BAB II

LANDASAN TEORI

2.1 Konsep Dasar Sistem

Pada dasarnya system ini merupakan sesuatu yang memiliki bagian atau komponen yang saling berinteraksi untuk mencapai tujuan tertentu dengan melalui tiga tahapan, yaitu : input, prosess dan output.

Menurut Jogiyanto (2003 :34) sistem dapat didefinisikan dengan pendekatan prosedur dan dengan pendekatan komponen. Ia juga mengungkapkan bahwa : "Dengan pendekatak prosedur, sistem dapat didefinisikan sebagai kumpulan dari prosedur-prosedur yang mempunyai tujuan tertentu, contoh dari sistem ini adalah sistem akuntasi. Pada sistem dapat didefinisikan sebagai kumpulan dari prosedur-prosedur penerimaan kas, pengeluaran kas, penjualan, pembelian dan buku besar.

Dari segi Etimologi, kata sistem sebenarnya berasal dari Bahasa Yunani yaitu "Systema", yang dalam Bahasa Inggris dikenal dengan "system", yang mempunyai satu pengertian yaitu sehimpunan bagian atau komponen yang saling berhubungan secara teratur dan merupakan satu keseluruhan yang tidak terpisahkan. Dalam arti luas dapat didefinisikan sebagai kumpulan elemen-elemen yang saling berhubungan dan saling bergantung untuk mencapai satu tujuan. Pendekatan definisi sistem berdasarkan pendekatan prosedur .

Sedangkan dengan pendekatan komponen, sistem yang dapat didefinisikan sebagai kumpulan dari komponen yang saling berhubungan satu dengan yang lainnya membentuk satu kesatuan untuk mencapai tujuan tertentu, contohnya adalah sistem computer. Pada sistem ini dapat didefinisikan sebagai kumpulan dari perangkat lunak dan perangkat keras".

"Sistem informasi dapat diartikan sebagai suatu susunan dari orang, data, proses dan teknologi informasi yang saling berkaitan untuk mengumpulkan, memproses, menyimpan, dan menyediakan keluaran informasi yang dibutuhkan untuk mendukung suatu organisasi".

Jadi dari pendefinisian diatas dapat diambil suatu kesatuan atau kumpulan yang terdiri dari dua atau lebih komponen yang saling berinteraksi untuk mencapai tujuan.

Suatu hal akan layak dikatakan sebuah sistem jika telah memenuhi delapan karakteristik sistem, yaitu sebagai berikut :

a. Suatu sistem mempunyai komponen-komponen sisten (components) atau subsistem-subsistem.
b. Suatu sistem mempunyai batas sistem (boundary)
c. Suatu sistem mempunyai lingkungan luar sistem (environment)
d. Suatu sistem mempunyai penghubung (interface)
e. Suatu sistem mempunyai masukan (input)
f. Suatu sistem mempunyai pengolahan (process)
g. Suatu sistem mempunyuai keluaran (output)
h. Suatu sistem mempunyai tujuan (goal) tau sasaran (objective)

2.1.1 Karakteristik Sistem

Sistem mempunyai beberapa karakteristik atau sifat-sifat tertentu, antara lain :

a. Komponen Sistem (*Component*)

   Suatu sistem terdiri daris sejumlah komponen yang saling berinteraksi, yang saling bekerja sama membentuk suatu komponen sistem atau bagian-bagian dari sistem.

b. Batasan Sistem (*Boundary*)

   Daerah yang membatasi antara suatu sistem dengan sistem yang lain atau dengan lingkungan kerjanya.

c. Subsistem

Bagian-bagian dari sistem yang beraktivitas dan berinteraksi satu sama lain untuk mencapai tujuan dengan sasarannya masing-masing.

d. Lingkungan Luar Sistem (*Enviroment*)

Suatu sistem yang ada di luar dari batas sistem yang dipengaruhi oleh operasi sistem.

e. Penghubung Sistem (*Interface*)

Media penghubung antara subsistem dengan subsistem yang lain. Adanya penghubung ini memungkinkan berbagai sumber daya mengalir dari suatu subsistem ke subsistem lainnya.

f. Masukan Sistem (*Input*)

Energi yang masuk ke dalam sistem, berupa perawatan dan sinyal. Masukan perawatan adalah energi yang dimasukkan supaya sistem tersebut dapat berinteraksi.

g. Keluaran Sistem (*Output*)

Hasil energi yang diolah dan diklarifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

h. Pengolah Sistem (*Prosess*)

Suatu sistem dapat mempunyai suatu bagian pengolah yang akan mengubah

i. Sasaran Sistem (*Object*)

Tujuan yang ingin dicapai oleh sistem, akan dikatakan berhasil apabila mengenai sasaran atau tujuan.

2.1.2 Klasifikasi Sistem

Suatu sistem dapat diklasifikasikan menjadi seperti berikut :

a. Sistem abstrak dan sistem fisik

   Sistem abstrak adalah suatu sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik, sedangkan sistem fisik adalah sistem yang ada secara fisik.

b. Sistem alamiah dan sistem buatan manusia

   Sistem alamiah adalah sistem yang terjadi melalui proses alam sedangkan sistem buatan manusia adalah sistem yang dirancang oleh manusia.

c. Sistem tertentu dan sistem tak tentu

   Sistem tertentu adalah suatu sistem yang operasinya dapat diprediksi secara tepat sedangkan sistem tak tentu adalah sistem dengan perilaku ke depan yang tidak dapat diprediksi.

d. Sistem tertutup dan sistem terbuka

   Sistem tertutup adalah sistem yang tidak terpengaruh oleh lingkungan luar ata otomatis, sedangkan sistem terbuka adalah sistem yang berhubungan dan terpengaruhi oleh lingkungan luar.

2.1.3 Konsep Dasar Informasi

Dari segi etimologi, kata sistem sebenarnya berasal dari Bahasa Yunani yaitu "*Systema*", yang dalam Bahasa Inggris dikenal dengan "*system*", yang mempunyai satu pengertian yaitu sehimpunan bagian atau komponen yang saling berhubungan secara teratur dan merupakan satu keseluruhan yang tidak terpisahkan.

Dalam arti luas dapat didefinisikan sebagai sekumpulan elemen-elemen yang saling berhubungan dan saling bergantungan utnuk mencapai suatu tujuan.

Jogiyanto HM, MBA Akt.,Ph.D. (2003:36) menyatakan bahwa informasi adalah data yang diolah menjadi bentuk yang berguna bagi para pemakainya. Data yang di olah saja tidak cukup dapat dikatakan sebagai suatu informasi. Untuk menjadi suatu informasi, maka data yang di olah tersebut harus berguna bagi pemakainya.

Menurut Jogiyanto kualitas informasi yang diharapkan tergantung dalam 4 (empat) hal pokok yaitu :

a. Akurat

Akurat mempunyai arti informasi yang dihasilkan harus bebas dari kesalahan-kesalahan, yang tidak biasa, tidak menyesatkan dan mencerminkan maksudnya. Informasi harus akurat karena dari seumber informasi sampai ke penerima informasi banyak terjadi gangguan yang dapat merubah atau merusak informasi tersebut.

b. Tepat Waktu

Tepat Waktu berarti informasi yang disampaikan ke penerima tidak terlambat, karena informasi adalah landasan untuk mengambil suatu keputusan. Informasi yang sudah using tidak akan mempunayai nilai lagi. Untuk itu diperlukan suatu teknologi untuk mengirim dengan cepat dan tepat.

c. Relevan

Berarti informasi yang mempunyai manfaat dan berguna bagi pemakainya, karena batas relevansi seseorang berbeda, makan infromasi dikatakan berguna jika benar-benar berguna dan dibutuhkan pemakainya.

d. Lengkap

Tidak boleh ada bagian informasi yang esensial bagi pengambilan keputusan atau pelaksanaan tugas yang hilang

e. Aman

Aman berarti informasi harus terbebas dari penyadapan oleh pihak orang yang tidak berwenang dalam penggunaan informasi tersebut.

2.1.4 Kualitas Informasi

Informasi yang berkualitas memiliki 3 kriteria, yaitu :

a. **Akurat (*Accurate*)**

   Informasi harus bebas dari kesalahan, tidak bias ataupun menyesatkan. Akurat juga berarti bahwa informasi itu harus dapat dengan jelas mencerminkan maksudnya.

b. **Tepat pada waktunya (*timeliness*)**

   Informasi yang datang pada penerima tidak boleh terlambat. Di dalam pengambilan keputusan, informasi yang sudah usang tidak lagi bernilai. Bila informasi datang terlambat sehingga pengambilan keputusan terlambat dilakukan, hal itu dapat berakibat fatal bagi perusahaan.

c. **Relevan (*relevance*)**

   Informasi yang disampaikan harus mempunyai keterkairan dengan masalah yang akan dibahas dengan informasi tersebut. Informasi harus bermanfaat bagi pemakainya. Di samping karakteristik, nilai informasi juga ikut menentukan kualitasnya. Nilai informasi (*value of information*) ditentukan oleh dua hal, yaitu manfaat dan biaya untuk mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih besar dibandingkan biaya utnuk mendapatkannya.

2.1.5 Konsep Dasar Sistem Informasi

Sistem informasi bukan merupakam hal yang baru. Yang baru. Yang barru adalah komputerisasinya. Sebelum ada computer, teknik penyaluran informasi yang memungkinkan manajer merencanakan serta mengendalikan operasi telah ada.

Abdul Kadir dalam bukunya yang berjudul Pengenalan Sistem Informasi (2003,5) mendefinisikan sistem informasi sebagai :"seperangkat komponen yang saling berhubungan dan berfungsi mengumpulkan, memproses, dan mendistribusikan informasi dan mendukung pembuatan keputusan dan pengawasan dalam organisasi".

2.1.6 Komponen Sistem Informasi

Dalam suatu sistem informasi terdapat komponen – komponen sebagai berikut :

a. Perangkat keras (*hardware*), mencangkup berbagai peranti fisik seperti komputer dan printer.
b. Perangkat lunak (*software*).

2.1.7 Kehandalan Sistem Informasi

Faktor – faktor yang menentukan keandalan dari suatu sistem informasi sbb:

a. Keandalan

   Keluaran sistem harus mempunyai tingkat ketelitian yang tinggi dari sistem itu sendiri harus mampu beroperasi secara efektif dan efisien yang dapat diandalkan.

b. Kegunaan

   Suatu sistem yang harus dapat menghasilkan informasi yang tepat waktu dan relevan untuk pengambilan keputusan.

c. Ekonomis

   Semua kegiatan komponen sistem harus dapat memberikan nilai manfaat yang besar dengan nilai biaya yang minimal.

d. Pelayanan

   Sistem harus dapat memberikan pelayanan – pelayanan dengan baik dan efisien kepada pengguna sistem sesuai dengan kondisi yang sedang terjadi dan diinginkan.

e. Kapasitas

   Sistem harus mempunyai kapasitas yang memadai untuk menangani periode – periode operasi puncak seperti halnya periode aktivitas normal.

f. Kesederhanaan

   Sistem harus cukup sederhana sehingga struktur operasinya dengan mudah di mengerti dan prosedurnya mudah di ikuti.

g. Fleksibilitas

Sistem harus cukup fleksibilitas untuk dapat menampung perubahan - perubahan kepentingan dalam kondisi dimana sistem beroperasi atau dalam kebutuhan yang diwajibkan.

2.2 Analisa dan Perancangan Sistem Berorientasi Obyek dengan UML

Analisa sistem dapat dinyatakan sebagai pemisahan suatu hal dengan bagian – bagian tertentu. Bagian – bagian tersebut kemudian dipelajari dan dievakuasi untuk mengetahui apakah terdapat cara – cara yang lebih baik untuk memenuhi kebutuhan manajemen.

Menurut Ariesto Hadi Sutopo (2002 : 242) : Pengertian Analisa sistem adalah sebagai berikut:

"Analisa sistem adalah proses menentukan kebutuhan sistem apa yang harus dilakukan sistem untuk memenuhi kebutuhan klien, bukanlah bagaimana sistem tersebut diimplementasikan." Konsep Dasar Berorientasi Objek (Object – Oriented). Konsep dasar berorientasi objek mencapai kematangannya pada saat masalah analisis dan desain menjadi lebih diperhatikan dari pada masalah coding. Secara fisik, pengertian "berorientasi objek" (Ariesto Hadi Sutopo, 2002:3) berarti bahwa "kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya.

2.2.1 Sejarah UML

Sejarah UML sendiri cukup panjang. Tahun 1950-an saat keterbatasan hardware, media penyimpanan dan software pemprogaman, muncul metode perancangan sistem yang berbasis proses. Muncul diagram -- diagram terkenal seperti Data Flow Diagram (DFD). Inti dari diagram ini adalah entitas apa dan melakukan proses apa dengan metode yang sangat terkenal SDLC : System Development Life Cycle. Tahun 1976, Chen menemukan Enity Relationship Diagram (ERD) yang berguna dalam memodelkan database dari suatu proses. Dimulailah era metode perancangan sistem berbasis DATA. Muncul istilah terkenal : Rational Database

Management System (RDBMS). Metode perancangan ini berusaha menutupi kelemahan metode perancangan berbasis proses. Perlu diketahui bahwa proses sangat cepat berubah dibandingkan data.

Sampai era tahun 1990, Tahun 90-an, diiringi membanjirnya software berorientasi object, bahwa hingga ke database seperti Oracle.SQL Server, dan lain – lain sudah menganut OR-DBMS (Object Relational –DBMS). Seperti kita ketahui puluhan metode pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah : metode booch [1], metodologi coad [2], metodologi OOSE [3], metodologi OMT [4], metodologi shlaer – mellor [5], metodologi wirfs-brock [6], dsb. Masa itu terkenal dengan masa perang metodologi (method war) dalam pendesainan berorientasi objek. Masing – masing metodologi membawa notasi sendiri –sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group / perusahaan lain yang menggunakan metodoloogi yang berlainan.

Dimulai pada bulan Oktobet 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlase draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group. Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek. Hingga saat ini UML sudah versi 2.2

2.2.2 Konsep Dasar UML

Untuk menguasai UML, sebenarnya cukup dua hal yang harus kita perhatikan, yaitu menguasai pembuatan diagram UML dan menguasai langkah – langkah dalam analisa dan pegembangan dengan UML.

Komponen penyusun utama dari UML adalah *things* dan *relationship* ; yang dikombinasikan dengan cara berbeda – beda dengan mengikuti aturan yang berbeda pula untuk menghasilkan tipe diagram yang berbeda. UML mendefinisikan diagram – diagram sebagai berikut :

a. *Use case diagram*
b. *Class diagram*
c. *Statechart diagram*
d. *Activity diagram*
e. *Sequence diagram*
f. *Collaboration diagram*
g. *Component diagram*
h. *Deployment diagram*

2.2.3 *Unifield Modeling Language* (UML)

UML adalah bahasa grafis untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML beriorientasi objek, menerapkan banyak level abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi, pemaduan beberapa notasi di beragam metodologi, usaha bersama dari banyak pihak, didukung oleh kakas – kakas yang diintegrasikan lewat XML. Stansar XML dikelola oleh OMG (*Object Management Group*). UML adalah bahasa pemodelan untuk menspesifikasikan, memvisualisasikan, UML bukanlah :

a. Bahasa pemprogaman visual, tapi bahasa pemodelan visual.
b. Spesifikasi kakas, tetapi spesifikasi bahasa pemodelan.
c. Proses, tetapi yang memungkinkan proses – proses.

2.2.4 Tujuan UML

Tujuan utama perancangan menggunakan UML adalah :

a. Menyediakan bahasa pemodelan visual yang ekspresif dan siap pakai untuk mengembangkan dan pertukaran model – model yang berarti.
b. Menyediakan mekanisme perluasan dan spesifikasi untuk memperluas konsep – konsep inti.
c. Mendukung spesifikasi independen bahasa pemprogaman dan proses pengembangan tertentu.
d. Menyediakan basis formal untuk pemahaman bahasa pemodelan
e. Mendorong pertumbuhan pasar kakas berorientasi objek
f. Mendukung konsep – konsep pengembangan level lebih tinggi seperti komponen, kolaborasi, *framework* dan *pattern*.

2.2.5 Diagram dan Teknik Pemodelan UML

Diagram mengemukakan banyak hal, penggunaan notasi yang terdefenisi baik dan ekspresif adalah penting pada proses pengembangan perangkat lunak, yaitu :

a. Notasi standar memungkinkan pengembang mendeskripsikan skenario atau rumusan arsitektur dan kemudian mengkomunikasikan secara tidak ambigu.
b. Notasi yang bagus membebaskan otak untuk berkonsentrasi pada masalah – masalah yang lebih lanjut.
c. Notasi yang baik memungkinkan mengeliminasi keperluan pemeriksaan konsistensi dan kebenaran keputusan dengan menggunakan *tool* terotomanisasi.

2.2.6 Sequence Diagram

Sequence Diagram menerangkan objek yang disusun dalam urutan tertentu. Urutan waktu yang dimaksud adalah ukuran yang dilakukan seseorang actor dalam menjalankan sistem. Diagram ini secara khusus berasosiasi dengan *use case sequence diagram* memeperlihatkan tahap demi tahap apa yang seharusnya terjadi. Untuk menghasilkan sesuatu dalam *use case diagram* ini sebaiknya digunakan di awal tahap perancangan atau analisis, karena kesederhanaannya dan mudah di mengerti.

Sequence diagram menunjukkan bagaimana data operasi dilakukan. Pesan apa yang dikirim dan kapan *Sequence diagram* didasarkan atas kelas diagram yang sudah dibuat hanya saja kelas diagram yang sudah dibuat belum menyertakan kelas *boundary* dan kelas *control*, maka sebelum membuat *Sequence diagram* perlu dibuat *class boundary* dan kelas *control* terlebih dahulu.

Beberapa symbol yang umun digunakan pada sequence diagram, yaitu :

a. *Entity object*, suatu objek yang berisi informasi kegiatan yang terkait yang tetap dan disimpan kedalam suatu database.
b. *Interface / Boundary object*, sebuah objek yang menjadi penghubung antara user dengan sistem. Contohnya window,dialog box atau screen (tampilan layar).
c. *Control Object*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab pada entitas. Contohnya adalah kalkulasi dan aturan bisnis yang melibatkan sebagai objek. Contoh object mengkoordinir pesan (message) antara boundary dengan entitas.
d. *Simple Message*, simbol pengiriman pesan dari sebuah objek ke objek lain.
e. *Recursive* sebuah objek yang mempunyai sebuah operation kepada dirinya sendiri.
f. *Activation*, activation mawakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebah operasi.
g. *Lifeline*, garis titik – titik yang terhubung dengan objek, sepanjang lifeline terdapat activation.

2.2.7 Diagram Struktur

Diagram ini untuk memvisualisasi, menspesifikasikan, membangun dan mendokumentasikan aspek statik dari sistem. Diagram struktur pada UML terdiri dari:

a. Diagram Kelas (*Class diagram*)

   Diagram ini menunjukkan sekumpulan kelas, *interface* dan kolaborasi serta keterhubungannya. Diagram kelas ditujukan untuk pandangan statik terhadap sistem.

b. Diagram Objek (*Objek diagram*)

   Diagram ini menunjukkan sekumpulan objek dan keterhubungannya. Diagram ini menunjukkan potongan statik dari insan - insan yang ada pada diagram kelas. Diagram ini memperlihatkan satu prototipe atau khusus tertentu yang mungkin terjadi. Diagram objek menyediakan notasi grafis formal guna memodelkan objek, kelas dan saling keterhubungan. Diagram objek berguna untuk *abstract modelling* dan perancangan program - program sesungguhnya.

c. Diagram Komponen (*Component diagram*)

   Diagram ini menunjukkan organisasi dan kebergantungan diantara sekumpulan komponen. Diagram ini merupakan statik terhadap implementasi sistem.

d. Diagram Pengembangan (*Deployment diagram)*

   Diagram ini menunjukkan konfigurasi pemprosesan saat jalan dan komponen - komponen yang terdapat didalamnya. Diagram ini merupakan pandangan statik dari arsitektur. Pilihan model dan diagram yang digunakan dipengaruhi oleh bagaimana persoalan ditangani dan bagaimana solusi dibentuk.

2.2.8 Diagram Perilaku

Diagram ini untuk memvisualisasi, menspesifikasi, membangun dan mendokumentasikan aspek dinamis dari sistem. Diagram perilaku pada UML terdiri dari :

a. Diagram Use case (*Use case diagram*)

   Diagram ini menunjukkan sekumpulan kasus fungsional dan aktor (jeniss kelas khusus) dan keterhubungannya.

b. Diagram sequence (*sequence diagram*)

   Diagram ini menunjukkan interaksi yang terjadi antar objek. Diagram ini merupakan pandangan dinamis terhadap sistem. Diagram ini menekankan pada basis keberurutan waktu dari pesan – pesan yang terjadi.

c. Diagram Kolaborasi (*Collaboration diagram*)

   Diagram ini merupakan diagram interaksi. Diagram ini menekankan pada organisasi struktur dari objek -- objek yang mengirim dan menerima pesan.

d. Diagram Statechart (*Statechart diagram*)

   Diagram ini berisi *state*, transisi, kejadian dan aktivitas. *Statechart* merupakan pandangan dinamis dari sistem. Diagram ini penting dalam memodelkan perilaku antarmuka, kelas, kolaborasi dan menekankan pada urutan kejadian. Penting untuk sistem reaktif yang dipicu kejadian di dunia nyata.

e. Diagram Aktivitas (*Activity diagram)*

   Diagram ini menunjukkan aliran di sistem . diagram ini adalah pandangan dinamis terhadap sistem. Diagram ini penting untuk memodelkan fungsi sistem dan menekankan pada aliran kendali diantara objek - objek.

2.2.9 Package Diagram

Package diagram adalah sebuah pengelompokan yang memungkinkan untuk mengambil setiap bentuk di UML dan mengelompokkan elemen – elemennya dalam tingkatan unit yang lebih tinggi. Kegunaannya yang paling umum untuk mengelompokkan class.

Dalam sebuah model UML setiap class merupakan anggota sebuah package tunggal. Package- package dapat juga merupakan anggota package lain. Jadi suatu struktur hirarkis dimana package yang paling atas memiliki beberapa subpackage dengan beberapa subpackage sendiri dan seterusnya sampai hirarki tersebut berakhir pada class. Sebuah package dapat terdiri dari sbupackage dan class. Setiap package mewakili sebuah namespace, artinya setiap class harus memiliki sebuah nama unik didalam packagenya.

2.2.10 Notasi dalam UML

UML menyediakan beberapa notasi dan artifak standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. Artifak di dalam UML didefinisikan sebagai informasi dalam berbagai bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat lunak. Yang perlu diperhatikan untuk menjaga konsisten antar artifak, selama proses analisis dan desain adalah bahwa setiap perubahan yang terjadi pada suatu artifak harus juga dilakukan pada artifak sebelumnya

a. Aktor

   Adalah segala sesuatu yang berinteraksi dengan sistem aplikasi komputer. Jadi aktor dapat berupa manusia, perangkat keras atau mungkin objek lain dalam sistem yang sama. Biasanya yang dilakukan aktor adalah memberikan informasi pada sistem dan atau memerintahkan sistem untuk melakukan sesuatu.

b. Kelas

Kelas merupakan pembentuk utama dari sistem berorientasi objek karena kelas menunjukkan kumpulan objek yang memiliki atribut dan operasi yang sama. Kelas digunakan untuk mengabstrasikan elemen – elemen dan sistem yang sedang dibangun. Kelas bisa untuk mempresentasikan baik perangkat lunak maupun perangkat keras, baik konsep maupun benda nyata.

c. *Interface*

Interface merupakan kumpulan operasi tanpa implementasi dari suatu kelas. Implementasi operasi dalam *Interface* dijabarkan oleh operasi dalam kelas. Oleh karena itu keberadaan *Interface* selalu disertai oleh kelas yang mengimplementasikan operasinya. *Interface* ini merupakan salah satu cara mewujudkan prinsip enkapsulasi dalam objek.

d. *Use Case*

Use – case menjelaskan urutan kegiatan yang dilakukan actor dan system untuk mencapai suatu tujuan tertentu walaupun menjelaskan kegiatan. Namun hanya menjelaskan "apa" yang dilakukan oleh actor dan system, bukan "bagaimana" actor dan system melakukan kegiatan tersebut.

e. Interaksi

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

f. Paket

Paket adalah container atau wadah konseptual yang digunakan untuk mengelompokkan elemen – elemen dari system yang sedang dibangun, sehingga bias dibuat model yang lebih sederhana. Tujuannya adalah untuk mempermudah penglihatan (*visibility*) dari model yang sedang dibangun.

g. Catatan (*Note*)

Note untuk memberikan keterangan dan komentar tambahan dari satu elemen sehingga bisa langsung terlampir dalam model. Catatan ini bisa ditampilkan ke semua elemen notasi yang lain.

h. Ketergantungan (*Depedency*)

Ketergantungan merupakan relasi yang menunjukkan bahwa perubahan pada salah satu elemen member pengaruh pada elemen lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada di bagian tanpa tanda panah.

2.3 Analisa Sistem Berorientasi Obyek

Konsep dasar berorientasi objek mencapai kematangan pada saat masalah analisis dan desain menjadi lebih diperhatikan dari pada masalah coding. Secara spesifik, pengertian "berorientasi objek" (Ariesto Hadi Sutopo, 2002 :3) berarti bahwa "kita mengoraganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya".

Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatak berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, polymorphism (polimorfisme) dan inheritance (pewarisan).

Metodologi pengembanagn sistem berorientasu objek mempunyai tiga karakteristik utama, yaitu :

a. Encapsulation

Encapsulation (pengkapsulan) merupakan dasar untuk pembahasan ruang lingkup program terhadap data yang diproses.

b. Inheritance

Inheritance (Pewarisan) adalah tehnik yang menyatakan bahwa anak dari objek akan mewarisi data / atribut dan metode dari induknya langsung. Sifat yang dimiliki oleh kelas induknya tidak perlu diulang dalam setiap subkelasnya.

c. Polymorphism

Polymorphism (polimorfisme) yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda.

2.3.1 *Activity Diagram*

Menurut Munawar (2005 :109) Menyatakan Bahwa : *Activity Diagram* adalah teknik untuk mendeskripsikan logika procedural proses bisnis dan aliran kerja dalam banyak kasus. Activity Diagram mempunyai peran seperti halnya *Flowchart*, akan tetapi perbedaannya dengan *Flowchart* adalah *activity diagram* biss mendukung perilaku pararel sedangakan *Flowchart* tidak bisa".

Sebuah *activity diagram* mempunyai :

a. *Start point* (*initial node*), menggambarkan awal dari aktifitas.
b. *End point* (*activity final node*), menggambarkan akhir dari aktifitas.
c. *Activities*, menggambarkan prosesa bisnis dan dikenal sebagai *activity state*.

Jenis – jenis *activities :*

1) *Black hole activities*

Ada masukan dan tidak ada keluaran, biasanya digunakan jika dikehendaki ada 1 atau lebih transaksi.

2) *Miracle activities*

Tidak ada masukan dan ada keluaran, biasanya dipakai pada waktu *start point* dan dikehendaki ada 1 atau lebih transaksi.

3) *Parallel activities*

Suatu *activity* yang berjalan secara bebarengan. Terdiri dari :

(a) *Fork* (percabangan)

Mempunyai 1 transaksi masuk dan 2 atau lebih transaksi keluar.

(b) *Join* (penggabungan)

Mempunyai 2 atauu lebih transisi masuk dan hanya 1 transisi keluar,

4) Decision Point

Digambarkan dengan lambing wajik atau belah ketupat.

Mempunyai transisi (sebuah garis dari / ke decision point). Setiap transisi yang ada harus mempunyai *GUARD* (kunci). Tidak ada sebuah keterangan (pertanyaan) pada tengah belah ketupat seperti pada *Flowchart.*

5) Guard (kunci)

Adalah sebah kondisi benar sewaktu melewati sebuah transaksi. Digambarkan dengan diletakkan diantara tanda [ ]. Tanda (*Otherwise*) *guard* untuk menangkap suatu kondisi yang belum terdeteksi. Setiap transisi dari/ke *decision* point harus mempunyai guard yang konsisten dan lengkap serta tidak *overlap.*

Contoh : X <0, X = 0 dan X > 0 konsisten

X < = 0 dan X > = 0 tidak konsisten

6) Swimlane

Sebuah cara untuk mengelompokkan *activity* berdasarkan actor (mengelompokkan activity dalam sebuah urutan yang sama). Actor bisa ditulis nama *actor* ataupun sekaligus dalam lambing *actor* (*stick figure*) pada *use case diagram. Swimlane* digambarkan secara *vertical,* walaupun kadang – kadang digambarkan secara horizontal.

7) Swimarea

Ketika sebuah *activity diagram* mempunyai banyak *swimlane,* perlu difikirkan dengan pendekatan *swimarea. Swimarea* mengelompokkan *activity* berdasarkan kegiatan di dalam *use case.*

2.3.2 Analisa Dokumen Keluaran

Analisa keluaran adalah analisa mengenai dokumen – dokumen keluaran yang dihasilkan dari sebuah sistem.

2.3.3 Analisa Dokumen Masukan

Analisa masukan adalah bagian dari pengumpulan informasi tentang sistem yang sedang berjalan. Tujuan analisa masukan adalah memahami prosedur berjalan.

2.3.4 *Use Case Diagram*

Use case Diagram menggambarkan kebutuhan system dari sudut pandang *user* dan menfokuskan pada proses komputerisasi. Menggambarkan hubungan antara *use case* dengan *actor*. Secara umum *use case* adalah pola perilaku system dan urutan transaksi yang berhubungan yang dilakukan oleh satu *actor*.

Use case diagram menggambarkan kebutuhan sistem dari sudut pandang *user* dan menfokuskan pada proses komputerisasi. Sebuah *use case* dapat menggambarkan hubungan antara *use case* dengan *actor*. Secara umum *use case* adalah pola perilaku sistem dan urutan transaksi yang berhubungan yang dilakukan. Oleh satu *actor*.

Secara umum *Use case diagram* terdiri dari :

a. *Actor*

menggambarkan orang, sistem atau eksternal entitas / stakeholder yang menyediakan atau menerima informasi dari sistem. *Actor* adalah entity eksternal yang berhubungan dengan sistem yang berpartisipasi dalam *use case*. *Actor* dilukiskan dengan peran yang mereka mainkan dalam *use case* seperti pelanggan, kasir, supplier, penjual dan lain-lain.

b. *Use case*

Use case dibuat berdasarkan keperluan *actor*, merupakan "apa" yang dikerjakan sistem, bukan "bagaimana" sistem mengerjakannya. *Use case* diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Nama *use case* boleh berdiri beberapa kata dan tidak boleh ada dua *use case* yang memiliki yang sama.

c. Associations

menggambarkan bagaimana actor terlihat dalam *use case* dan bukan menggambarkan aliran data atau informasi. *Associations* digambarkan dengan sebuah garis berpanah terbuka pada salah satu ujungnya yang menunjukkan arah relasi.

Empat jenis relasi yang bisa timbul pada *use case diagram*.

a. *Association* antar *Actor* dan *use case*

Ujung panah ada *Association* antar *Actor* dan *use case* mengindikasikan siapa / apa yang meminta interaksi dan bukannya mengindikasikan aliran data. *Association* antar *Actor* dan *use case* sebaiknya menggunakan garis tanpa panah. Antar *use case* yang menggunakan panah terbuka untuk mengindikasikan bila *actor* berinteraksi secara pasif dengan sistem.

b. *Association* antar *use case*

1) *<< include >>*

Digunakan ketika dalam penulisan *use case* yang berbeda-beda terdapat deskripsi-deskripsi yang sama, maka relasi ini dapat digunakan untuk menghindari penulisan deskripsi yang berulang-ulang. 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 setiap kali *Use case* 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.

2) *<< extend >>*

Sebuah *use case* juga dapat meng-extend *use case* lain dengan *behavior*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

3) *Generalization / inheritance* antara *use case*

Generalization / inheritance digunakan ketika ada sebuah keadaan yang lain sendiri / perilaku khusus. *Generalization / inheritance* digambarkan antara *use case* secara vertical dengan *inheritance use case* dibawah *base / parent use case.*

4) *Generalization / inheritance* antara *actor*

Generalization / inheritance antara *actor* secara vertical dengan *inheriting actor* dibawah *base / parent use case.*

2.4 Perancangan Sistem Berorientasi Objek

Selama analisis, perhatian kita adalah pada yang harus dikerjakan sistem, selama perencanaan keputusan dibuat tentang bagaimana pemecahan masalah akan dikerjakan. Perancangan sistem berorientasi objek merupakan proses spesifikasi yang terperinci atau pendefenisian dari kebutuhan – kebutuhan fungsional dan persiapan untuk rancang bangun implementasi yang menggambarkan bagaimana suatu sistem dibentuk, untuk mengembangkan suatu sistem baru dilakukan dengan menguraikan hubungan proses – proses dalam bentuk diagram – diagram.

"Object – Oriented Design merupakan tahap lanjutan setelah Analisis Berorientasi Objek dimana tujuan sistem diorganisasikan ke dalan subsistem berdasar struktur dan arsitektur yang dibutuhkan" (Ariesto Hadi Sutopo 2002 :244).

Tahap –tahap yang dilakukan dalam perancangan berorientasi objek sebagai berikut :

1. Perancangan Basis Data

   Merupakan tahap merancang basis data yang diterapkan oleh sistem.

2. Diagram Interaksi (Interaction Diagram)

   Interaction Diagram menggambarkan interaksi antar objek (instan dari kelas) didalam dan sekitar sistem yang menekankan pada pesan (message) apa yang disampaikan dan digambarkan dengan menekankan dimensi waktu atau oeran masing-masingnya.

3. Diagram Kelas (Class Diagram)

   Class adalah sebuah spesifikasi yang akan dihasilkan sebuah objek dan merupakan inti dari penegmbanagn dan desain berorientasi objek.

4. Diagram Komponen (Component Diagram)

   Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak termasuk ketergantungan (dependency) diantaranya.

5. Diagram Arsitektur Sistem (Deployment Sistem)

   Deployment diagram menggambarkan arsitektur sistem dan bagaimana komponene di-deploy dalam infrastruktur sistem.

6. Normalisasi

   Proses pengelompokkan atribut dari relasi sehingga membentuk Well Structure Relation, normalisasi merupakan sebuah teknik dalam logical design sebuah database.

2.4.1 *Entity Relationship Diagram* (ERD)

ERD adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam system secara abstrak. Jadi, jelaslah bahwa ERD ini berbeda dengan DFD yang merupakan suatu model jaringan fungsi yang akan dilaksanakan oleh system, sedangkan ERD merupakan model jaringan data yang menekankan pada struktur-struktur dan relationship data. Alas an menggunakan Model ERD adalah untuk mudah dimengerti oleh pemakai dan mudah disajikan oleh perancang database. Komponen – komponen penyusun *Entity Relationship Diagram* adalah berikut :

a. Entitas

Entitas adalah segala sesuatu yang dapat digambarkan oleh data. Entitas juga dapat diartikan sebagai individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain (Fathansyah, 1999)

b. Atribut

Atribut merupakan pendeskripsian karakteristik dari entitas. Atribut digambarkan dalam bentuk lingkaran atau elips. Atribut yang menjadi kunci entitas atau key diberi garis bawah.

Jenis-jenis kunci (*key*) diberi garis bawah :

1) *Primary Key*, yaitu suatu *field* atau atribut yang mengidentifikasi record dalam *Field* dan bersifat untuk satu kejadian dari *Entity*.
2) *Secondary Key*, yaitu *field* atau atribut yang dapat menghilangkan kemungkinan *primary key* yang tidak unik.
3) *Candidate Key*, yaitu *field* yang hanya dapat dicalonkan sebagai *Primary Key*.
4) *Alternate Key*, yaitu dari beberapa kandidat tidak dapat dipakai sebagai *Primary Key*.
5) *Composite Key*, beberapa *field* yang digabungkan menjadi satu karena tidak ada satupun *field* yang bisa dijadikan *Primary Key*.
6) *Foreign Key*, yaitu *field* yang bukan *key*, tetapi berupa *key* pada *field* lain atau satu *field* yang melengkapi satu relasi yang menunjukkan ke induknya.

c. Relasi atau Hubungan

Relasi menunjukkan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan *entitas* yang berbeda.

2.4.2 Transformasi ERD ke LRS (*Logical Record Structure*)

Sebuah model sistem yang digambarkan dengans sebuah Diagram-ERD akan mengikuti pola/aturan pemodelan tertentu. Dalam kaitannya dengan konversi ke LRS, maka perubahan yang terjadi adalah mengikuti aturan-aturan.

2.4.3 *Logical Record Structure* (LRS)

LRS adalah digambarkan oleh kotak persegi panjang dan dengan nama yang unik. File record pada LRS ditempatkan dalam kotak. LRS terdiri dari link-link tipe record lainnya, banyaknya link dari LRS yang diberi nama oleh field-field yang kelihatannya pada kedua link tipe record.

Ada tiga kemungkinan hubungan yang terjadi :

a. *One to one* (1 : 1)

Tingkat hubungn dinyatakan satu pda satu, jika suatu kejadian pada entitas yang pertama hanya membuat satu hubungan dengan satu kejadian pada entitas kedua. Demikian juga sebaliknya, satu kejadian pada entitas kedua hanya biasa mempunyai satu hubungan satu kejadian pada entitas pertama.

b. *One to Many* (1 : M)

Tingkat hubungan satu pada banyak adalah sama dengan banyak pada satu, tergantung dari mana tingkat hubungan tersebut dilihat dengan satu kejadian pada entitas dari mana tingkat hubungan tersebut dilihat untuk satu kejadian pada entitasyang pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas kedua. Sebaliknya satu kejadian pada entitas yang kedua hanya bisa mempunyai satu hubungan dengan satu kejadian pada entitas yang pertama.

c. *Many to many* (M : N)

Tingkat hubungan banyak kebanyak terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan.

2.4.4 *Class Diagram*

Diagram *kelas* (*class diagram*) sangat membantu dalam visualisasi kelas dari suatu sistem. Hal ini disebabkan Karena class adalah deskripsi kelompok objek-objek dengan atribut (*roperty*), perilak (*operation*) dan relasi yang sama. Disamping itu class diagram bisa memberikan pandanganglobal atas sebuah sistem. Hal tersebut tercermin dari class-class yang ada dan relasinya satu dengan lainnya.

2.4.5 Konsep Normalisasi

Normalisasi merupakan sebuah teknik dalam logical desain sebuah basis data / database, teknik pengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik (tanpa *redundansi*)

Kegunaan normalisasi :

a. Memanipulasi pengulangan informasi.
b. Memudahkan identifikasi entity / obyek.
c. Menghindari kehilangan datatanpa sepengetahuan.

Langkah-langkah Normalisasi :

a. Nornal Pertama (1^{st} *Normal Form*)

   Aturan :

   1) Mendefinisikan atribut kunci
   2) Tidak adanya group berulang
   3) Semua atribut bukan kunci tergantung pada atribut kunci

b. Nomalisasi Kedua (2^{nd} *Normal Form)*

   Aturan :

   1) Sudah memenuhi dalam bentuk normal kesatu
   2) Sudah tidak ada ketergantungan parsial, dimana seluruh field hanya tergantung pada sebagian field kunci.

c. Normalisasi Ketiga (3^{rd} *Normal Form*)

Aturan :

1) Sudah berada dalam bentuk normal kedua
2) Tidak ada ketergantungan transitif (dimana field bukan kunci tergantung pada field bukan kunci lainnya).

Normalisasi seharusnya berada dalam bentuk normal tertinggi dan bergerak dari bentuk normal satu dan seterusnya untuk setiap kali membatasi hanya satu jenis redudansi.

Keseluruhannya Cuma ada lima bentuk normal. Tiga bentuk normal pertama menekankan redudansi yang muncul dari *Function Depedencies* sedangkan bentuk keempat dan kelima menekankan redudansi yang muncul.

2.5 Konsep Dasar Basis Data

Basis data merupakan kumpulan dari data yang saling berhubungan antara data yang satu dengan data yang lainnya yang tersimpan di simpanan luar computer dan digunakan perangkat lunak tertentu untuk memanipulasinya. Data merupakan salah satu komponen terpenting dalam suatu sistem informasi, karena berfungsi sebagai basis penyedia informasi bagi para pemakainya. Sistem Basis Data merupakan sistem informasi yang mengintegrasikan kumpulan dari data yang saling berhubungan. Dibawah ini merupakan istilah-istilah yang digunakan dalam Basis Data :

a. *Entity*

Adalah suatu obyek yang nyata dan dapat direkam. Contoh pada bidang kesehatan entity adalah pasien, dokter, obat, kamar dan sebagainya.

b. *Atribute*

Setiap entity mempunyai atribut atau sebutan untuk mewakili entity, misalnya atribut dari pasien adalah nama pasien

c. *Data Value*

Adalah data aktual atau informasi yang disimpan pada tiap data elemen atau atribut.

d. *Record*

Kumpulan elemen yang saling berkaitan menginformasikan tentang suatu entity secara lengkap. Satu record mewakili satu data atau informasi tentang seseorang : nomor register, nama pasien, alamat, tanggal berobat.

e. *File*

Kumpulan record-record sejenis yang mempunyai pasangan elemen yang sama, namun berbeda data valuenya.

f. *Data Base*

Merupakan kumpulan file-file yang mempunyai antara satu file dengan file yang lain sehingga membentuk satu bangunan basis data.

2.5.1 Tujuan Basis Data

Basis data bertujuan untuk mengatur data sehingga diperlukan kemudahan, ketepatan dan kecepatan dalam pengambilan keputusan. Syarat sebuah basis data yang baik adalah tidak ada redudansi dan inkonsistensi data, mempermudah pengaksesan data dan Mendukung Multiple User.

2.5.2 Manfaat Basis Data

Banyak manfaat atau kelebihan yang dapat kita peroleh dengan menggunakan basis data. Manfaat atau kelebihan basis data diantaranya adalah sebagai berikut :

a. Kecepatan dan kemudahan (*Speed*) : Basis data memiliki kemampuan dalam mengelompokkan, mengurutkan bahkan perhitungan dengan matematika. Dengan perancangan yang benar, maka penyajian informasi akan dapat dilakukan dengan cepat dan mudah.

b. Kebersamaan pemakai (*Sharability*) : sebuah basis data digunakan oleh banyak user dan banyak aplikasi. Untuk data-data yang diperlukan oleh banyak / bagian orang, tidak perlu dilakukan pencatatan di masing-masing bagian, tetapi cukup dengan satu basis data untuk dipakai bersama.

c. Pemusatan Kontrol data : Karena cukup dengan satu basis data untuk banyak keperluan, pengontrolan terhadap data juga ckup dilakukan di satu tempat saja.

d. Efisiensi ruang penyimpanan (*Space*) : Dengan pemakaian bersama, kita tidak perlu menyediakan tempat penyimpanan di beberapa tempat, tetapi sukup satu saja sehingga ini akan menghemat ruang penyimpanan yang yang dimiliki oleh sebuah organisasi. Dengan teknik basis data yang benar, jika akan dapat menyederhanakan penyimpanan sehingga tidak semua data harus disimpan.

e. Keakuratan (*Accuracy*) : Penerapan secara ketat aturan tipe data, domain data, keunikan data, hubungan antar data dan lain-lain, dapat menekan ketidakakuratan dalam pemasukan / penyimpanan data.

f. Ketersediaan (*Availability*) : Dengan basis data kita dapat mem-backup data, memilih-milih data mana yang masih diperlukan dan data mana yang perlu kita simpan ke tempat lain. Hal ini mengingat pertumbuhan transaksi suatu organisasi dari waktu ke waktu membutuhkan media penyimpanan yang semakin besar.

g. Kelengkapan (*Completeness*)

h. Keamanan (*Security*) : Pengguna diberikan hak akses yang berbeda - beda sesuai dengan kepentingan dan posisinya. Basis data bisa diberikan *Password* untuk membatasi orang yang mengaksesnya.

i. Kemudahan dalam pembuatan program aplikasi baru.

j. Pemakaian secara langsung : Basis data memiliki fasilitas untuk melihat datanya secara langsung dengan tool yang disediakan oleh DBMS dengan menggunakan *query*.

k. Kebebasan Data (*Data Independence*) : jika sebuah program telah selesai dibuat, dan ternyata ada perubahan ini / struktur data. Maka dengan basis data, perubahan ini hanya perlu dilakukan pada level DBMS tanpa harus membongkar kembali program aplikasinya.

l. *User View* : Basis data menyediakan pandangan yang berbeda – beda untuk tiap – tiap pengguna

2.5.3 Operasi Dasar Basi Data

Beberapa operasi dasar basis data yang sering digunakan adalah :

a. Pembuatan basis data
b. Penghapusan basis data
c. Pembuatan file / table
d. Penghapusan file / table
e. Pengubahan table
f. Penambahan / pengisian
g. Pengambilan data
h. Penghapusan data

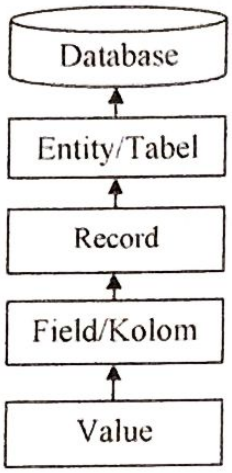
2.5.4 Hierarki Basis Data

Dalam Basi Data (*database*), struktur data disimpan dalam berbagai tingkatan / hierarki hingga membentuk suatu database. Tingkatan tertinggi berupa database dan tingkat terendah berupa *value* / nilai.

a. *Value* / nilai merupakan tingkatan terendah yang menyusun suatu database. *Value* atau nilai adalah data yang disimpan di dalam setiap kolom / elemen.
b. Field / kolom / Atribut adalah tingkatan kedua yang menyusun suatu *Record* yang menjelaskan kumpulan data yang disimpan.
c. *Record* / baris adalah kumpulan dari *field* / kolom yang akan saling berhubungan yang membentuk suatu tabel.

d. *Entity* / Tabel adalah kumpulan dari record data yang menjelaskan tentang subyek data.

e. *Database* / *File* adalah kumpulan tabel – tabel yang menjelaskan suatu subyek data.



Gambar 2.1
Tingkatan Data

2.5.5 Rancangan Dokumen Keluaran

Rancangan keluaran merupakan informasi yang akan dihasilkan dari keluaran sistem yang dirancang.

2.5.6 Rancangan Dokumen Masukan

Rancangan masukan merupakan data yang dibutuhkan untuk menjadi masukan sistem yang dirancang.

2.5.7 Rancangan Layar Program

Rancangan tampilan merupakan bentuk tampilan sistem layar computer sebagai antar muka dengan pemakaian yang akan dihasilkan dari sistem yang dirancang

2.5.8 Microsoft Access

Access (atau Microsoft Office Access) adalah sebuah program aplikasi basis data computer relasional yang ditujukan untuk kalangan rumahan desa dan perumahan kecil hingga menengah. aplikasi ini merupakan anggota dari beberapa aplikasi Microsoft Office, selain tentunya Microsoft Word, Microsoft Excel, dan Microsoft PowerPoint. Aplikasi ini menggunakan mesin basis data Microsoft jet Database Engine, dan juga menggunakan tampilan grafis yang intuitif sehingga memudahkan pengguna. Versi terakhir adalah Microsoft Office Access 2007 yang termasuk ke dalam Microsoft Office System 2007.

Microsoft Access dapat menggunakan data yang disimpan di dalam format Microsoft Access, Microsoft Jet Database Engine, Microsoft SQL Server, Oracle Database, atau semua kontainer basis data yang mendukung standar ODBC. Para pengguna/ programmer yang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang kompleks, sementara para programmer yang kurang mahir dapat menggunakannya untuk mengembangkan perangkat lunak yang sederhana. Access juga mendukung teknik-teknik pemprogaman berorientasi objek, tetapi tidak dapat digolongkan ke dalam perangkat lunak pemprogaman berorientasi objek.

2.6 Visual Basic.Net

Visual Basic.net adalah salah satu bahasa pemprogaman Komputer Tingkat Tinggi. Bahasa Pemprogaman adalah perintah – perintah yamg dimengerti oleh computer untuk melakukan tugas-tugas tertentu. Bahasa pemprogaman VB.NET dikembangkan oleh Microsoft, merupakan Salah Satu bahasa pemprogaman yang

Object Oriented Program (OOP) atau pemprogaman yang berorientasi Pada Object. Kata "Visual" meunjukkan cara yang digunakan untuk membuat Graphical User Interface (GUI). Dengan Cara ini, kita tidak perlu lagi menuliskan instruksi pemprogaman dalam kode-kode baris hanya untuk membuat sebuah Desaign Form / Aplikasi. Tetapi dengan sangat mudah yakni kita cukup melakukan Drag and drop object-object yang akan kita gunakan. VB.NET dapat kita jadikan alat Bantu untuk membuat berbagai macam program computer. Aplikasi VB.NET hanya dapat dijalankan pada system Operasi Windows.

2.6.1 *Crystal Report*

Crystal Report merupakan program khusus untuk membuat laporan yang terpisah dengan program visual basic 6.0 tetapi keduanya dapat dihubungkan (*Linkage*). *Crystal Report* merupakan salah satu reporting tools yang disediakan mulai di NET versi pertama keluar yaitu NET versi 1.0. Sebelum NET muncul *Crystal Report* merupakan reporting tools yang harus diinstal secara terpisah dan di referensi secara manual library nya apabila ingin digunakan. Hal tersebut sudah tidak berlaku lagi semenjak kemunculan NET pertama sehingga *Crystal Reports* sudah di include kan didalam Visual Studio.NET dan tidak perlu diinstal secara terpisah. *Crystal Reports* yang terdapat didalam Visual Studio 2008 merupakan *Crystal Imports* versi 2008 Basic Edition. Penggunaan *Crystal Imports* pada versi sebelum NET muncul sangat berbeda sekali . NET framework menyediakan library yang berbeda dengan library *Crystal Imports* yang biasa digunakan pada Visual Studio 6 dengan VB6 nya.

2.7 Teori Pendukung

a. Pengertian Rumah Sakit

"Rumah sakit adalah salah satu sarana kesehatan tempat menyelenggarakan upaya kesehatan dengan memberdayakan berbagai kesatuan personel terlatih dan terdidik dalam menghadapi dan menangani masalah medik untuk pemulihan dan pemeliharaan kesehatan yang baik".

Upaya kesehatan adalah setiap kegiatan untuk memelihara dan meningkatkan kesehatan yang bertujuan untuk mewujudkan derajat kesehatan yang optimal bagi masyarakat dan tempat yang digunakan untuk melakukan upaya kesehatan dasar, kesehatan dasar, kesehatan rujukan, dan atau upaya kesehatan penunjang. Upaya kesehatan diselenggarakan dengan pendekatan pemeliharaan, peningkatan kesehatan (promotif), pencegahan penyakit (preventif), penyembuhan penyakit (kuratif) dan pemulihan kesehatan (rehabilitative) yang diselenggarakan secara menyeluruh, terpadu dan berkesinambungan.

b. Pengertian Rawat Jalan

" Pelayanan Rawat Jalan adalah kegiatan fungsional yang dilakukan petugas medis, perawat dan / atau non medis yang melayani berbagai jenis pelayanan kesehatan yang dilaksanakan di Instalasi Rawat jalan (Poliklinik)"

c. Pengertian Pasien

Pasal 1 Undang-undang No. 29 Tahun 2004 Tentang Praktik Kedokteran menjelaskan **definisi pasien** adalah setiap orang yang melakukan konsultasi masalah kesehatannya untuk memperoleh pelayanan kesehatan yang diperlukan baik secara langsung maupun tidak langsung kepada dokter atau dokter gigi.

d. Pengertian Dokter

Secara operasional, definisi "Dokter" adalah seorang tenaga kesehatan (dokter) yang menjadi tempat kontak pertama pasien dengan dokternya untuk menyelesaikan semua masalah kesehatan yang dihadapi tanpa memandang jenis penyakit, organologi, golongan usia, dan jenis kelamin, sedini dan sedapat mungkin, secara menyeluruh, paripurna, bersinambung, dan dalam koordinasi serta kolaborasi dengan profesional kesehatan lainnya, dengan menggunakan prinsip pelayanan yang efektif dan efisien serta menjunjung tinggi tanggung jawab profesional, hukum, etika dan moral. Layanan yang diselenggarakannya adalah sebatas kompetensi dasar kedokteran yang diperolehnya selama pendidikan kedokteran.

e. Pengertian Perawat

Menurut Keputusan Menteri Kesehatan Nomor 1239/Menkes/SK/XI/2001 tentang Registrasi dan Praktik Perawat pada pasal 1 ayat 1.

Perawat adalah seseorang yang memiliki pengetahuan, keterampilan dan kewenangan untuk memberikan asuhan keperawatan pada orang lain berdasarkan ilmu dan kiat yang dimilikinya dalam batas-batas kewenangan yang dimilikinya.

Perawat adalah seseorang yang telah lulus pendidikan baik didalam maupun diluar negeri sesuai dengan peraturan perundang- undangan (Permenkes, 2010)

f. Pengertian Obat

Obat ialah suatu bahan atau paduan bahan-bahan yang dimaksudkan untuk digunakan dalam menetapkan diagnosis, mencegah, mengurangkan, menghilangkan, menyembuhkan penyakit atau gejala penyakit, luka atau kelainan badaniah dan rohaniah pada manusia atau hewan dan untuk memperelok atau memperindah badan atau bagian badan manusia termasuk obat tradisional.

Menurut Bagian Farmakologi, Fakultas Kedokteran, Universitas Indonesia, obat dalam arti luas ialah zat kimia yang dapat mempengaruhi proses hidup, maka farmakologi merupakan ilmu yang sangat luas cakupannya. Namun untuk seorang

doketr, ilmu ini dibatasi tujuannya yaitu agar dapat menggunakan obat untuk maksud pencegahan, diagnosis, dan pengobatan penyakit. Selain itu, agar mengerti bahwa penggunaan obat dapat mengakibatkan berbagai macam penyakit.

2.8 Teori Proyek

a. Pengertian Stakeholder

Stakeholder merupakan individu, sekelompok manusia, komunitas atau masyarakat baik secara keseluruhan maupun secara parsial yang memiliki hubungan serta kepentingan terhadap perusahaan. Individu, kelompok, maupun komunitas dan masyarakat dapat dikatakan sebagai *stakeholder* jika memiliki karakteristik yaitu mempunyai kekuasaan, legitimasi, dan kepentingan terhadap perusahaan.

b. Pengertian Work Breakdown Structure (WBS)

WBS adalah metode pengorganisasian proyek menjadi struktur pelaporan hierarkis. WBS digunakan untuk melakukan Breakdown atau memecahkan tiap proses pekerjaan menjadi lebih detail. Hal ini dimaksudkan agar proses perencanaan proyek memiliki tingkat yang lebih baik. WBS disusun berdasarkan dasar pembelajaran seluruh dokumen proyek yang meliputi kontrak, gambar-gambar, dan spesifikasi. Proyek kemudian diuraikan menjadi bagian-bagian dengan mengikuti pola struktur dan hirarki tertentu menjadi item-item pekerjaan yang cukup terperinci, yang disebut sebagai Work Breakdown Structure

c. Pengertian Milestone

Milestone adalah suatu bagian item pekerjaan yang dibuat seolah-olah menjadi temporary finish atau selesai sementara atas sekelompok atau serangkaian pekerjaan-pekerjaan yang menjadi bagian dari schedule besar. Item pekerjaan yang dijadikan milestone harulah item pekerjaan yang dianggap menjadi bagian penting sebelum melanjutkan pekerjaan berikutnya atau berpengaruh atas kelangsungan pekerjaan berikutnya.

d. Pengertian Rencana Anggaran Biaya (RAB)

RAB (Rencana Anggaran Biaya) adalah penghitungan banyaknya biaya yang diperlukan untuk bahan dan upah, serta biaya-biaya lain yang berhubungan dengan pelaksanaan bangunan atau proyek, baik secara kasar/taksiran maupun secara teliti. Dalam penghitungan RAB suatu proyek, sering kali membutuhkan sebuah aplikasi program computer agar penghitungan RAB cepat dan akurat.

e. Pengertian Manajemen Proyek

Manajemen dapat diartikan sebagai ilmu dan seni tentang upaya memanfaatkan semua sumber daya yang dimiliki untuk mencapai tujuan secara efektif dan efisien. Sebuah proyek adalah sebuah pekerjaan berbatas waktu yang menghasilkan produk tertentu yang unik, layanan, atau bentuk hasil lainnya. Sedangkan manajemen proyek adalah aplikasi dari sebuah pengetahuan, keahlian, alat bantu dan teknik tertentu untuk menyelaraskan kegiatan-kegiatan proyek agar berjalan sesuai dengan kebutuhan proyek utama.

Mengelola proyek terdiri dari : mengidentifikasikan kebutuhan, menentukan tujuan yang jelas, menyeimbangkan antara kualitas, cakupan, waktu, dan biaya proyek, serta mampu mengadaptasi berbagai spesifikasi, rencana, dan pendekatan-pendekatan berbeda dari berbagai pemangku kepentingan yang berbeda.

f. Pengertian *Responsibility Assignment Matrix* (RAM)

Responsibility assignment matrix (RAM) atau lebih dikenal dengan istilah RACI, adalah matriks yang menggambarkan peran berbagai pihak dalam penyelesaian suatu pekerjaan dalam suatu poyek atau proses bisnis. Matriks ini terutama bermanfaat dalam menjelaskan peran dan tanggung jawab antar bagian di dalam suatu proyek atau proses.

g. Pengertian Deliverables

Deliverables adalah sebuah output dari proyek yang dapat diukur seperti laporan, alpha tested produk, dan indicator-indicator kemajuan pengerjaan perangkat lunak.

h. Pengertian Project risk

Peristiwa tidak pasti yang bila memiliki pengaruh positif atau negative terhadap minimal satu tujuan proyek (waktu, biaya, ruang lingkup, mutu). Resiko mungkin memiliki satu atau lebih penyebab, yang bila terjadi memiliki satu atau lebih dampak.

i. Pengertian Project Execution Plan

Sebuah rencana eksekusi suatu proyek sangat erat kaitannya dengan estimasi biaya, dimana keduanya saling bergantung dan tidak akan terpenuhi keduanya secara total jika satu diantara keduanya tidak terselesaikan.