BAB II

TINJAUAN PUSTAKA

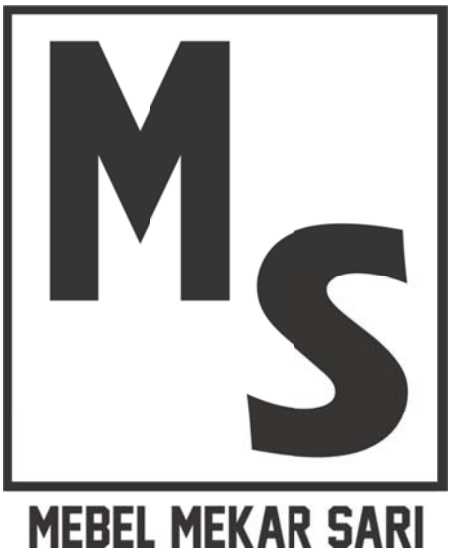
2.1 Tinjauan Perusahaan

2.1.1. Sejarah Instansi

Toko Mebel Mekar Sari didirikan pada tahun 2000, terletak di Jl.Raya kadungora No.202 A Garut. Tujuan utama didirikannya Toko Mebel Mekar Sari untuk menyediakan kebutuhan perlengkapan rumah tangga yang ada di garut.

Saat ini Toko Mebel Mekar Sari menpunyai 3 cabang yg terleteak tidak berjauhan dari toko pertama. Untuk kedepannya Toko Mebel Mekar Sari ingin menambah lebih banyak lagi cabang.

2. 1. 2. Logo Instansi

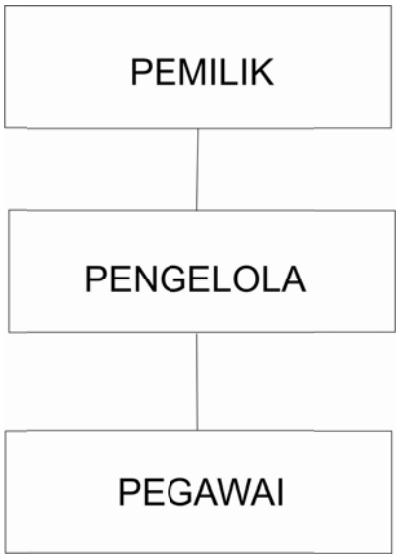


Gambar 2.1 Logo Toko Mebel Mekar Sari

2. 1. 3. Struktur Organisasi dan *Job Description*

Struktur Organisasi dan *Job Description* menjelaskan struktur organisasi yang ada di Toko Mebel Mekar Sari dan penjelasan pekerjaan yang dilakukan dari setiap bagiannya.

2. 1. 3. 1. Struktur Organisasi



Gambar 2.2 Struktur Organisasi Toko Mebel Mekar Sari

2. 1. 3. 2. *Job Description*

Adapun *Job Description* dari setiap bagian yaitu :

A. Pemilik memiliki tugas sebagai berikkut :
   1. Mengelola Toko Utama
   2. Melayani Setiap Pembeli
   3. Mencatat setiap transaksi penjualan dan pembelian barang di semua cabang

B. Pengelola memiliki tugas sebagai berikkut :
   1. Mengelola Toko Cabang
   2. Melayani Setiap Pembeli
   3. Mencatat setiap transaksi penjualan dan pembelian barang

C. Pegawaimemiliki tugas sebagai berikkut :
   1. Menjaga Toko
   2. Membersihkan Toko
   3. Mengantarkan Barang

2. 2. Landasan Teori

Landasan teori menjelaskan beberapa teori yang berkaitan dengan permasalahan yang dibahas sebagai dasar pemahaman dalam sebuah sistem serta metode yang dipakai untuk kegiatan pengembangan terhadap sistem sendiri.

2. 2. 1. Aplikasi

Perangkat lunak aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media.

Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut sebagai suatu paket atau suite aplikasi (*application suite*). Contohnya adalah Microsoft Office dan Open Office.org, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja, serta beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah.

2.2.2. OOP

2.2.2.1. Pengertian OOP

Object Oriented Programming (OOP) adalah suatu metode pemrograman yang berbasiskanpada objek, secara singkat pengertian dari OOP adalah koleksi objek yang saling berinteraksi dan saling memberikan informasi satu dengan yang lainnya.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

2.2.2. Konsep OOP

Konsep dasar dari Pemrograman Berorientasi Objek Pemrograman orientasi-objek menekankan konsep berikut:

1. Kelas - kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh 'class of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi object. Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.
2. Objek -membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
3. Abstraksi - Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.
4. Enkapsulasi - Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.
5. Polimorfisme melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di

mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas adminiistrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri

2. 2. 3. Basis Data

2. 2. 3. 1. Pengertian Basis Data

Secara umum, basis data dapat diartikan sebagai kumpulan data yang tersimpan dalam suatu tabel yang saling berelasi.

Adapun kegunaan basis data adalah :

1. Basis data merupakan komponen penting dalam sistem informasi, karena merupakan dasar dalam menyediakan informasi.
2. Menentukan kualitas informasi yang akurat, tepat pada waktunya, dan relevan, dimana informasi dikatakan bernilai jika manfaatnya lebih efektif dibandingkan dengan biaya untuk mendapatkannya.
3. Mengurangi duplikasi data (*data redudancy*).
4. Meningkatkan relasi antar data (*data relatability*).

5. Mengurangi pemborosan *external storage*.

Di dalam basis data terdapat beberapa istilah berikut yang juga dikenal sebagai urutan jenjang data :

1. *Characters*

   Merupakan bagian data yang terkecil yaitu berupa karakter numerik, huruf, maupun karakter spesial (simbol) yang membentuk item data/ *field*.

2. *Field*

   Merepresentasikan suatu atribut dari *record* yang menunjukan suatu item dari data, misalnya nama.

3. *Record*

   Merupakan kumpulan dari *field*, menggambarkan suatu unit data individu tertentu. Kumpulan *record* membentuk suatu *file*. Misalnya, *file* personalia, tiap-tiap *record* mewakili data tiap-tiap karyawan.

4. *File*

   File terdiri dari *record-record* yang menggambarkan satu kesatuan data yang sejenis. Misalnya, *file* mata kuliah berisi data tentang semua mata kuliah yang ada.

5. *Database*

   Merupakan kumpulan dari *file*/ tabel.

2. 2. 3. 2. Normalisasi Basis Data

Normalisasi merupakan sebuah teknik dalam logika desain sebuah basis data, yaitu mengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik (tanpa redudansi).

1. Normalisasi ke-1

   Suatu data dikatakan normal pertama jika sudah memiliki karakter berikut:

   a. Mendefinisikan atribut kunci (*key*).
   b. Tidak ada grup yang berulang.
   c. Semua atribut bukan *key* tergantung pada atribut kunci.

2. Normalisasi ke-2

   Suatu data dikatakan normal kedua jika sudah memiliki karakter berikut:

   a. Sudah memenuhi dalam bentuk normal pertama.
   b. Sudah tidak ada ketergantungan parsial, dimana seluruh *field* tergantung pada sebagian *key field.*

2. 2. 3. 3. ERD (*Entity Relationship Diagram*)

ERD merupakan notasi grafis dalam pemodelan data konseptual yang mendeskripsikan hubungan antara penyimpanan (tabel). ERD digunakan untuk memodelkan struktur data dan hubungan antar data, kita dapat menguji data dengan mengabaikan proses yang harus dilakukan serta bisa mengetahui data apa yang kita perlukan, serta bagaimana data-data tersebut bisa saling berhubungan. Pada dasarnya terdapat 3 macam simbol yang digunakan untuk menggambarkan struktur dan hubungan antar data, yaitu :

1. *Entity*

   Suatu objek, merupakan bagian dari sitem yang dapat diidentifikasi dalam lingkungan pemakai, sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat. *Entity* dapat berupa sebuah kelompok dari sesuatu, harus dapat dibedakan.

2. *Atribut*

   Merupakan elemen dari *entity*. Setiap *entity* memiliki atribut yang berfungsi untuk mendeskripsikan karakternya. Misalnya, *entity* mahasiswa memiliki atribut nim, nama, dan sebagainya.

3. *Relationship* (hubungan)

   Merupakan penghubung antar satu entitas dengan entitas yang lain, ataupun satu entitas ke dalam entitas itu sendiri. *Relationship* dibagi menjadi 3 jenis, yaitu :

   1) Satu ke satu (1-1)

      Hubungan relasi satu ke satu yaitu setiap entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B.

   2) Satu ke banyak/ banyak ke satu(1-n / n-1)

      Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi setiap entitas pada entitas B dapat berhubungan dengan satu entitas pada himpunan entitasA.

   3) Banyak ke banyak (n-n)

      Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B.

2. 2. 3. 4. Sistem Manajemen Basis Data

Sistem Manajemen Basis Data adalah perangkat lunak yang didesain untuk membantu dalam hal pemeliharaan dan utilitas kumpulan data dalam jumlah besar, serta menjadi alternatif penggunaan secara khusus untuk aplikasi, seperti penyimpanan data dalam *file* dan menulis kode aplikasi yang spesifik untuk pengaturannya. Adapun keuntungan dari DBMS (*Database Management System*) adalah :

1. *Data independence*, DBMS dapat mengurangi kebergantungan antar data.
2. *Efficient data access*, DBMS dapat menimpa data dan *meretrive* data dengan efisien. Apabila suatu saat data tersebut dibutuhkan, maka efektif dalam pemanggilannya.
3. *Data integrity and security*, saat data diakses menggunkan DBMS, DBMS mengintergrasi *constraints* dari tiap data tersebut.
4. *Data administration*, Administrasi data digunakan untuk mengatur hal-hal yang berhubungan dengan distribusi data, hak akses user yang mendistribusikan atau menggunakan data tersebut, mengorganisasikan data untuk mengurangi redudansi data dan *fine-tuning* penyimpanan data sehingga *retrive* data berjalan dengan efisien.
5. *Concurrent access and crash recovery*, DBMS dapat menjadwalkan *concurrent access* (pengaksesan secara bersamaan) terhadap suatu data sehingga menghindari dari *crash*.
6. *Reduced application development time*, DBMS mendukung banyak fungsi penting yang diakses oleh banyak aplikasi yang tersimpan dalam media penyimpanan.

2. 2. 3. 5. Sistem Manajemen Basis Data Relasional

RDBMS (*Relational Database Management System*) merupakan sekumpulan data yang saling beruhubungan yang disimpan sedemikian rupa sehingga mudah diambil informasinya bagi pengguna. Ada tiga prinsip dalam RDBMS :

1. *Data Definition*

   Mendefinisikan data yang akan dibuat, seperti nama tabel dan *field* serta keterangan mengenai *field* tersebut.
2. *Data Manipulation*

   Memanipulasi data yang sudah dibuat, seperti mengubah nama *field* dan keterangannya, menghapus *record*.
3. *Data Control*

   Bagian ini berkenaan pada mengendalikan data kepada siapa saja yang bisa melihat isi data.

2.2.4. Microsoft Accsess 2010

Microsoft Access 2010 digunakan kebanyakan oleh bisnis-bisnis kecil dan menengah, di dalam sebuah organisasi yang kecil bahkan mungkin juga digunakan oleh perusahaan yang cukup besar, dan juga para programmer untuk membuat sebuah sistem buatan sendiri untuk menangani pembuatan dan manipulasi data. Access juga dapat digunakan sebagai sebuah basis data untuk aplikasi Web dasar yang disimpan di dalam server yang menjalankan Microsoft Internet Information Services (IIS) dan menggunakan Microsoft Active Server Pages (ASP). Meskipun demikian, penggunaan Access kurang disarankan, mengingat telah ada Microsoft SQL Server yang memiliki kemampuan yang lebih tinggi. Beberapa pengembang aplikasi profesional menggunakan Microsoft Access untuk mengembangkan aplikasi secara cepat (digunakan sebagai Rapid Application Development/RAD tool), khususnya untuk pembuatan purwarupa untuk sebuah program yang lebih besar dan aplikasi yang berdiri sendiri untuk para salesman. Microsoft Access kurang begitu bagus jika diakses melalui jaringan sehingga aplikasi-aplikasi yang digunakan oleh banyak pengguna cenderung menggunakan solusi sistem manajemen basis data yang bersifat klien/server. Meskipun demikian, tampilan muka Access (form, report, query, dan kode Visual Basic) yang dimilikinya dapat digunakan untuk menangani basis data yang sebenarnya diproses oleh sistem manajemen basis data lainnya, seperti halnya Microsoft Jet Database Engine (yang secara default digunakan oleh Microsoft Access), Microsoft SQL Server, Oracle Database, dan beberapa produk lainnya yang mendukung ODBC.

Microsoft mengungkapkan bahwa pada Office 2010 ini akan lebih ringan dari versi sebelumnya. Pada Microsoft Office 2010, Microsoft Office akan mendukung dua versi yang terpisah, yaitu versi 32 bit dan versi 64 bit dan Office 2010 akan menjadi yang pertama yang melakukan dua versi yang berbeda, di kedua versi yang berbeda. Tak banyak disinggung keunggulan Microsoft Office 2010.banyak kelebihan-kelebihan yang ditawarkan oleh office 2010 ini, diantaranya adalah :

1. LEBIH RINGAN.

   Saat pertama kali dijalankan Microsoft Office 2010 hanya membutuhkan 2 detik untuk masuk ke lembar kerja. Sedangkan untuk penggunaan ke-2 dan seterusnya, Microsoft Office 2010 hanya membutuhkan sepersekian detik untuk load ke lembar kerja. Wow!

2. MENU FILE HADIR KEMBALI.

   Saat masuk ke lembar kerja kita tidak akan menemukan lagi The Office 2007 Button, image berlogo Windows layaknya jantung pada Office 2007. Fungsi ini telah

digantikan dengan menu File, dengan tampilan penuh (full-window) bahkan fitur dan fungsi didalamnya jauh lebih beragam.

3. TAMPILAN RIBBON SERAGAM

   Ribbon atau pita menu kini juga dapat ditemukan pada aplikasi Office lainnya, yakni Outlook, Visio, OneNote, Project, dan Publisher.

4. SMART-ART DAN TEMPLATE BARU.

   Office 2010 memberikan SmartArt dan Template baru plus koleksi sebelumnya dari Office 2007.

5. SAVE AS LEBIH VARIATIF.

   Kejutan baru yang ditawarkan oleh Office 2010 adalah fungsi menyimpan dokumen menjadi format PDF, XPS (portable online), ODT (OpenOffice) secara langsung. Format PDF dan XPS awalnya hanya dapat difungsikan dengan add-ons pada Office 2007. Sedangkan ODT merupakan format atau ekstensi dokumen pada OpenOffice, ini bukti universalitas dan support Microsoft Office bahkan terhadap open source.

6. PRINT FUNCTION INTEGRATED

   Fungsi print preview dan print telah terintegrasi dengan full-window pada menu File pada offce 2010 yang sangat memudahkan kita.

7. FUNGSI JUMP-LISTS.

   Dengan fungsi jump-lists kita dapat langsung menuju dokumen yang sebelumnya pernah kita buka atau jalankan melalui StartMenu. Bisa dibilang layaknya RecentDocuments tetapi terbagi atas tiap aplikasi Office tersendiri dan hanya Anda dapatkan pada Windows 7 (Seven).

8. FITUR THUMBNAIL.

   Microsoft Office 2010 meniru fitur thumbnail atau preview layaknya Adobe Reader untuk file PDF. Dengan begitu kita dapat mengintip tampilan atau isi file dokumen (.DOCX) tanpa harus membukanya terlebih dahulu.

Microsoft Office 2010 terdiri dari Microsoft Office Word 2010, Microsoft Office Excel 2010, Microsoft Office PowerPoint 2010, Microsoft Office Outlook 2010,Microsoft Office OneNote 2010, Microsoft Office Access 2010, Microsoft Office Publisher 2010, Microsoft Office InfoPath 2010, SharePoint Workspace, Communicator.

2.2.5. MICROSOFT VISUAL STUDIO

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (suite) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu 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 dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam 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).

Visual Studio kini telah menginjak versi Visual Studio 9.0.21022.08, atau dikenal dengan sebutan Microsoft Visual Studio 2008 yang diluncurkan pada 19 November 2007, yang ditujukan untuk platform Microsoft .NET Framework 3.5. Versi sebelumnya, 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 memang membutuhkan Microsoft .NET Framework. Sementara itu, sebelum muncul Visual Studio .NET, terdapat Microsoft Visual Studio 6.0 (VS1998).

2. 2. 6. UML

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem blue print, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database*, dan komponen-komponen yang diperlukan dalam sistem *software*.

2. 2. 6. 1. Artifact UML

UML menyediakan beberapa notasi dan *artifact* standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. *Artifact* didalam UML didefinisikan sebagai informasi dalam bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat. Contohnya adalah source code yang dihasilkan oleh proses pemrograman.

Yang harus diperhatikan untuk menjaga konsistensi antar artifact selama proses analisis dan desain adalah bahwa setiap perubahan yang terjadi pada satu *artifact* harus juga dilakukan pada *atifact* sebelumnya.

Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut :

1. *Use Case Diagram*
2. *Class Diagram*
3. *Statechart Diagram*
4. *Activity Diagram*
5. *Sequence Diagram*
6. *Collaboration Diagram*
7. *Component Diagram*
8. *Deployment Diagram*

Dibuatnya berbagai jenis diagram diatas karena :

a. Setiap sistem yang kompleks selalu paling baik jika didekati melalui himpunan berbagai sudut pandang yang kecil yang satu sama lain hampir saling bebas (*independent*). Sudut pandang tunggal senantiasa tidak mencukupi untuk melihat sistem yang besar dan kompleks.
b. Diagram yang berbeda-beda tersebut dapat menyatakan tingkatan yang berbeda-beda dalam proses rekayasa.
c. Diagram-diagram tersebut dibuat agar model yang dibuat semakin mendekati realitas

2.2.6.2. Tujuan UML

Tujuan utama UML diantaranya untuk :

1. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.

3. Menyatukan praktek-praktek terbaik yang terdapat dalam bahasa pemodelan.

2.2.6.3. Notasi dalam UML

1. *Actor*

   Actor menggambarkan segala pengguna software aplikasi (*user*). *Actor* memberikan suatu gambaran jelas tentang apa yang harus dikerjakan *software* aplikasi. Sebagai contoh sebuah actor dapat memberikan input kedalam dan menerima informasi dari *software* aplikasi, perlu dicatat bahwa sebuah actor berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. Sebuah *actor* mungkin seorang manusia, satu *device, hardware* atau sistem informasi lainnya.

2. *Use Case*

   Use case menjelaskan urutan kegiatan yang dilakukan *actor* dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, namun *use case* hanya menjelaskan apa yang dilakukan oleh *actor* dan sistem bukan bagaimana *actor* dan sistem melakukan kegiatan tersebut.

   a. *Use-case* Konkret adalah *use case* yang dibuat langsung karena keperluan actor. Actor dapat melihat dan berinisiatif terhadapnya
   b. *Use-case* Abstrak adalah *use case* yang tidak pernah berdiri sendiri. *Use case* abstrak senantiasa termasuk didalam (*include*), diperluas dari (*extend*) atau memperumum (*generalize*) *use case* lainnya.

   Untuk menggambarkannya dalam *use case* model biasanya digunakan *association relationship* yang memiliki *stereotype include, extend* atau *generalization relationship*. Hubungan *include* menggambarkan bahwa suatu *use case* seluruhnya meliputi fungsionalitas dari *use case* lainnya. Hubungan *extend* antar *use case* berarti bahwa satu *use case* merupakan tambahan fungsionalitas dari *use case* yang lain jika kondisi atau syarat tertentu terpenuhi.

3. *Class*

   Class merupakan pembentuk utama dari sistem berorientasi *obyek,* karena *class* menunjukkan kumpulan *obyek* yang memiliki atribut dan operasi yang sama. *Class* digunakan untuk mengimplementasikan *interface.*

   Class digunakan untuk mengabstraksikan elemen-elemen dari sistem yang sedang dibangun. *Class* bisa merepresentasikan baik perangkat lunak maupun perangkat keras, baik konsep maupun benda nyata.

Notasi *class* berbentuk persegi panjang berisi 3 bagian: persegi panjang paling atas untuk nama *class*, persegi panjang paling bawah untuk operasi, dan persegi panjang ditengah untuk atribut.

Atribut digunakan untuk menyimpan informasi. Nama atribut menggunakan kata benda yang bisa dengan jelas merepresentasikan informasi yang tersimpan didalamnya. Operasi menunjukkan sesuatu yang bisa dilakukan oleh *obyek* dan menggunakan kata kerja.

4. *Interface*

Interface merupakan kumpulan operasi tanpa implementasi dari suatu *class*. Implementasi operasi

5. *Interaction*

Interaction digunakan untuk menunjukkan baik aliran pesan atau informasi antar *obyek* maupun hubungan antar *obyek*. Biasanya *interaction* ini dilengkapi juga dengan teks bernama *operation signature* yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan.

6. *Note*

Note digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa disertakan ke semua elemen notasi yang lain.

7. *Depedency*

Dependency merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada dibagian tanpa tanda panah. Terdapat 2 *stereotype* dari *dependency*, yaitu *include* dan *extend*. *Include* menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah). *Extend* menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan kedalam elemen yang ada di garis dengan panah.

8. *Association*

Association menggambarkan navigasi antar *class* (*navigation*), berapa banyak obyek lain yang bisa berhubungan dengan satu obyek (*multiplicity* antar *class*) dan apakah suatu *class* menjadi bagian dari *class* lainnya (*aggregation*). *Navigation* dilambangkan dengan penambahan tanda panah di akhir garis. *Bidirectional navigation* menunjukkan bahwa dengan mengetahui salah satu *class* bisa didapatkan

informasi dari *class* lainnya. Sementara *Uni Directional navigation* hanya dengan mengetahui *class* diujung garis association tanpa panah kita bisa mendapatkan informasi dari *class* di ujung dengan panah, tetapi tidak sebaliknya.

Aggregation mengacu pada hubungan "*has-a*", yaitu bahwa suatu *class* memiliki *class* lain, misalnya Rumah memiliki *class* Kamar.

9. *Generalization*

   Generalization menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan *generalization*, *class* yang lebih spesifik (*subclass*) akan menurunkan atribut dan operasi dari *class* yang lebih umum (*superclass*) atau "*subclass is superclass*". Dengan menggunakan notasi *generalization* ini, konsep *inheritance* dari prinsip hirarki dapat dimodelkan.

10. *Realization*

    Realization menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah. Misalnya *class* merealisasikan *package*, component merealisasikan *class* atau *interface.*