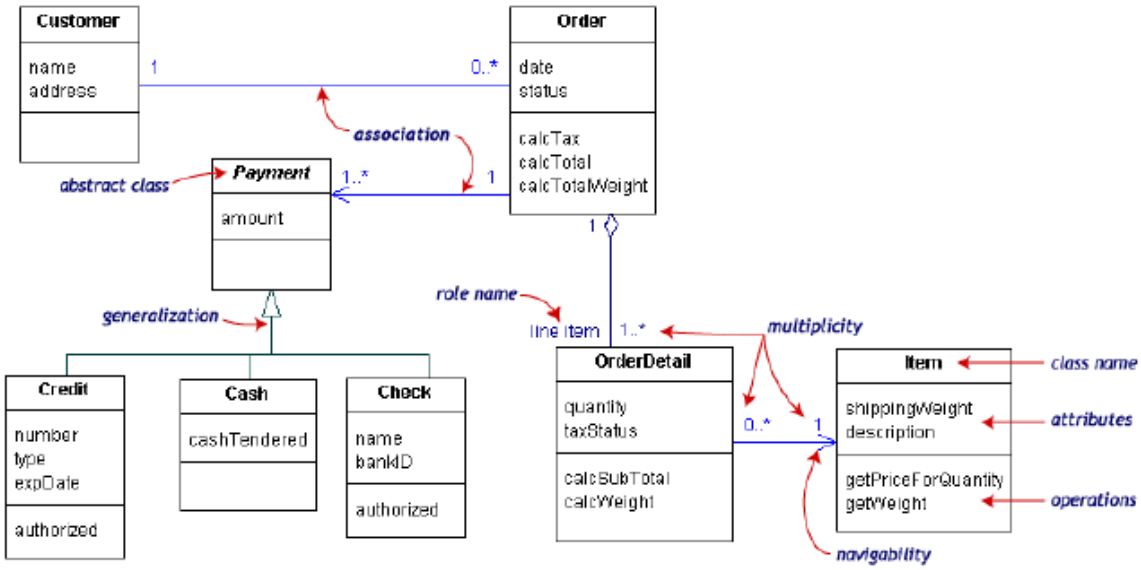
Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.

2. Agregasi, yaitu hubungan yang menyatakan bagian (“*terdiri atas..”*).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

Penggambaran Class diagram dapat di lihat pada gambar 2.10 di bawah ini.

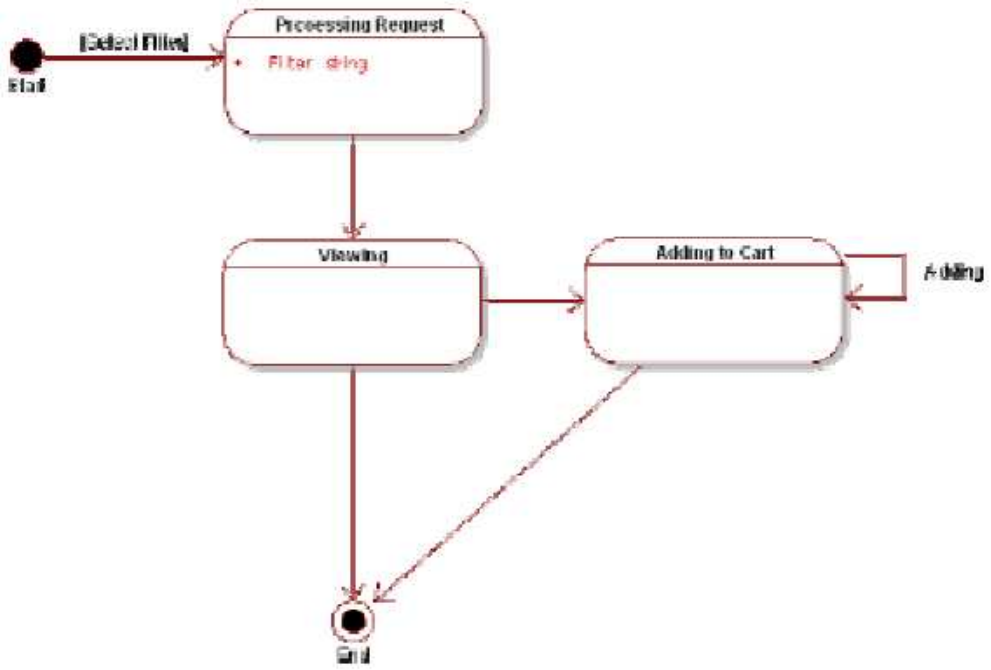


Gambar 2.10 Contoh Class diagram

3. Statechart diagram

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

segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* 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. Bisa di lihat dalam gambar 2.11 di bawah ini.



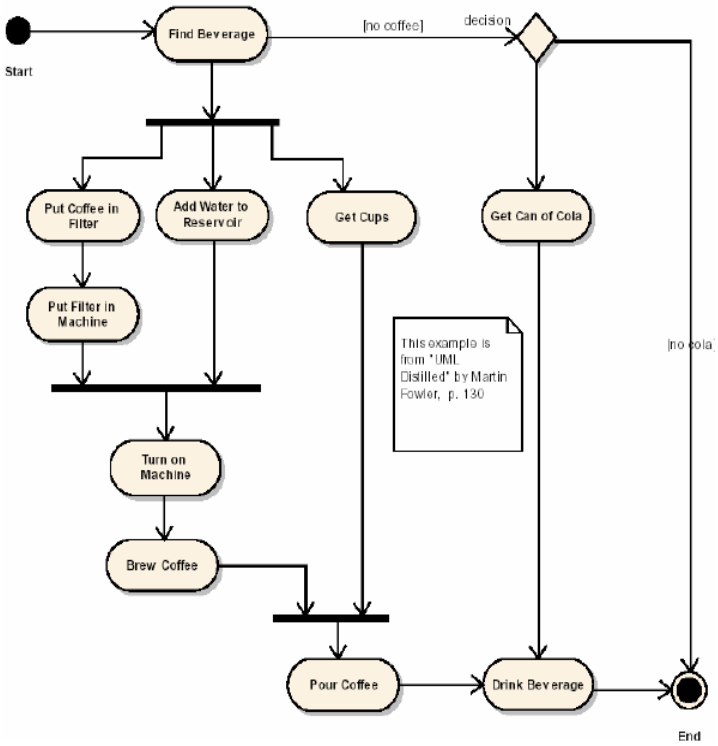
Gambar 2.11 Contoh Statechart diagram

4. Activity diagram

Activity diagrams 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 (dan interaksi antar subsistem) 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. *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. Contoh Activity diagram tanpa swimlane dapat di lihat pada gambar 2.12 berikut ini

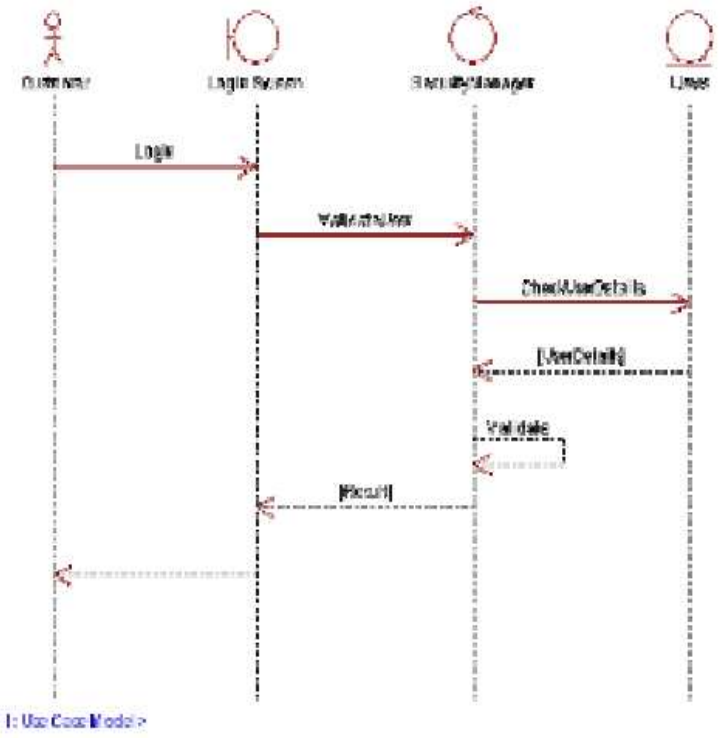


Gambar 2.12 Contoh Activity diagram

5. Sequence diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* 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 respons dari sebuah *event* untuk menghasilkan *output* tertentu. Di awali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

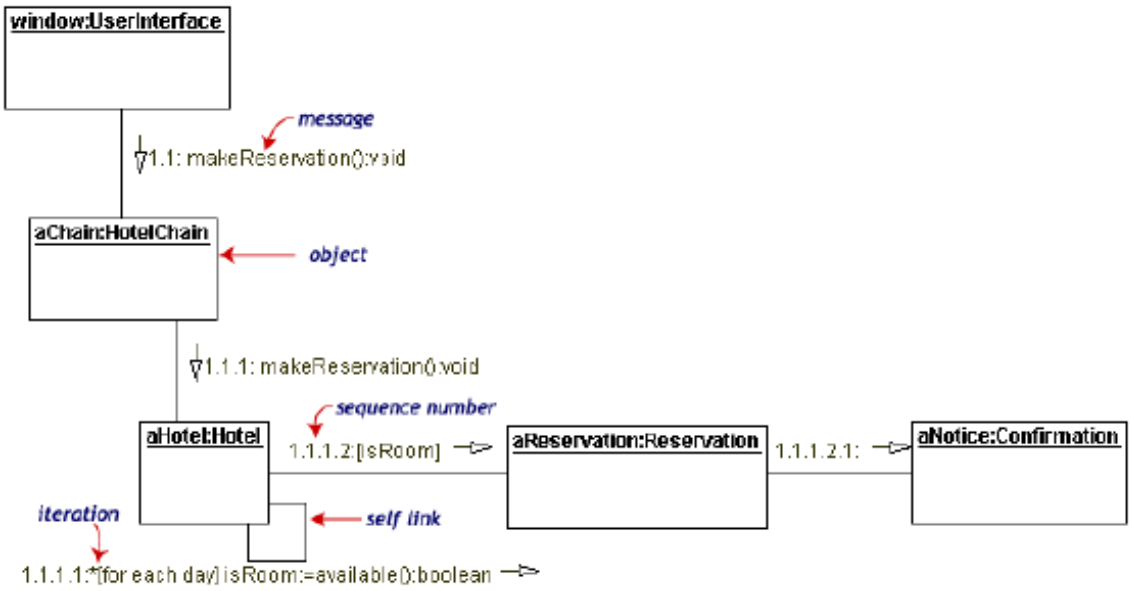
Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Adapun contoh dari sequence diagram dapat di lihat pada gambar 2.13 di bawah ini.



Gambar 2.13 Contoh Sequence diagram

6. Collaboration diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama. Contoh gambar dari Collaboration Diagram dapat di lihat pada gambar 2.14 di bawah ini.



Gambar 2.14 Contoh Collaboration diagram

Selain dari pada 6 diagram tersebut di atas terdapat juga *Component diagram* dan *Deployment diagram* yang menggambarkan hubungan antar komponen piranti perangkat lunak dan menggambarkan detail bagaimana dan di mana komponen tersebut di proses.