BAB II

LANDASAN TEORI

2.1 Definisi Sistem

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan sasaran yang ditentukan. (Jogiyanto H.M, 2005, Hal 1).

Pendekatan sistem merupakan jaringan kerja dari prosedur lebih menekankan urut-urutan operasi di dalam sistem. Suatu sitem mempunyai maksud tertentu. Ada yang menyebutkan maksud dari suatu sistem adalah untuk mencapai suatu tujuan (*goal*) dan ada yang menyebutkan untuk mencapai suatu sasaran (*objectives*). *Goal* biasanya dihubungkan dengan ruang lingkup yang lebih luas dan sasaran dalam ruang lingkup yang lebih sempit. Bila merupakan suatu sistem utama, seperti misalnya sistem bisnis, maka istilah *goal* lebih tepat diterapkan. Untuk sistem pembelian spare part atau sistem-sistem yang lainnya yang merupakan bagian atau subsistem dari sistem bisnis, maka istilah *objectives* yang lebih tepat. Jadi tergantung dari ruang lingkup dari mana memandang sistem tersebut. Seringkali tujuan (*goal*) dan sasaran (*objectives*) digunakan bergantian dan tidak dibedakan.

Sistem adalah Seperangkat unsur-unsur yang terdiri dari manusia, mesin atau alat dan prosedur serta konsep-konsep yang dihimpun menjadi satu untuk maksud tujuan yang bersama.

2.1.1 Karakteristik Sistem

Dalam teori sistem, unsur–unsur dapat mewakili sistem dengan cara umum yaitu masukan (*Input*), Pengolahan (*Proses*), dan Keluaran (*Output*). Hal ini merupakan konsep sebuah sistem yang sangat sederhana, sebab sebuah sistem dapat mempunyai beberapa masukan dan keluaran. Selain itu juga sistem mempunayai karakteristik atau sifat-sifat tertentu, yang merincikan hal tersebut bisa dikatakan sebagai suatu sistem. Suatu sistem memiliki karakteristik atau sifat tertentu, yaitu memiliki komponen-komponen,batasan sistem, lingkungan luar sistem, penghubung, masukan, keluaran,pengolahan, sasaran atau tujuan. Adapun karakteristik sistem yang dimaksud adalah sebagai berikut :

a. Komponen Sistem

   Suatu sistem terdiri sebuah komponen yang saling berinteraksi, yang artinya saling berkerja sama membentuk sebuah kesatuan. Komponen-komponen sistem dapat berupa subsistem atau bagian-bagian dari sistem.

b. Batasan Sistem

   Batasan sistem *(Boundary)* merupakan daerah yang membatasi antara satu sistem dengan sistem lainya, atau dengan lingkungan luarnya.Batasan sistem ini memungkinakan suatu sistem dipandang sebagai satu kesatuan.

c. Lingkungan Luar Sistem

   Lingkungan luar sistem *(environment)* dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem.

   Lingkungan luar sistem dapat bersifat menguntungkan juga merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi dari sistem dan dengan demikian harus tetap dijaga dan dipelihara. Sedangkan

lingkungan luar yang merugikan harus ditahan dan dikendalikan, jika tidak maka akan menggangu kelangsungan hidup dari sistem.

d. Penghubung Sistem

Penghubung *(interface)* merupakan media penghubung antara satu subsistem dengan yang lainnya. Melalui penghubung ini memungkinkan sunber-sunber daya mengalirdari subsistem lainnya. Keluaran dari satu subsistem akan menjadi masukan untuk subsistem dapat terintegrasi dengan subsistem lainnya membentuk satu kesatuan.

e. Masukan Sistem

Masukan *(input)* adalah energi yang dimasukan kedalam sistem. Masukan dapat berupa masukan perawatan *(maintenance input)* dan masukan sinyal *(signal input)*. Masukan perawatan adalah energi yang dimasukan agar sistem tersebut dapat beroperasi, dan masukan sinyal adalah energi yang diproses untuk didapatkan keluaran.

f. Keluaran Sistem

Keluaran *(output)* adalah hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. Keluaran dapat merupakan masukan untuk subsistem yang lain. Misalnya untuk sistem komputer, panas yang dihasilkan adalah keluaran yang tidak berguna dan merupakan sisa pembuangan, sedangkan informasi adalah keluaran yang dibutuhkan.

g. Sasaran Sistem

Suatu sistem pasti mempunyai tujuan atau sasaran. Jika suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan gunanya. Sasaran dari

sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya.

2.2 Definisi Informasi

Informasi merupakan data yang telah diproses sedemikian rupa sehingga meningkatkan pengetahuan seseorang yang menggunakan data tersebut.

(Abdul Kadir, 2003, Hal 31).

Informasi dikatakan berkualitas jika bergantung dari tiga hal sebagai berikut :

a. Akurat (*Accurate*)

Informasi harus bebas dari kesalahan-kesalahan dan tidak bisa atau menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya. Informasi harus akurat karena dari sumber informasi sampai ke penerima informasi kemungkinan banayk terjadi gangguan (*noise*) yang dapat merubah atau merusak informasi.

b. Tepat Waktu (*Timeliness*)

Informasi yang datang pada penerima tidak boleh terlambat, karena informasi merupakan landasan didalam pengambilan keputusan. Bila pengambilan keputusan terlambat, maka dapat berakibat fatal untuk organisasi.

c. Relevan (*Relevance*)

informasi tersebut mempunyai manfaat untuk pemakainya. Relevansi untuk tiap-tiap orang yang satu dengan yang lainnya berbeda.

2.3 Definisi Sistem Informasi

Sistem informasi merupakan sebuah rangkaian prosedur formal dimana data dikelompokan, diproses menjadi informasi dan didistribusikan kepada pemakai. (Abdul Kadir, 2003, Hal 11)

2.4 UML (*Unified Modelling Language*)

Unified Modelling Language **(UML)** adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. (Suhendar, 2002, Hal 1).

UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan 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 merupakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti *C++*, *Java*, dan *Visual Basic*.

2.4.1 Komponen Dasar UML

UML terdiri dari diagram, notasi, konsep dan aturan yang digunakan dalam memodelkan sistem. Diagram UML terdiri dari 9 jenis diagram yang memiliki fungsi dan notasi masing-masing. Kesembilan diagram ini dapat dibagi menjadi 2 kategori, yaitu :

1. Diagram yang menggambarkan struktur yang statis dari sistem.
2. Diagram yang menggambarkan struktur yang dinamis dari sistem.

2.4.2 Diagram Struktur Statis.

Adalah diagram yang menggambarkan struktur hubungan statis dari elemen-elemen yang ada dalam sebuah model diantaranya adalah :

a) *Class diagram*

b) *Statechart diagram*

c) *Component diagram*

d) *Deployment diagram*

2.4.2.1 *Class Diagram*

Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

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 memiliki tiga area pokok :

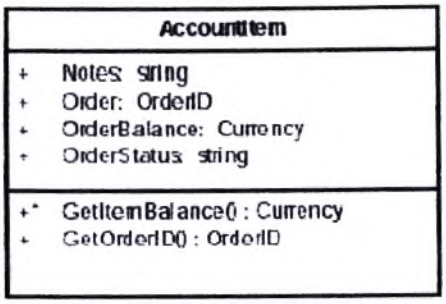
1. Nama (dan stereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan

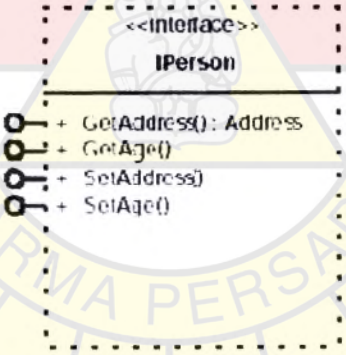
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya

c. *Public*, dapat dipanggil oleh siapa saja.



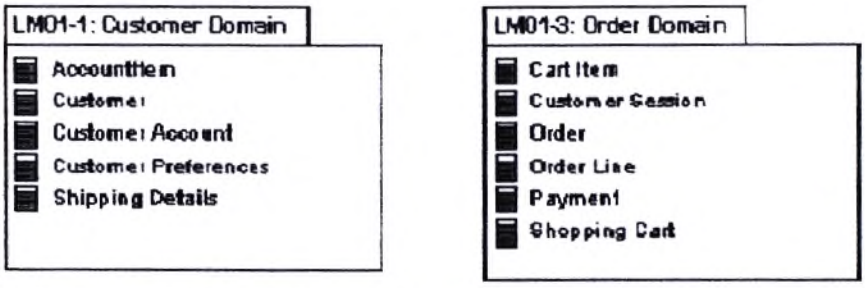
Gambar 2.1 Notasi *class*

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



Gambar 2.2 Notasi *Interface class*

Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.



Gambar 2.3 *Package* dari sebuah *class*

Pada *object diagram* digambarkan hubungan antar elemen dalam model, tapi dengan memakai objeknya, bukan *class*. *Class* ialah kumpulan dari objek-objek yang memiliki *attribute*, *behaviour* atau *operation* yang sama.

Class dan *object* di dalam tahapan *design* digambarkan dengan letak yang memiliki tiga bagian. Pada bagian atas diberi nama *class* atau *object*. Bagian tengah merupakan bagian yang berisi *attribute* yang dimiliki dan bagian bawah berisi *operation*.

Dalam *class* dan *object diagram* tersebut terdapat beberapa istilah-istilah, diantaranya yaitu :

1. *Association Link*

   Merupakan *link* yang mewakili hubungan antar dua objek. *Association* adalah hubungan antar *class* dan mewakili kelompok *link*.

2. *Multiplicity*

   Merupakan banyaknya hubungan yang mungkin terjadi antar *class*.

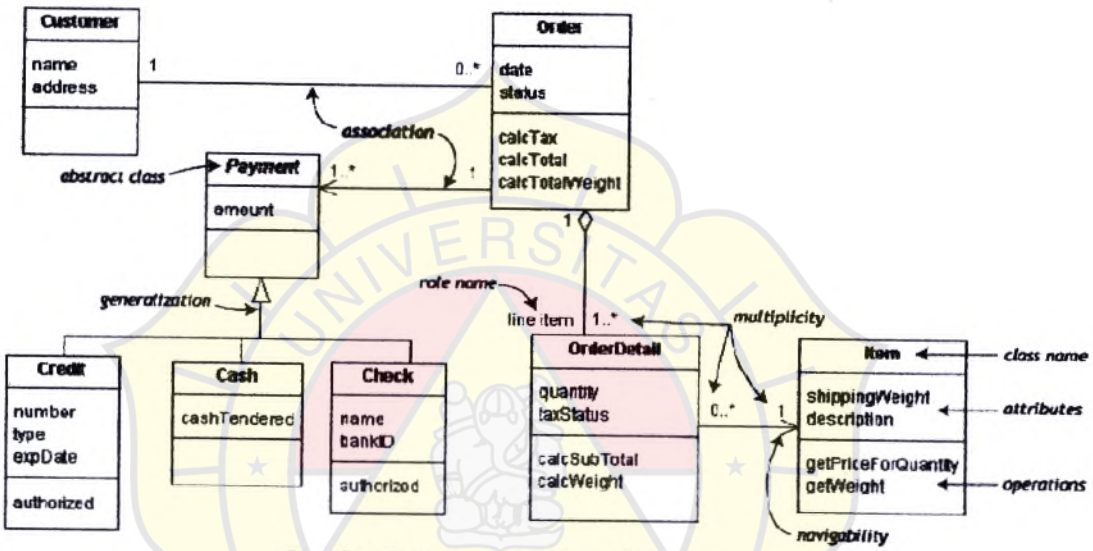
3. *Aggregation*

   Merupakan bentuk khusus dari *association* yang menggambarkan bahwa satu *class* merupakan bagian dari *class* lainnya, "*a part of*". Dalam beberapa kasus, satu *class* dapat terbagi menjadi beberapa *class* lagi.

4. *Generalization*

Merupakan hubungan antara *class* induk (*super class*) dengan *class* anak (*sub class*). Hubungan yang terjadi adalah "*is a*". Pada hubungan generalisasi *attribute* dan *behaviour* yang terdapat pada *super class* akan diwarisi oleh *sub class.*

Berikut adalah contoh sebuah class diagram :



Gambar 2.4 Contoh *Class diagram*

2.4.2.2 *Component Diagram*

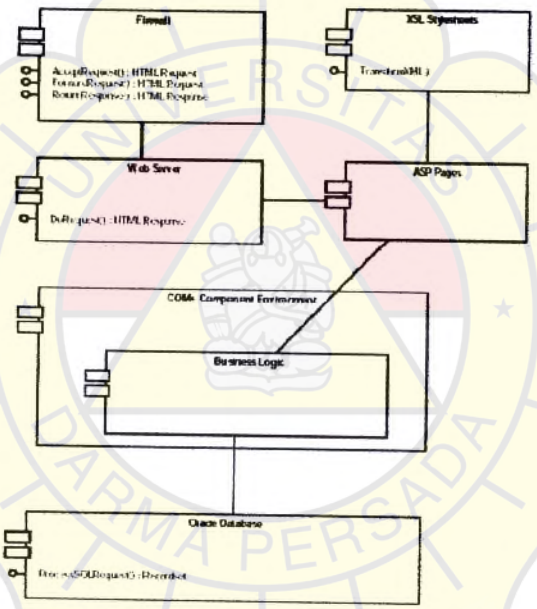
Component Diagram merupakan gambaran aspek fisik sistem berbasis objek dengan menunjukkan hubungan dan ketergantungan dalam serangkaian komponen. Menggambarkan komponen fisik *software* termasuk *source code, run time (binary) code, executable file, table, library,* dan dokumen. Meliputi komponen, *interface, dependency, generalization, association, realization, notes, constraint, packages, subsystem* dari sebuah model.

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.

Diagram ini digunakan untuk memodelkan implementasi sistem yang sifatnya statis sehingga dapat mendukung untuk mengatur konfigurasi dari bagian sistem.

Berikut ini adalah sebuah contoh dari *component diagram* :



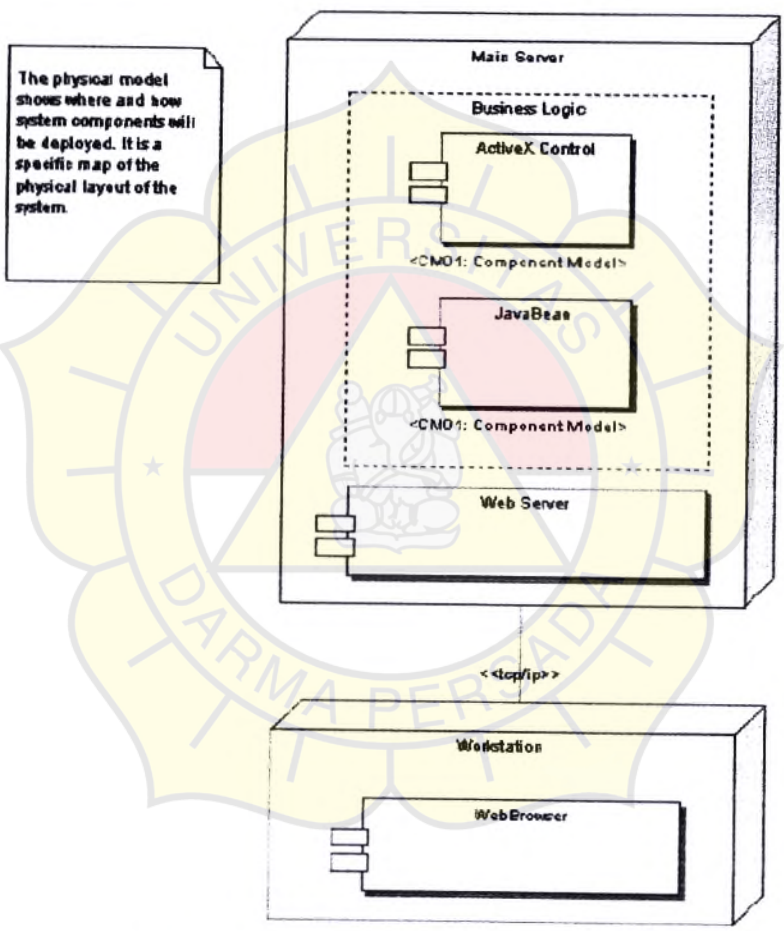
Gambar 2.5 Contoh *Component diagram*

2.4.2.3 *Deployment diagram*

Deployment diagram menggambarkan sumber fisik dalam sistem, termasuk node, komponen dan koneksi (model implementasi sistem yang statistik). Dalam hal ini meliputi topologi *hardware* yang dipakai sistem.

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. Berikut adalah contoh dari *Deployment diagram* :



Gambar 2.6 Contoh *Deployment diagram*

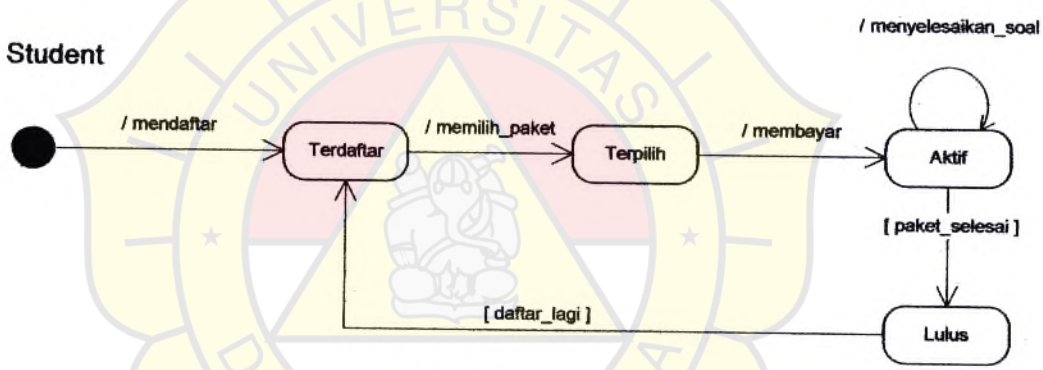
2.4.2.4 *Statechart diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli*

yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.

Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah. Berikut adalah contoh dari *Statechart diagram* :



Gambar 2.7 Contoh *Statechart diagram*

2.4.3 Diagram Struktur Dinamis.

Adalah kumpulan diagram yang menggambarkan hubungan dinamis antara class yang berada dalam komponen model.

2.4.3.1 *Use case diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor

dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya.

Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

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

Use case merupakan salah satu metode dalam analisis dan desain sistem berorientasi objek (*Object Oriented Analysis and Design*). *Use case* juga merupakan bagian dari UML (*Unified Modelling Language*). *Use case modelling* digunakan untuk mendokumentasikan *system behaviour* dan *subsystem* pada saat pengembangan sistem, termasuk di dalamnya fungsi internal suatu sistem (*use case*), pengguna sistem (*user*) dan hubungan interaksi antara keduanya (*use case diagram*).

Use case diwujudkan dalam bentuk diagram dengan beberapa notasi baku

yang ditujukan untuk memudahkan kita melihat keseluruhan *behaviour* dari sebuah sistem. *Use case* tidak hanya digambarkan dalam bentuk diagram saja, namun diwujudkan pula dalam bentuk teks, yang dikenal dengan *narrative use case*, dimana proses yang ada dalam *use case* digambarkan dengan kata-kata sehingga menjadi lebih jelas.

Terdapat 3 bagian utama dalam *use case modeling* sebagaimana dijelaskan berikut ini :

a. *Actor*

Actor sebagai perwujudan dari pengguna sistem, proses dan segala sesuatu yang berinteraksi dalam sistem tersebut. *Actor* tidak termasuk dalam sistem, tetapi dapat menggambarkan interaksi dari *external user* dengan sistem tersebut. Setiap *actor* berinteraksi dengan satu atau lebih *use case* dengan pertukaran pesan atau informasi.

b. *Use Case*

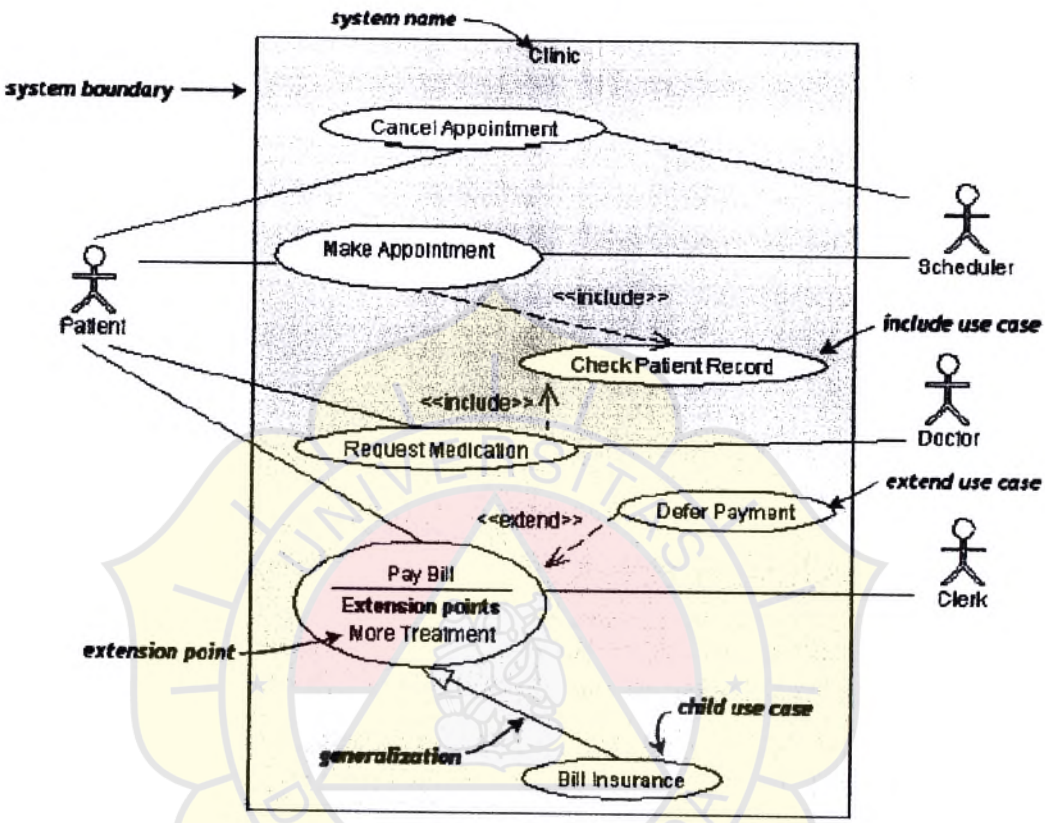
Use case merupakan bagian dari sebuah sistem yang menyediakan sebuah fungsi atau tugas tertentu dan terdiri dari serangkaian aksi, *use case* memperlihatkan *external behaviour* dari sebuah sistem yang dilihat dari segi pengguna eksternal. *Use case* tidak seperti *operation* karena sebuah *use case* dapat terus menerima input dari *actor* pada saat dijalankan, dan *use case* dapat diterapkan pada unit sistem yang lebih kecil seperti subsistem.

c. System Boundary

System boundary menjelaskan batasan suatu sistem dengan lingkungannya, sehingga memberi batasan yang jelas sampai mana suatu sistem bekerja, termasuk

membatasi sistem dengan *actor* yang berada di luar sistem. Di dalam *system boundary* terletak kumpulan *use case* dari sebuah sistem.

Berikut adalah contoh dari *use case diagram* :



Gambar 2.8 Contoh *Use case diagram*

2.4.3.2 *Sequence diagram*

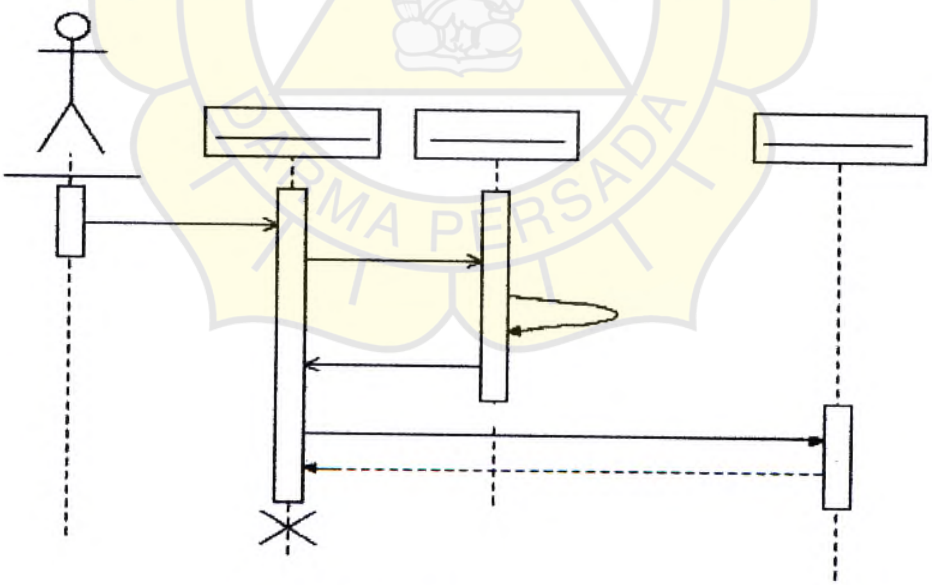
Sequence diagram merupakan diagram yang menggambarkan pola hubungan diantara sekumpulan objek yang saling mempengaruhi menurut urutan waktu. Sebuah objek berinteraksi dengan objek lain melalui pengiriman pesan (*messages*). *Sequence diagram* biasanya digunakan untuk mengilustrasikan sebuah *use case.*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang

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

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.

Berikut adalah contoh notasi dari *Sequence diagram* :



Gambar 2.9 Notasi *Sequence diagram*

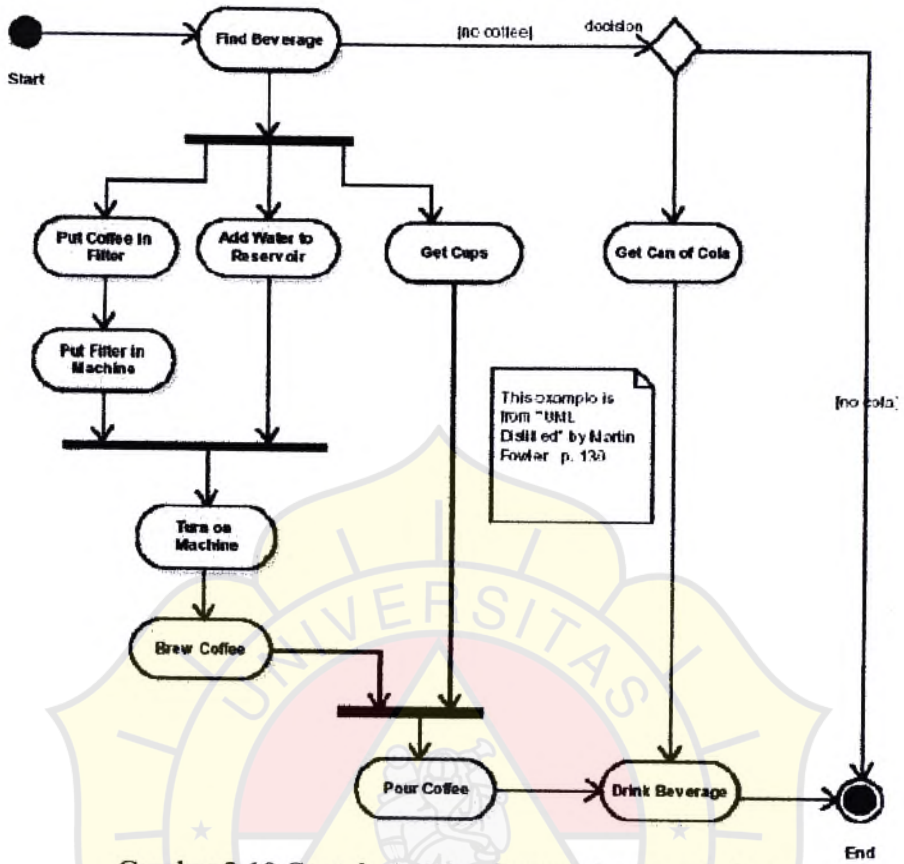
2.4.3.3 *Activity diagram*

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

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

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

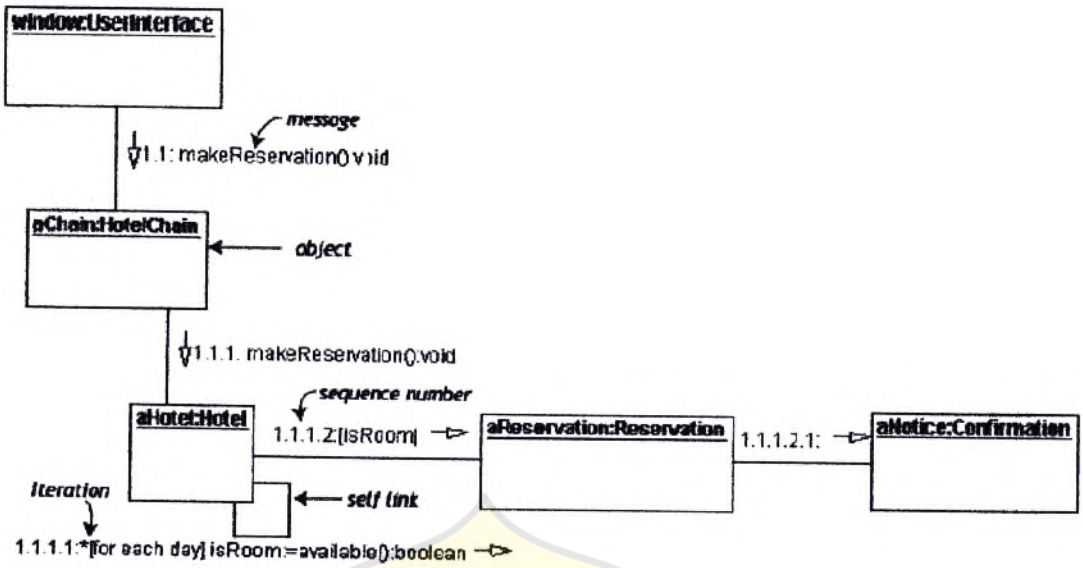
Berikut adalah contoh *Activity diagram* tanpa *swimlane* :



Gambar 2.10 Contoh *Activity diagram*-tanpa *swimlane*

2.4.3.4 *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 2.11 Contoh *Collaboration diagram*

2.5 Pengertian Display Andon

Andon adalah sebuah istilah dalam manufaktur yang mengacu pada sebuah sistem yang memberitahukan management, maintenance dan pekerja lainnya akan adanya masalah kualitas, kerusakan mesin, maupun masalah dalam proses. Titik sentralnya adalah pada sebuah papan/*display* yang menggabungkan lampu sinyal untuk menunjukan adanya line produksi yang memiliki masalah. Peringatan tersebut dapat diaktifkan secara manual oleh seorang pekerja menggunakan tombol.

2.6 Visual Basic 6.0

Visual basic merupakan salah satu aplikasi pemrograman visual yang memiliki bahasa pemrograman yang cukup populer dan mudah untuk dipelajari, basis bahasa pemrograman yang digunakan dalam visual basic adalah bahasa BASIC (*Beginners All-Purpose Symbolic Instruction Code*) yang merupakan

salah satu bahasa pemrograman tingkat tinggi yang sederhana dan mudah dipelajari. (Madcoms, 2008, Hal1).

Visual Basic (yang sering juga disebut dengan VB) selain disebut dengan bahasa pemrograman, juga sering disebut sebagai sarana (tool) untuk menghasilkan program-progam aplikasi berbasiskan windows. Beberapa kemampuan atau manfaat dari Visual Basic diantaranya seperti :

1. Untuk membuat program aplikasi berbasiskan windows.
2. Untuk membuat objek-objek pembantu program seperti misalnya kontrol ActiveX, file Help, aplikasi Internet dan sebagainya.
3. Menguji program (*debugging*) dan menghasilkan program berakhiran EXE yang bersifat executable atau dapat langsung dijalankan,

2.7 PLC (*Programmable Logic Controller*).

Merupakan piranti elektronik kendali logika terprogram yang dirancang untuk dapat beroperasi secara digital dengan menggunakan memori sebagai media penyimpanan instruksi-instruksi internal untuk menjalankan fungsi logika seperti fungsi pencacah, fungsi urutan proses, fungsi waktu, fungsi aritmatika dan fungsi lainnya dengan cara memprogramnya untuk mengontrol mesin atau proses melalui modul-modul I/O digital maupun analog. (Budiyanto, 2006, Hal 1)

Berdasarkan namanya konsep PLC adalah sebagai berikut :

1. *Programmable*, menunjukkan kemampuan dalam hal memori untuk menyimpan program yang telah dibuat yang dengan mudah diubah-ubah fungsi atau kegunaannya.

2. *Logic,* menunjukkan kemampuan dalam memproses input secara aritmatik dan logic yakni melakukan operasi membandingkan, menjumlahkan, mengalikan, membagi, mengurangi, negasi, AND, OR, dan lain sebagainya.
3. *Controller,* menunjukkan kemampuan dalam mengontrol dan mengatur proses sehingga menghasilkan output yang diinginkan.

2.8 *Microsoft Access.*

Merupakan sebuah program aplikasi basis data komputer relasional yang menggunakan mesin basis data *Microsoft jet database engine* dan juga menggunakan tampilan grafis yang intuitif sehingga memudahkan pengguna. *Microsoft access* dapat menggunakan data yang disimpan didalam format *Microsoft access, Microsoft jet database engine, Microsoft SQL server, oracle database,* atau semua kontainer basis data yang mendukung standar ODBC.

2.9 *Photo Electric Switch* (Saklar Sensor).

Merupakan sebuah komponen elektronika yang berfungsi mendeteksi benda yang melewatinya, dalam hal ini hasil produksi yang ditampilkan di *display.* Komponen ini bekerja pada saat benda yang berjalan mengenai atau memutus signal kontak antara receiver dan transmitter pada sensor