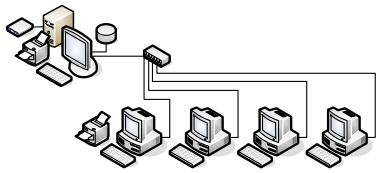
BAB II

LANDASAN TEORI

2.1 Client server

Arsitektur jaringan *client server* merupakan pengembangan dari arsitektur *file server*. Arsitektur ini adalah model konektivitas pada jaringan yang mengenal adanya *server* dan *client*, dimana masing-masing memiliki fungsi yang berbeda satu sama lain. *Server* dapat berbagi pakai data, aplikasi dan *peripheral* seperti *harddisk*, *printer*, *modem* dan lain-lain. Oleh karena itu, tidak jarang juga tercipta sebutan *print server*, *communication server* dan lain sebagainnya. Prinsip kerjanya sangat sederhana, dimana *server* akan menunggu permintaan dari *client*, memproses dan memberikan hasilnya kepada *client*. Sedangkan *client* akan mengirimkan permintaan ke *server*, menunggu proses dan melihat visualisasi hasil prosesnya.

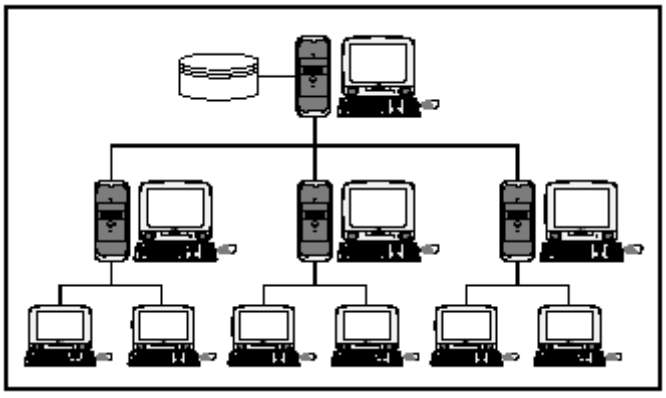


Gambar 2.1 Konektivitas *Client Server*

Sistem *client server* ini menggunakan protocol TCP/IP (*Transmission Control Protocol/Internet Protocol*). Unix dan Windows NT merupakan contoh yang baik dari sistem operasi jaringan *client server*.

2.1.1 Sistem *Client Server*

Sistem *Client* dan *Server* terdiri atas dua komponen (mesin) utama, yaitu Client dan Server. Client berisi aplikasi basis data dan server berisi DBMS dan basis data. Setiap aktifitas yang dikehendaki para pemakai akan lebih dahulu ditangani oleh client. Client menangani proses yang menjadi tanggung jawabnya. Jika ada proses yang harus melibatkan data yang tersimpan pada basis data yang terletak di server, barulah client mengadakan hubungan denga server. Pada bentuk sistem client server untuk memenuhi kebutuhan client akan megirimkan pesan atau perintah *Query* pengambilan data. Selanjutnya server yang menerima pesan tersebut akan menjalankan *Query t*ersebut dan hasilnya akan dikirimkan kembali ke client. Dengan begitu, transfer datanya jauh lebih efisien. Untuk lebih jelasnya, dapat dilihat pada gambar sistem client server berikut ini :



Gambar 2.2 Sistem *Client-Server* Kompleks
Sumber : http://images?hl=id&q=client+server&um=1&ie=UTF-8&ei=ZQRTSrqpH4uCswP1pID-Bg&sa=X&oi

2.1.2 Komponen dasar Client Server

1. Client

Client merupakan terminal yang digunakan oleh pengguna untuk meminta layanan tertentu yang dibutuhkan. Terminal client dapat berupa PC, ponsel, komunikator, robot, televisi dan peralatan lain yang membutuhkan informasi.

2. Middleware

Midleware merupakan komponen perantara yang memungkinkan client dan server untuk saling terhubung dan berkomunikasi satu sama lain. *Midleware* ini dapat berupa *Transaction Monitor* /TP. *Remote Procedure Call* atau *Object Request Broker*/ORB.

3. Server

Server merupakan komputer khusus yang bertugas melayani aplikasi-palikasi jaringan / pihak yang menyediakan layanan. Server ini akan dapat berupa basis data SQL, Monitor TP, server *groupware*, server objek dan web. Secara umum, server berperan menerima pesan permintaan layanan dari client, memproses permintaan tersebut dan mengirimkan hasil permintaan kepada client.

2.1.3 Karakteristik Server dan Client

1. Karakteristik Server

a. Pasif

b. Menunggu request

c. Menerima request, memproses mereka dan mengirimkan balasan berupa service

2. Karakteristik Client

a. Aktif

b. Mengirim request

c. Menunggu dan menerima balasan dari server

2.1.4 Ciri-ciri Client Server

Beberapa ciri dari arsitektur sistem terdistribusi Client Server diantaranya :

1. Berbasis layanan

   Server memberikan sejumlah layanan yang dibutuhkan dan diminta oleh client, antara lain : berbagai pakai berkas, dan peralatan pendukung.

2. Sumber daya yang digunakan bersama

   Server mengelola sejumlah sumber daya yang dimiliki agar dapat diakses dan digunakan secara bersama-sama oleh terminal-terminal client yang terhubung pada server.

3. Hubungan dan interaksi Client Server

   Hubungan yang terjadi antara server dan client adalah *one-to many*, yang berarti bahwa satu server melayani banyak client. Client selalu memulai transaksi dengan meminta layanan sedangkan server menanti permintaan layanan secara pasif.

4. Client tidak perlu mengetahui lokasi fisik server

   Server dapat terletak di berbagai tempat yang belum tentu diketahui oleh client, Walaupun demikian client tetap dapat mengakses server untuk mendapatkan layanan sesuai kebutuhannya.

5. Interoperabilitas perangkat lunak dan perangkat keras

   Perangkat lunak dan keras yang digunakan oleh masing-masing client tidak harus sama dengan yang digunakan pada server, namun masih dapat saling terkoneksi antara satu dan yang lain.

6. Pertukaran berbasis pesan

   Mekanisme dari Client Server berdasar pada pertukuran pesan. Pesan yang dipertukarkan adalah permintaan layanan dan umpan balik dari permintaan layanan tersebut.

7. Enkapsulasi layanan

   Client tidak perlu mengetahui Sistem Operasi pengelolaan permintaan yang terjadi dalam server sehingga client tidak dapat mengontrol Sistem Operasi pengelolaan permintaan.

8. Skalabilitas

   Skalabilitas adalah kemampuan untuk diperbesar atau diperkecil. Ukuran sistem Client Server dapat diubah secara horizontal maupun vertikal. Perubahan vertikal berarti berpindah ke server lebih besar atau lebih cepat atau mendistribusikan tugas melayani client ke beberapa server. Pengubahan horizontal berarti menambah atau mengurangi jumlah client.

9. Konsistensi data

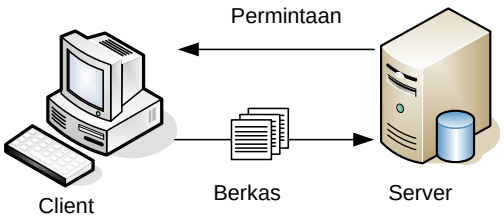
   Data hanya dikelola pada server pusat sehingga konsistensi dan data lebih terjamin dan biaya pemeliharaan pun menjadi lebih murah.

2.1.5 Tipe Jaringan Client Server

Berdasarkan tipe layanan yang diberikan server kepada client, jaringan Client Server dapat dibagi menjadi kedalam banyak tipe, tipe-tipe tersebut antara lain :

1. Server Berkas

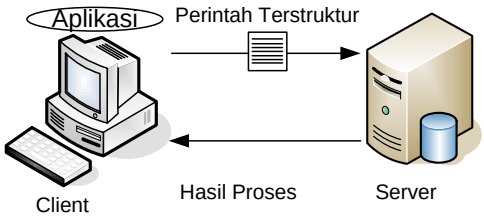
   Sistem jaringan berkas adalah sistem jaringan yang dimana layanan yang diberikan server berupa berkas, baik berkas aplikasi seperti aplikasi pengolahan kata, pengolahan angka, pengolahan data, pengolahan gambar dan lain sebagainya, maupun berkas yang dihasilkan oleh aplikasi tersebut, seperti dokumen pengolahan kata, tabel-tabel pengolahan angka, berkas presentasi dan lain sebagainya.



Gambar 2.3 *Server* Berkas

2. *Server* Basis Data

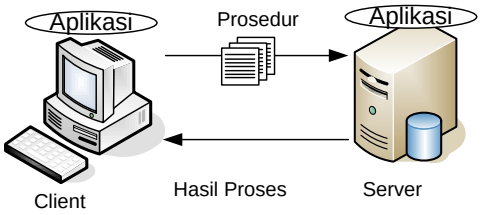
   Sistem jaringan *server* basis data adalah merupakan sistem jaringan dimana layanan yang diberikan oleh server berupa pengolahan dan penyajian data berdasarkan perintah terstruktur (*query*) yang diberikan client. Pada jaringan ini, server menyimpan berbagai macam data yang dapat diakses oleh pengguna melalui terminal-terminal client.



Gambar 2.4 *Server* Basis Data

3. *Server* Transaksi

Sistem jaringan *serve*r transaksi adalah sistem jaringan dimana layanan yang diberikan *server* berupa hasil Sistem Operasi dari sekelompok perintah terstruktur yang diberikan client. Jaringan ini pada dasarnya hampir sama dengan sistem jaringan basis data sebelumnya. Perbedaan terletak pada *server* transaksi yang memproses sekelompok perintah terstruktur dari client, dan sekelompok perintah terstruktur ini disebut prosedur.



Gambar 2.5 *Server* Transaksi

4. Groupware *Server*

Sistem jaringan *groupware* adalah sistem jaringan dimana layanan yang diberikan *server* berupa fasilitas pemakaian bernama informasi semi terstruktur diantara pengguna jaringan. Pada jaringan ini, server menyimpan, mengelola dan menyebarkan informasi antar pengguna dalam jaringan, misalnya teks, gambar, surat dan ruang diskusi.

5. Server Objek

Sistem jaringan server objek adalah sistem jaringan dimana layanan yang diberikan server berbentuk objek. Dalam jaringan ini, client dan server berkomunikasi melalui objek-objek yang miliki client dan server.

6. Web Server

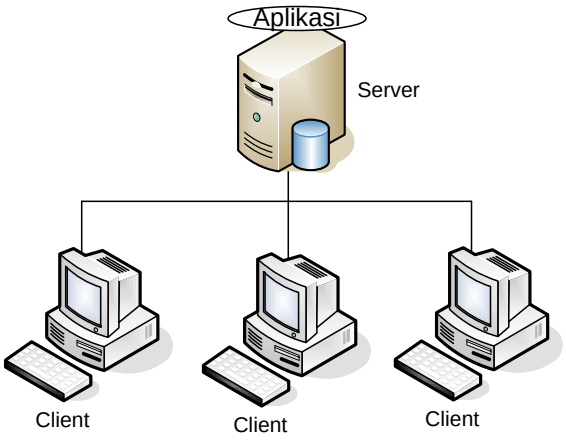
Sistem jaringan web server adalah sistem jaringan dimana layanan yang diberikan server berupa pengelolaan dan pemakaian bersama dokumen-dokumen yang saling terhubung. Jaringan ini merupakan jaringan yang memungkinkan tiap dokumen dalam jaringan memiliki hubungan ke dokumen lain sehingga dokumen-dokumen dalam jaringan terhubung satu dengan yang lain, semacam jaringan laba-laba.

2.1.6 Arsitektur Client Server

1. Two Tier

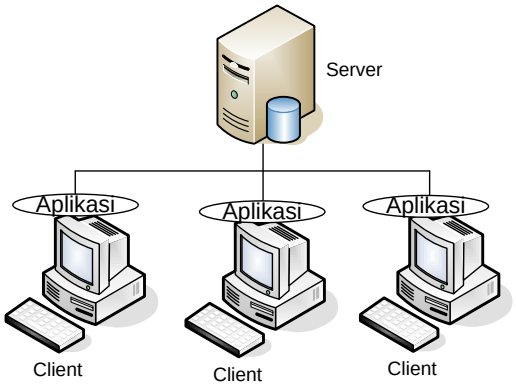
Arsitektur *two tier* merupakan arsitektur yang disebut Client Server dimana terdapat komputer sebagai client dan server yang berinteraksi melalui protokol dan media komunikasi tertentu Model arsitektur *Two Tier* dikelompokan menjadi dua macam yaitu

a. Thin Client – Thick Server



Gambar 2.6 *Thin Client – Thick Server*

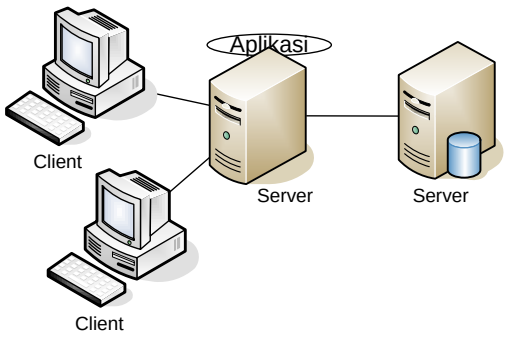
b. Thick Client – Thin Server



Gambar 2.7 *Thick Client – Thin Server*

2. Three Tier

Arsitektur Client Server ini memisahkan antara data (*Data Management Tier*), aplikasi (*Middle Tier*) dan penyajian (*Presentation Layer*) sebagaimana ditampilkan pada gambar dibawah ini :



Gambar 2.8 *Arsitektur Three Tier*

a. Data Management Tier

Merupakan komputer server yang dikhususkan untuk menangani pengolahan basis data.

b. Middle Tier

Merupakan komputer server yang dikhususkan untuk menangani aplikasi-aplikasi dimana prosedur-prosedur dan perhitungan-perhitungan yang kompleks dilakukan di komputer.

c. Presentation Layer

Merupakan komputer client yang menjadi *interface* bagi pengguna untuk memasukkan data, mengajuan permintaan layanan kepada server dan melihat hasilnya.

3. n-Tier

Istilah *n-tier* menunjukan lapisan yang ada dalam sebuah aplikasi. Sebuah aplikasi terdiri dari beberapa komponen utama, yaitu lapisan *Presentation* (*Presentation Layer*), lapisan *applica*

tion (*Application Layer*) dan logika bisnis (*bussiness logic layer*) dan lapisan data (*Data layer*).

Lapisan *Presentation* menghubungkan antarmuka dengan pengguna aplikasi, dapat berupa model grafis atau berupa teks. Pengguna dapat berinteraksi dengan aplikasi tersebut menggunakan lapisan *presentation* ini. Lapisan *Application* berisi inti dari aplikasi dan lapisan data yang digunakan oleh aplikasi tersebut. Lapisan data dapat berbentuk satu atau lebih server basis data yang lokasinya tersebar dibeberapa tempat.

2.2 *Unified Modelling Language* (UML)

Unified Modelling Language (UML) adalah bahasa standart untuk melakukan spesifikasi visualisasi, konstruksi, dan dokumentasi dari komponen-komponen perangkat lunak, dan digunakan untuk pemodelan bisnis. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat 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 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

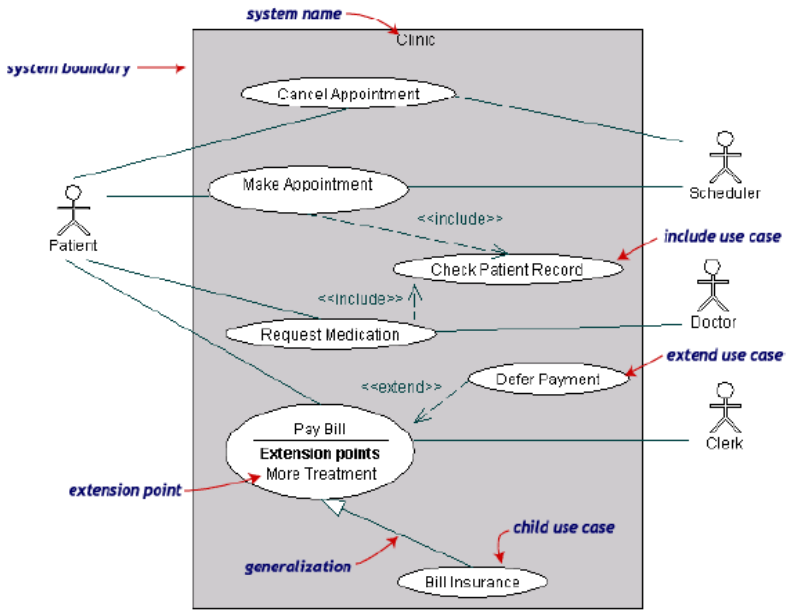
memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari tiga notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

2.2.1 Konsepsi Dasar UML

Abstraksi konsep dasar UML yang terdiri dari *structural classification*, *dynamic behavior*, dan *model management*, bisa kita pahami dengan mudah apabila kita melihat gambar diatas dari *Diagrams*. *Main concepts* bisa kita pandang sebagai *term* yang akan muncul pada saat kita membuat diagram. Dan *view* adalah kategori dari diagaram tersebut. Dalam UML mendefinisikan berbagai macam diagram, yaitu sebagai berikut :

a. 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. Berikut contoh dari *use case diagram :*



Gambar 2.9 *Use Case Diagram*

b. Class Diagram

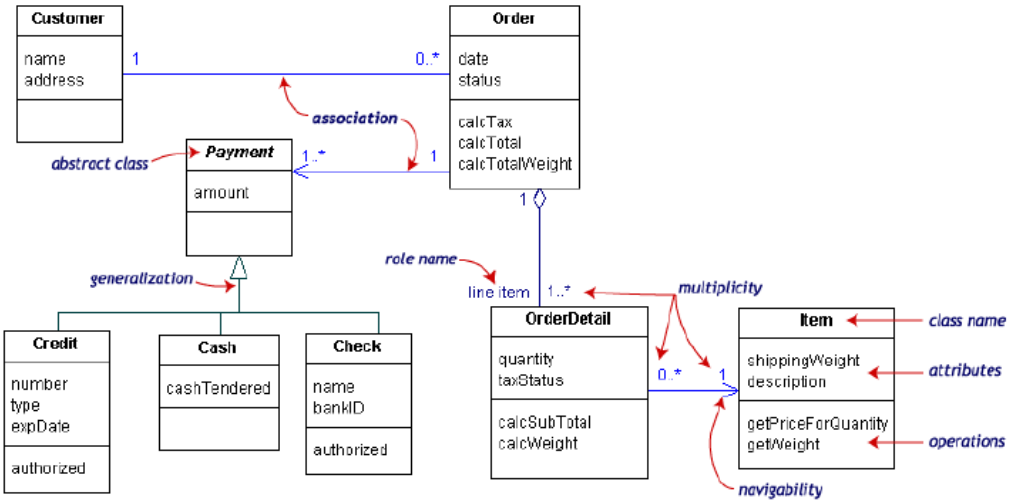
Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class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

a. Nama (dan *stereotype*)

b. Atribut

c. Metode

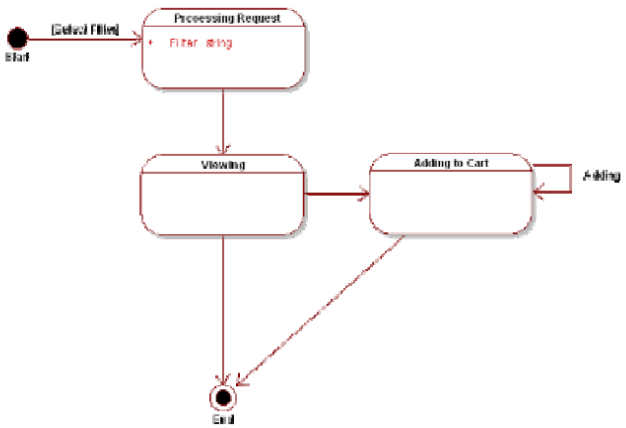
Berikut ini contoh gambar *class diagram* :



Gambar 2.10 *Class Diagram*

b. Statechart Diagram

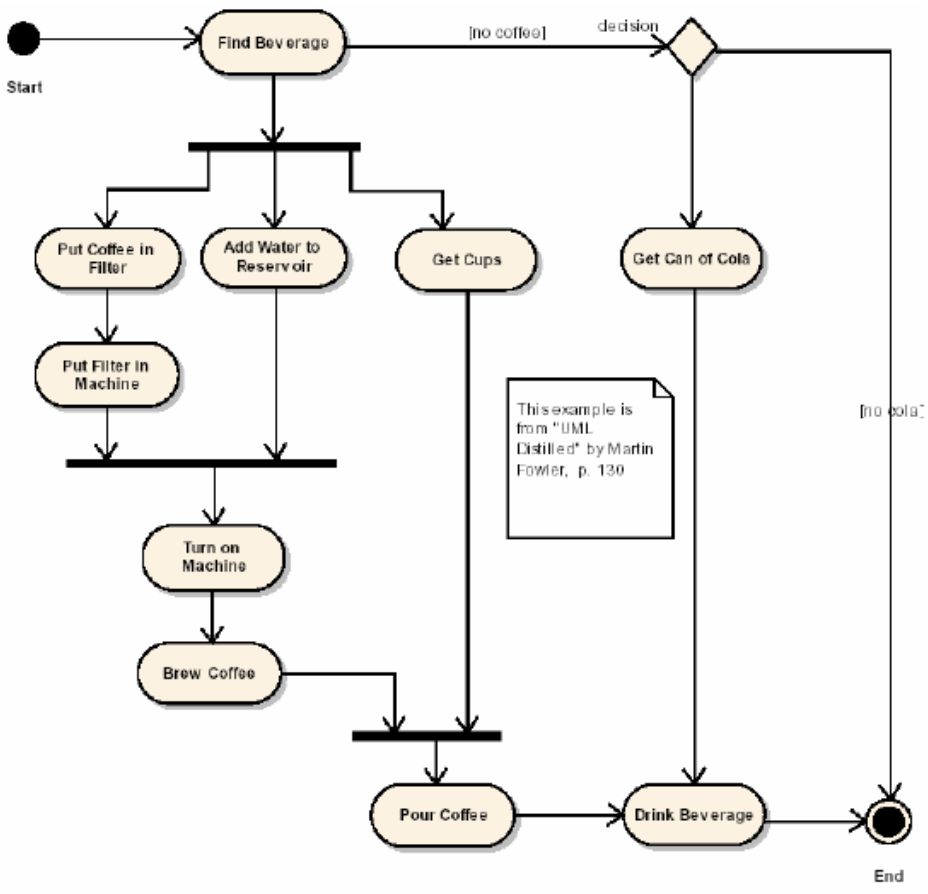
Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimulus* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*). Berikut ini contoh gambar dari *statechart diagram* :



Gambar 2.11 *Statechart Diagram*

c. Activity Diagram

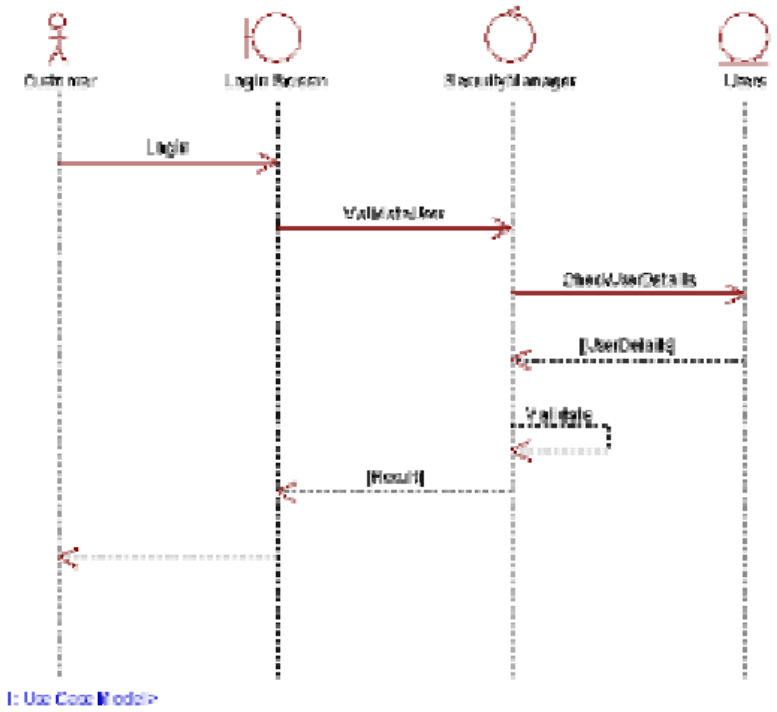
Activity diagram merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). *Activity diagrams* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan Sistem Operasi paralel yang mungkin terjadi pada beberapa eksekusi. Contoh *activity diagram* tanpa *swimlane*:



Gambar 2.12 *Activity Diagram*

d. Sequence Diagram

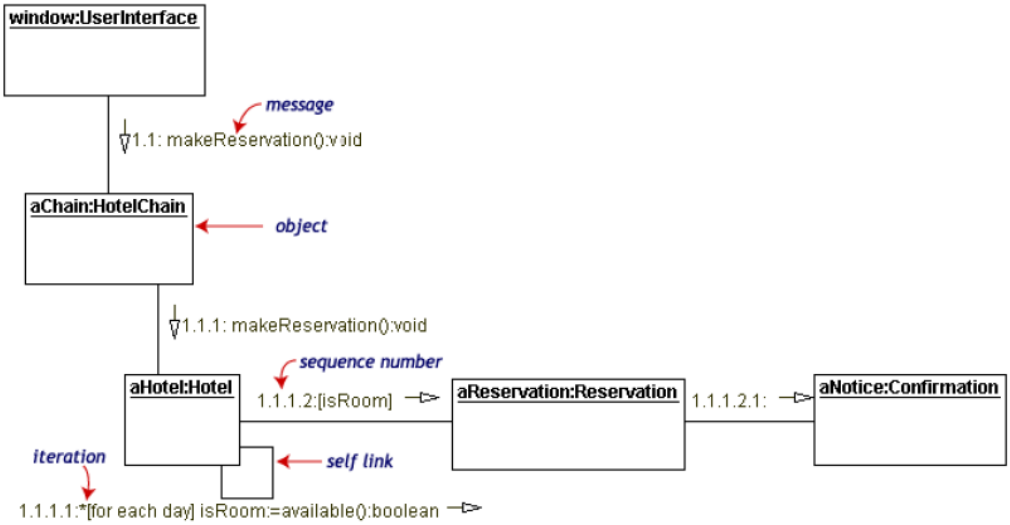
Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Contoh *sequence diagram*



Gambar 2.13 *Sequence Diagram*

e. 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.



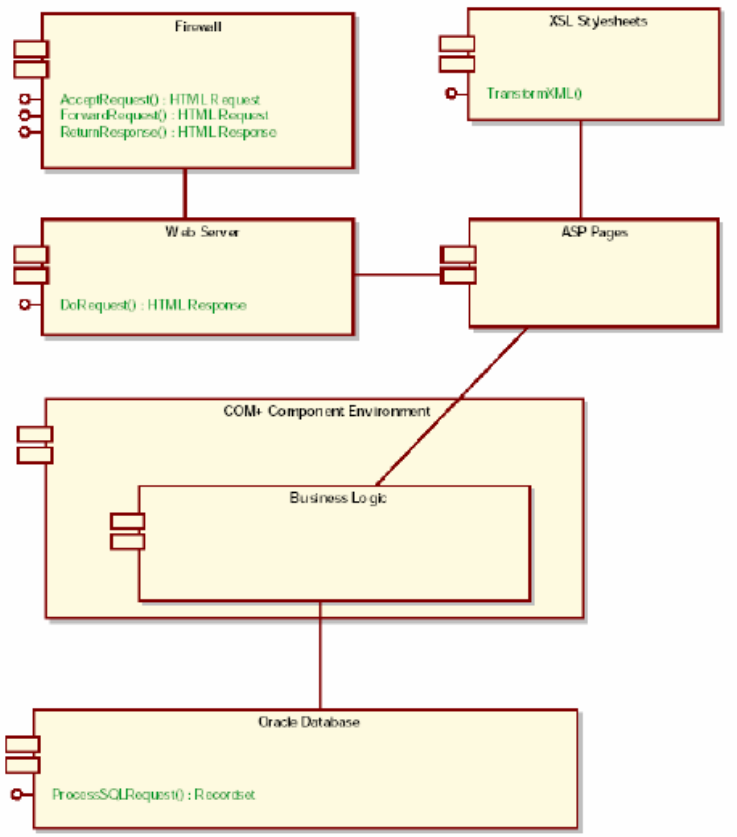
Gambar 2.14 *Collaboration Diagram*

f. Component Diagram

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

Contoh *component diagram*:

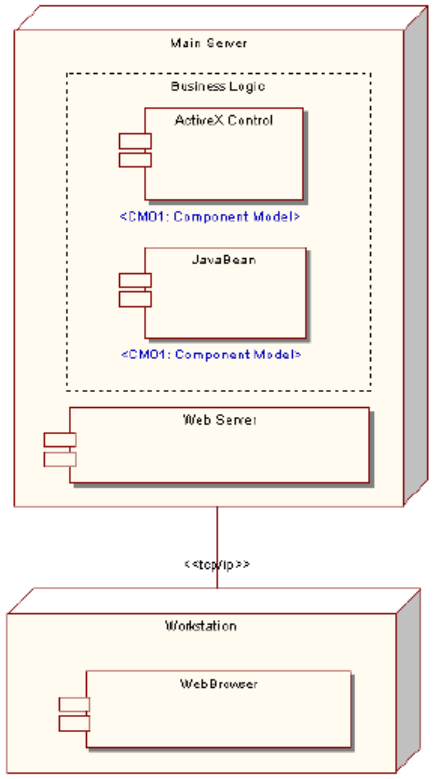


Gambar 2.15 *Component Diagram*

g. Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Sebuah *node* adalah server, *workstation*, atau

piranti 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.



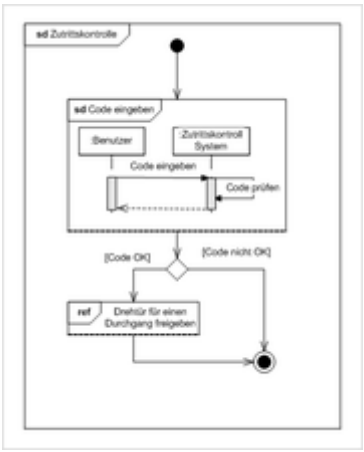
Gambar 2.16 *Deploy Diagram*

h. Interaction overview diagram

Interaction Overview Diagram dalam *Unified Modeling Language* (UML) adalah jenis kegiatan diagram yang mewakili simpul diagram interaksi. Mereka adalah mekanisme penataan tingkat tinggi untuk urutan diagram. Diagram menggambarkan interaksi ikhtisar ikhtisar aliran kontrol di mana setiap node dapat berupa diagram interaksi. *Interaction Overview Diagram* menggabungkan aktivitas diagram dan *in-line* urutan diagram. Diagram interaksi ini dapat mencakup urutan, komunikasi, interaksi

ikhtisar dan *timing diagram.* Dalam diagram interaksi node urutan diagram, dan ujung-ujungnya menunjukkan urutan interaksi ini terjadi. Anda juga dapat menggunakan diagram gambaran interaksi membongkar skenario yang rumit yang kalau tidak akan membutuhkan banyak jika-maka-lain jalan yang akan digambarkan sebagai sebuah diagram sekuens.

Sebagian besar notasi untuk interaksi overview diagram untuk kegiatan yang sama diagram. Sebagai contoh, awal, akhir, keputusan, menggabungkan, garpu dan bergabung node semua sama. Namun, overview diagram interaksi memperkenalkan dua elemen baru: interaksi kejadian-kejadian dan unsur-unsur interaksi.



Gambar 2.17 ***Interaction overview diagram***

2.3 Pemrograman Berorientasi Objek (PBO)

Pemrograman berorientasi objek (*object-oriented programming* disingkat OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Model data berorientasi objek

dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar.

2.3.1 Pengertian Objek, State, Behaviour

Objek bisa kita definisikan sebagai segala sesuatu yang memiliki state (keadaan) dan behavior (kelakuan/ tingkah laku). Setiap objek memiliki atribut sebagai status yang kemudian akan disebut sebagai *state*. Setiap objek memiliki tingkah laku yang kemudian akan disebut sebagai *behaviour*.

2.3.2 Pemrograman orientasi-objek menekankan konsep berikut:

2.3.2.1 Kelas

Kelas merupakan prototipe yang medefinisikan variabel-variabel dan method-method secara umum. Kelas adalah kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. 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.

2.3.2.1 Objek

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.

2.3.2.1 Abstraksi

Abstraksi adalah suatu cara melihat suatu objek dalam bentuk yang sederhana. Abstraksi yaitu 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.

2.3.2.4 Enkapsulasi

Enkapsulasi yaitu pembungkusan variabel dan method dalam sebuah objek dalam bagian yang terlindungi. Enkapsulasi dapat diartikan sebagai bungkusan (*wrapper*) pelindung program dan data yang sedang diolah. Pembungkusan ini medefinisikan perilaku dan melindungi program

dan data yang sedang diolah agar diakses sembarangan oleh program lain. Manfaat dari proses enkapsulasi adalah Modularitas.

2.3.2.5 ***Inheritance***

Inheritance (pewarisan) merupakan pewarisan atribut dan methode pada sebuah kelas yang diperoleh dari kelas yang telah terdefinisi. Setiap subclass akan mewarisi *state* (variabel-variabel) dan *behaviour* (*method-method*) dari superkelas-nya. Super-kelas, digunakan untuk menunjukkan hirarki class yang berarti kelas dasar dari sub-kelas/kelas. Sub-kelas adalah kelas anak atau turunan secara hirarki dari superkelas. Mengatur *polimorfisme* dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada - objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut (tidak selalu memiliki inheritas.)

2.3.2.6 *Polimorfosisme*

Polimorfosisme (Banyak bentuk) berarti satu objek dengan banyak bentuk yang berbeda, adalah konsep sederhana dalam bahasa pemogramana berorientasi objek yang berarti kemampuan dari suatu variabel referensi objek untuk memiliki aksi berbeda bila method yang sama dipanggil, dimana aksi method tergantung dari tipe objeknya.

2.3.2.7 *Interface*

Interface merupakan *device* yang digunakan untuk komunikasi antar objek berbeda yang tidak memiliki hubungan apapun. *Interface* bisa dikatakan sebagai protokol komunikasi antar objek tersebut.

2.4 Perangkat Lunak Pendukung

Perangkat lunak (*software*) adalah peralatan untuk menunjang untuk kerja dari perangkat keras (*hardware*). Perangkat lunak memberikan instruksi-instruksi yang dapat ditanggapi dan dimengerti oleh perangkat keras komputer. Perangkat lunak komputer (*software*) dapat dikelompokkan ke dalam tiga kelompok yaitu :

a. Sistem Operasi (*Operating Sistem*)

   Merupakan program yang berfungsi untuk mengendalikan dan mengkoordinasikan kegiatan sistem komputer.

   Contoh : Windows 9x, NT, Me, Xp Linux, dan lain-lain.

b. Perangkat lunak aplikasi (*Aplication Software*)

   Merupakan program yang ditulis dan dierjemahkan oleh bahasa pemrograman untuk keperluan aplikasi tertentu.

c. Bahasa Pemrograman (*Programming Language*)

   Merupakan program yang digunakan untuk menterjemahkan suatu bahasa pemrograman ke dalam bahasa mesin, agar dapat dimengerti oleh komputer.

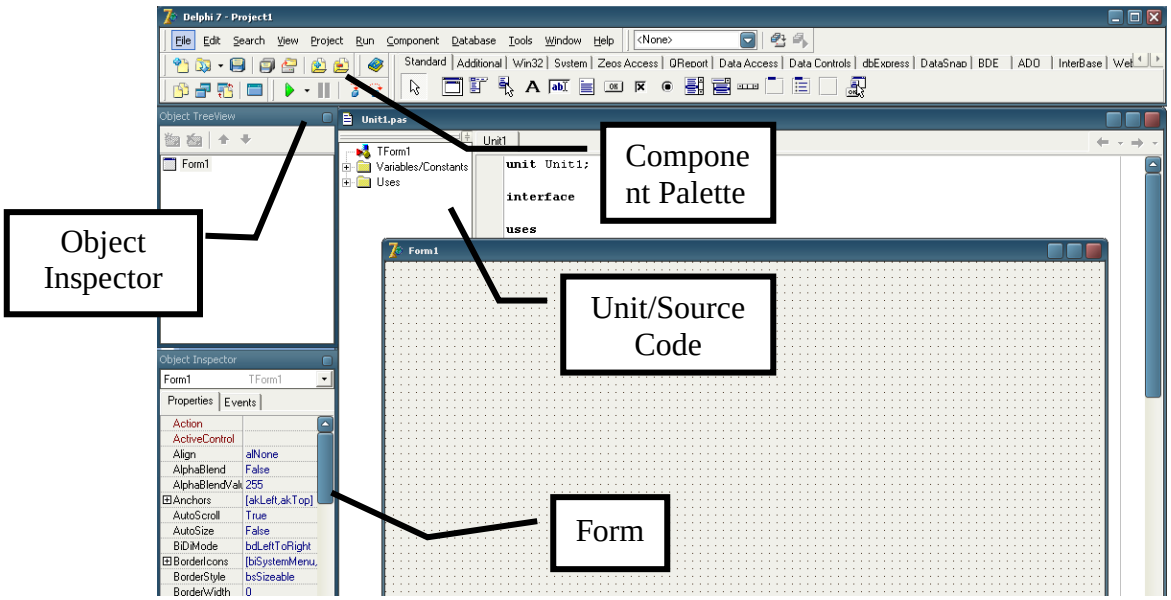
   Contoh : Microsoft Word, Excel, Corel drae, dan lain-lain.

   Adapun program aplikasi yang digunakan adalah Borland Delphi 7.0, dan MySQL 5.0 sebagai basis data.

2.4.1 **Pengenalan Borland Delphi 7.0**

Delphi adalah suatu program berbasis bahasa Pascal yang berjalan dalam lingkungan Windows. Delphi telah memanfaatkan suatu teknik pemrograman yang disebut RAD yang telah membuat pemrograman menjadi lebih mudah. Delphi adalah suatu bahasa pemrograman yang telah memanfaatkan metode pemrograman *Object Oriented Programming* (OOP).

Lingkungan kerja Borland Delphi dapat dilihat pada gambar di bawah ini.



Gambar 2.18 Elemen-elemen Borland Delphi 7.0

Fungsi dari elemen-elemen di atas adalah :

- *Object Inspector* : suatu window yang berguna untuk mengatur suatu object baik properti, events dan method.

- *Form* : Digunakan sebagai layar/window yang digunakan sebagai lembar kerja kita. Di form-lah semua komponen seperti tombol dan komponen lainnya disimpan.
- *Window Unit/Source Code* : Window/layar yang berisi perintah-perintah yang akan dieksekusi oleh komputer. Di layar inilah kita mengisikan program-program.
- *Component Palette* : Layar yang berisikan komponen-komponen yang dipakai dipakai dalam program kita.

2.4.2 ***Database***

Perancangan basis data / *database* dilakukan untuk memperoleh gambaran *database* yang akan dibuat untuk mempermudah proses pengelolaan manajemen *database*.

2.4.2.1 **Pengertian *Database***

Database merupakan komponen terpenting dalam pembangunan sistem informasi karena menjadi tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sebuah sistem sehingga dapat dieksplorasi untuk membentuk informasi-informasi dalam berbagai bentuk.

Menurut Jogiyanto (2005:46) *database* adalah kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.

Dari definisi ini, terdapat tiga hal yang berhubungan dengan basis data, yaitu :

1. Data itu sendiri yang diorganisasikan dalam bentuk basis data (*database*).
2. Simpanan permanen (*storage*) untuk meyimpan basis data tersebut.
3. Perangkat lunak untuk memanipulasi basis datanya. Perangkat lunak ini dapat dibuat sendiri dengan menggunakan bahasa pemrograman komputer atau dibeli dalam bentuk suatu paket. Banyak paket perangkat lunak yang disediakan untuk memanipulasi basis data. Paket perangkat lunak ini disebut dengan DBMS (*Data Base Management System*). Contoh DBMS yang terkenal misalnya adalah dBASE, Fox Base, Microsoft Access, Oracle, dan lain-lain.

   DBMS yang populer untuk mengolah basis data sekarang ini adalah RDBMS (*Relational Data Base Management System*). RDBMS menggambarkan suatu file basis data seperti suatu tabel, yaitu bagian kolom menggambarkan *field* dari data dan bagian baris menunjukkan *record* dari data yang ada didalam tiga file basis data.

   Adapun penerapan database ini antara lain untuk pembangunan sistem informasi, penyediaan barang, akuntansi, pemasaran produksi, layanan pelanggan yang digunakan dalam perusahaan retail, perbankan, perhotelan, sekolah-sekolah, dan sebagainya.

   Tingkatan data dalam *database*, dapat disusun berdasarkan sistem tigkatan unik, yaitu :

   a. Database

Merupakan kumpulan *file* yang saling terkait satu sama lain.

b. File

Merupakan kumpulan dari *record* yang saling terkait dan memiliki format *field* yang sama dan sejenis.

c. Record

Merupakan kumpulan *field* yang menggambarkan suatu unit data individu tertentu.

d. Field

Merupakan atribut dari *record* yang menunjukkan suatu item dari data.

e. Byte

Yaitu atribut dari *field* yang berupa huruf yang membentuk nilai dari sebuah *field*. Huruf tersebut dapat berupa numerik maupun abjad ataupun karakter khusus.

f. Bit

Yaitu bagian kecil dari data secara keseluruhan, yaitu berupa karakter ASCII nol atau satu yang merupakan komponen pembentuk *byte*.

Database yang penulis pakai dalam aplikasi pengolahan apotek dan pengobatan ini adalah dengan menggunakan MySQL 5.0.

2.4.2.2 **My SQL 5.0**

MySQL adalah sebuah program *database server* yang mampu menerima dan mengirimkan datanya dengan sangat cepat, multi user serta menggunakan perintah standar SQL (*Structured Query Language*).

MySQL memiliki dua bentuk lisensi, yaitu *Free Software* dan *Shareware*. MySQL yang biasa kita gunakan adalah MySQL *Free Software* yang berada dibawah lisensi GNU/GPL (*General Public License*).

MySQL merupakan sebuah *database server* yang *free*, artinya kita bebas menggunakan *database* ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayarlisensinya. MySQL pertama kali dirintis oleh seorang programmer *database* bernama *Michael Widenius.* Selain sebagai *database server*, MySQL juga merupakan program yang dapat mengakses suatu *database* MySQL yang berposisi sebagai *server*. Pada saat itu bearti progam kita berposisi sebagai *Client.* Jadi MySQL adalah sebuah database yang dapat digunakan baik sebagai *Client* maupun *Server.*

Database MySQL merupakan suatu perangkat lunak *database* yang berbentuk database relasional atau dalam bahasa basisdata sering disebut dengan *Relation Database Management System (RDBMS)* yang menggunakan suatu bahasa permintaan bernama SQL.

2.5 *System Development Life Cycle*

System Development Life Cycle (SDLC) atau yang dikenal dengan Sistem Daur Hidup merupakan suatu bentuk yang digunakan untuk menggambarkan tahapan utama dan langkah-langkah di dalam tahapan tersebut dalam proses pengembangan sistemnya.

Metode daur hidup terdiri dari dua tahap yaitu untuk *front end* (bagian konsep) terdapat tahapan proses perencanaan, analisis, rancangan sistem general, evaluasi dan seleksi sedangkan di tahap *back end* (bagian

fuingsional) terdapat tahapan proses rancangan sistem terinci, implementasi dan pemeliharaan. Di setiap tahapan proses daur hidup dilakukan proses pendokumentasian untuk laporan atas segala yang telah dilakukan atau disepakati dalam setiap tahap tersebut. Tahapan-tahapan seperti ini sebenarnya merupakan tahapan di dalam pengembangan system teknik (*engineering systems*).

1. Tahap Perencanaan

   Pada tahap ini, tim pembuat sistem mencoba memahami permasalahan yang muncul dan mendefinisikannya secara rinci setelah manajemen puncak menetapkan kebijakan untuk mengembangkan suatu pengembangan sistem, kemudian menentukan tujuan pembuatan system dan mengidentifikasikan kendala-kendalanya. Hasil dituangkan dalam proposal proyek. Perencanaan sistem ini menyakut estimasi dari kebutuhan-kebutuhan fisik seperti bahan-bahan bangunan yang diperlukan, tenaga kerja yang akan menjalankan proyek pengembangan konstruksi gedung atau jembatan, dan dana yang dibutuhkan untuk mendukung pengembangan sistem ini serta untuk mendukung operasinya setelah diterapkan pengembangan konstruksi gedung.

2. Tahap Analisis

   Tahap ini, tim pengembangan sistem konstruksi akan menganalisis permasalahan secara lebih mendalam dengan menyusun suatu studi kelayakan. Pada tahap ini langkah pertama yang dilakukan adalah mengidentifikasi masalah. Tugas-tugas yang dilakukan sebagai berikut mengidentifikasi penyebab kenapa suatu konstruksi bangunan

harus dikembangkan dan mengidentifikasi keputusan yang baik. Langkah kedua memahami kerja dari konstruksi bangunan yang telah ada, langkah ketiga menganalisis system dan langkah yang terakhir membuat laporan analisis.

3. Tahap rancangan Sistem General, Evaluasi dan rancangan Sistem terinci

   Dengan memahami sistem sebelumnya dan kriteria-kriteria sistem yang akan dibangun, tim pengembang konstruksi dapat membuat rancangan sistem secara general kemudian rancangan tersebut di evaluasi sesuai dengan apa yang diinginkan pada sistem yang akan dibangun dengan membuat suatu rancangan sistem secara lebih terinci.

   Proses ini sangat diperlukan untuk menghasilkan suatu rancangan sistem yang baik, karena dengan adanya rancangan yang tepat akan menghasilkan sistem yang stabil dan mudah dikembangkan. Dalam perancangan suatu pemodelan, video ini menggunakan UML yang bersifat *object oriented*.

4. Tahap Implementasi dan Pemeliharaan

   Tahap ini merupakan kegiatan untuk mengimplementasikan rancangan yang telah disusun agar dapat diwujudkan. Proses implementasi untuk prosedur dalam teknologi komputer akan menggunakan bahasa pemprograman seperti yang dilihat, bahasa pemprograman yang digunakan pada video pengembangan konstruksi ini adalah Delphi. Pertimbangan untuk memilih bahasa pemprograman didasarkan pada dua hal, yaitu kemampuan bahasa itu untuk menangani dan mengimplementasikan proses-proses yang dirancang. Setelah

pengembangan selesai kemudian dilakukan pemeliharaan sistem yang telah terancang.

Gambar 2.19 Model *System Development Life Cycle*