BAB II
LANDASAN TEORI

Didalam buku Jogiyanto HM(2005:129) mendefiniskan analisa sistem *(system analysis)* sebagai penguraian dari suatu sistem informasi yang utuh kedalam bagian-bagian komponennya, dengan maksud unutk mengidentifikasikan dan mengevalusai permasalahan-permasalahan, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

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 suatu sasaran tertentu, Jogiyanto HM(2005:1)

Prosedur oleh Ricahrd F. Neuschel adalah suatu urutan-urutan operasi kerikal(tulis menulis), biasanya melibatkan beberapa orang didalam satu atau lebih didalam departemen, yang diterapkan untuk menjamin penanganan yang seragam dari transaksi-transaksi bisnis yang terjadi, Jogiyanto HM(2005:1)

Menurut Prof. Dr. S. Prajudi Atmosudirdjo dalam Tata Sutabri(2005:10), menyatakan bahwa sistem terdiri atas objek-objek, unsur-unsur, dan komponen-komponen yang berkaitan dan berhubungan satu sama lainnya dengan sedemikian rupa sehingga unsur-unsur tersebut merupakan satu kesatuan pemrosesan atau pengolahan yang tertentu.

Sistem dapat didefinisikan dalam dua kelompok yaitu, yang menekankan pada prosedurnya dan yang menekankan pada komponen atau elemen.

Menurut Jery FitzGerald, Ardra F. FitzGerld,Warmen D. Stalling dalam Jugiyanto HM(2005:1-2), pendekatan sistem yang menekankan pada prosedurnya mendefinisikan suatu Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan untuk menyelesaikan suatu sasaran yang tertentu.

2.1.1 Karakteristik Sistem

Menurut Tata Sutabri(2005:12), model umum sebuah sistem terdiri dari input, proses, dan output. Hal ini merupakan konsep sebuah sistem yang sederhana mengingat sebuah sistem dapat mempunyai beberapa masukan dan sekaligus keluaran. Selain itu sistem juga memiliki karakteristik atau sifat-sifat tertentu yang menncirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem adapun karakteristik yang dimaksud sebagai berikut:

1. Komponen Sistem *(components)*

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa berbentuk subsistem. Setiap subsistem memiliki sifat-sifat dari sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar yang disebut dengan Supra sistem.

2. Batasan sistem *(Boundary)*

   Ruang lingkup sistm merupakan daerah yang membatasi antara sistem lainnya atau sistem dengan lingkungan luarnya.

Batasan sistem ini memunkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkungan luar sistem *(Environment)*

   Bentuk apapun yang ada di luar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut dengan lingkungan luar sistem. Lingkungan luar sistem ini dapat menguntungkan dan dapat juga merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi bagi sistem tersebut, yang demikian lingkungan luar tersebut di jaga dan dipelihara. Sedangkan lingkungan luar yang merugikan harus dikendalikan karena kalau tiadak maka akan menggangu kelangsungan hidup sistem tersebut.

4. Penghubung sistem *(interface)*

   Sebagai media yang menghubungkan sistem dengan subsistem yang lain disebut dengan penghubung sistem luar atau *interface*. Penghubung ini memunkinkan sumbeer-sumber daya mengalir dari subsistem ke subsistem yang lain. Keluaran subsistem akan menjadi masukan untuk subsistem yang lain dengan melewati panghubung. Dengan demikian terjadi suatu integrasi sistem yang membentuk satu kesatuan.

5. Masukan sistem *(Input)*

   Energi yang dimasukan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan *(maintenance input)* dan sinyal *(signal input)*, "program adalah *maintenance* input yang digunakan untuk mengoperasikan komputer sementara "data" adalah signal input yang akan diolah menjadi informasi.

6. Keluaran sistem *(Output)*

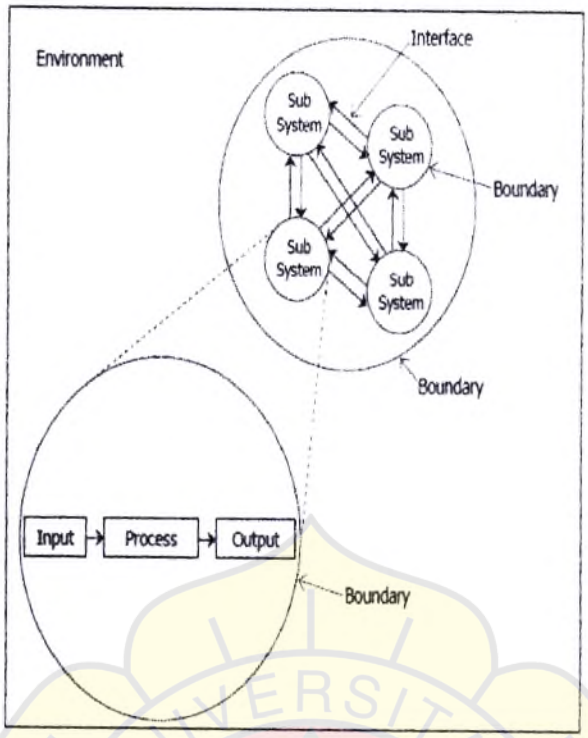
   Hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain. Seperti contoh sistem informasi. Keluaran yang dihasilkan adalah informasi, yang mana informasi ini dapat digunakan sebagai masukan untuk pengambilan keputusan data hal-hal lain yang merupakan input bagi subsistem lainnya.

7. Pengolah sistem *(Process system)*

   Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Sebagai contoh sistem akuntansi. Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran sistem *(objectives)*

   suatu sistem memiliki tujuan dan sasaran yang pasti bersifat deterministik. Kalau suatu sistem tidak memiliki sasaran, maka operasi tida ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan.



***Gambar** 2.2.1 Karakteristik sistem*

2.1.2 Klasifikasi Sistem

Menurut Jogiyanto HM(2005:6-7), sistem dapat diklasifikasikan dari beberapa sudut pandang, diantaranya adalah sebagai berikut :

1. Sistem diklasifikasikan sebagai Sistem Abstrak *(abstract system)* dan Sistem fisik *(physical system)*. Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik. Sistem Fisik merupakan sistem yang ada secara fisik.
2. Sistem diklasifikasikan sebagai Sistem Alamiah *(Nature System)* dan Sistem Buatan Alamia *(Human Made System)*. Sistem Alamiah adalah sistem yang terjadi tanpa adanya campur tangan manusia. Sedngkan Sistem Buatan Manusia adalah sistem kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

3. Sistem diklasifikasikan sebagai Sistem Tertentu *(Deterministic System)* dan Sistem Tak Tentu *(Probabilitas System)*. Sistem Tetentu beroperasi dengan tingkah laku yang sudah dapat diprediksi dan Sistem Tak Tentu adalah sistem yang kondisi masa depannya tidak dapat diperdiksi karena mengandung unsur probabilitas.
4. Sistem diklasifikasikan sebagai Sistem Tertutup *(Closed System)* dan Sistem Terbuka *(Open System)*. Sistem Tetutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya, secar teortis sistem tertutup ini tidak ada tetapi kenyataan ini tidak ada sistem yang benar-benar tertutup yang ada hanyalah *Relativly Closed System* (secara relatif tertutup, tidak benar-benar tertutup). Sistem Terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk lingkungan luar atau subsistem lainnya.

2.2 Definisi Informasi

Informasi adalah data yang telah diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya. Data merupakan bentuk yang masih mentah yang belum dapat bercerita banyak, sehingga perlu diolah lebih lanjut data diolah melalui suatu model untuk dihasilkan informasi, Jogiyanto HM(2005:8).

Didalam bukunya menurut Tata Sutabri(2005:17-18), informasi merupakan proses lebih lanjut dari data yang sudah memiliki nilai tambah. Informasi dapat dikelompokan menjadi 3 bagian, yaitu:

1. Informasi strategis, informasi ini digunakan untuk mengambil keputusan jangka panjang, mencangkup informasi eksernal, rencana perluasan perusahaan dan sebagainya.
2. Informasi taksis, informasi ini dibutuhkan untuk mengambil keputusan jangka menengah, seperti informasi trend pejualan yang dapat dimanfaatkan untul menyusun rencana penjualan.
3. Informasi teknis, informasi ini dibutuhkan untuk keperluan operasional sehari-hari, seperti informasi persediaan barang stock, retur penjualan, dan kas harian.

2.3 Definisi Sistem Informasi

Sistem informasi menurut Jogiyanto HM(2005:11) adalah suatu sistem didalam struktur organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi yang menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.

Sistem informasi dapat didefinisikan sebagai kumpulan elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan untuk mengintegrasikan data, memproses, dan menyimpan serta mendistribusikan informasi, Budi Sutedjo .D.O(2005:12).

Menurut Budi Sutedjo.D.O(2005:17), sistem informasi mempunyai beberapa tingkatan berdasarkan lini manajerial, adapun tingkatan sistem informasi yaitu:

1. Sistem Pemrosesan Transaksi *(transaction processing system)*, merupakan hasul perkembangan dari bentuk kantor elektronik, dimana sebagian dari pekerjaan rutin diotomatisasi termasuk untuk pemrosesan transakasi.

Pada data yang dimasukan merupakan data-data transaksi yang terjadi kemudian data-data tersebut akan diproses untuk menghasilkan informasi yang akurat sesuai dengan kebutuhan.

2. Sistem Infomasi Manajemen(SIM) adalah sebuah kelengkapan pengelolaan dari proses-proses yang menyediakan informasi untuk manajer guna mendukung operasi-operasi dan pembuatan keputusan dalam sebuah organisasi, pada SIM masukan yang diberikan berupa data transaksi yang telah diproses , beberapa data yang asli, model-model pengolahan data, kemudian data-data tersebut akan dijadikan laporan-laporan ringkas.
3. Sistem Penunjang Keputusan(SPK) merupakan peningkatan dari sistem informasi manejemen dengan penyediaan prosedur-prosedur khusus dan pemodelan yang unik yang akan membantu manajer dalam memperoleh alternatif-alternatif keputusan.
4. Sistem Infomasi e-Business dibangun untuk menjawab tantangan pengintegrasian data dan informasi dari proses bisnis berbasis internet.

2.4 *Unified Modeling Language* (UML)

2.4.1 Definsi UML

Menurut Pilone UML *(Unified Modeling language)* merupakan alat komunikasi yang konsisten dalam mensuport para pengembang sistem sebagai gambar perancangan suatu sistem, Herawati.P(7:2011). Beberapa literatur menyebutkan bahwa UML menyediakan sembilan jenis diagram berdasarkan sifatnya yaitu statis dan dinamis antara lain(Herawati.P:2011):

1. Diagram kelas bersifat statis. Diagram ini melihatkan himpunan kelas-kelas, antar muka-antar mukan, kolaborasi-koaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek meskipun bersifat statis sering pula diagram kelas memuat kelas-kelas aktif.
2. Diagram Paket (*Package Diagram),*bersifat statis. Diagram ini memperlihatkan kumpulan-kumpulan kelas-kelas merupakan bagian dari digram komponen.
3. Diagram *Use-Case*, bersifat statis. Diagram ini memperlihatkan himpunan use*-case* dan actor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.
4. Diagram Interaksi dan *sequence*(urutan), bersifat dinamis. Diagram uruta adalah diagram interaksi yang menekankan pada pengiriman dalam suatu waktu tertentu.
5. Diagram Komunikasi (*Communication DiagramI),* berifat dinamis. Diagram sebagai pengganti diagram kolaborasi UML yang menekankan organisasi struktural dari objek-objek yang menerima serta mengirim pesan.
6. Diagram *Statechart* (*Statechart Diagram)*, bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*state)*, transisi, kejadian seta aktifitas. Diagram ini terutama sangat penting untuk memperlihatkan sifat dinamis dari antar muka

(interface), kelas, kolaborasi dan terutama penting pada pemodelan sistem-sistem yang reaktif.

7. Diagram Aktivitas *(Activitty Diagram)*, bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas keaktivitas lainnya dalam suatu sistem. Diaram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan memberi tekanan pada aliran kendali antar objek.
8. Diagram Komponen *(Component Diagram)*, bersifat statis. Diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yang telah ada pada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan kedalam satu atau lebih kelas-kelas, antarmuka-antaumuka, serta kolaborasi-kolaborasi.
9. Diagram Deployment *(Deployment Diagram)*, bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time)*. Memuat simpul-simpul beserta komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi yang dijalankan pada banyak mesin *(distributed computing)*.

Dari diagram UML terdiri dari 9 jenis diagram yang memiliki fungsi dan notasi masing-masing.Kesembilan diagram ini dapat dibagi menjadi 2 kategori, yaitu :

a. Diagram yang menggambarkan struktur yang statis dari sistem.

b. Diagram yang menggambarkan struktur yang dinamis dari sistem

2.4.2 Diagram Struktur Statis Dari Sistem

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

- *Class diagram*
- *Component diagram*
- *Deployment diagram*
- *Statechart 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 :

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan

- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
- *Public*, dapat dipanggil oleh siapa saja.

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

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 :

- *Association Link*

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

- *Multiplicity*

  Merupakan banyaknya hubungan yang mungkin terjadi antar *class.*

- *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.

- *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.*

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.

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.

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 *statechartdiagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechartdiagram*).

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.

2.4.3 Diagram struktur dinamis dari sistem

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 *usecase* 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 :

- *Actor*

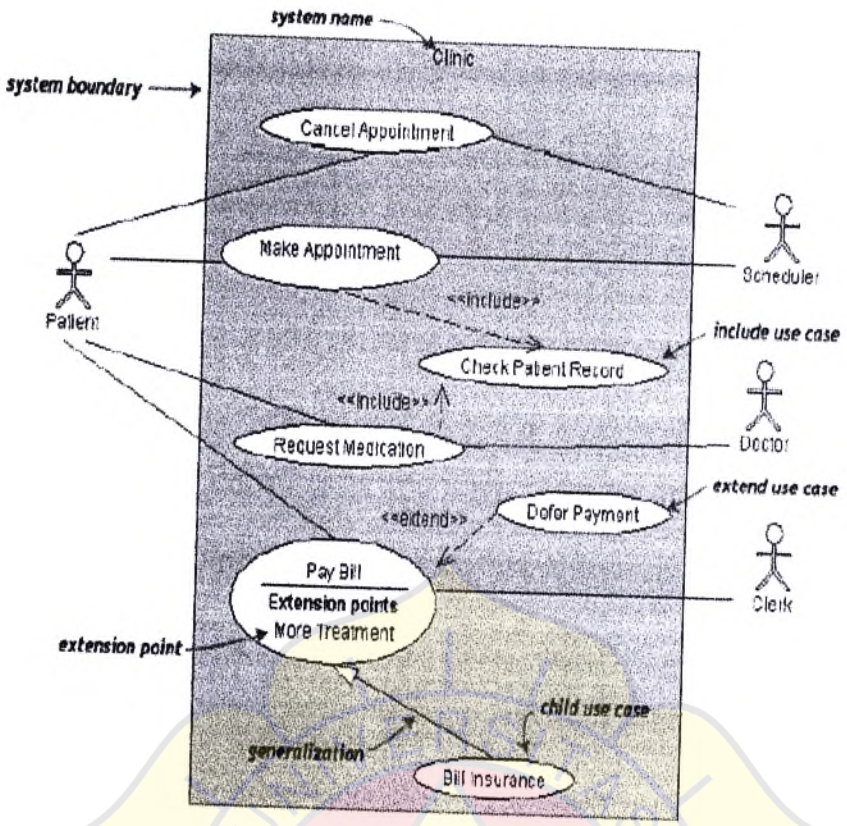
  Actor sebagai perwujudan dari pengguna sistem, proses dan segala sesuatu yang berinteraksi dalam sistem tersebut.*Actor* tidak termasuk dalam sistem, tetapi dapat menggambarkan interaksi dari *external user* dengan sistem tersebut. Setiap *actor* berinteraksi dengan satu atau lebih *use case* dengan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

- *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.

- *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.



Gambar 2.4.3.1 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.

Sequencediagram 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.

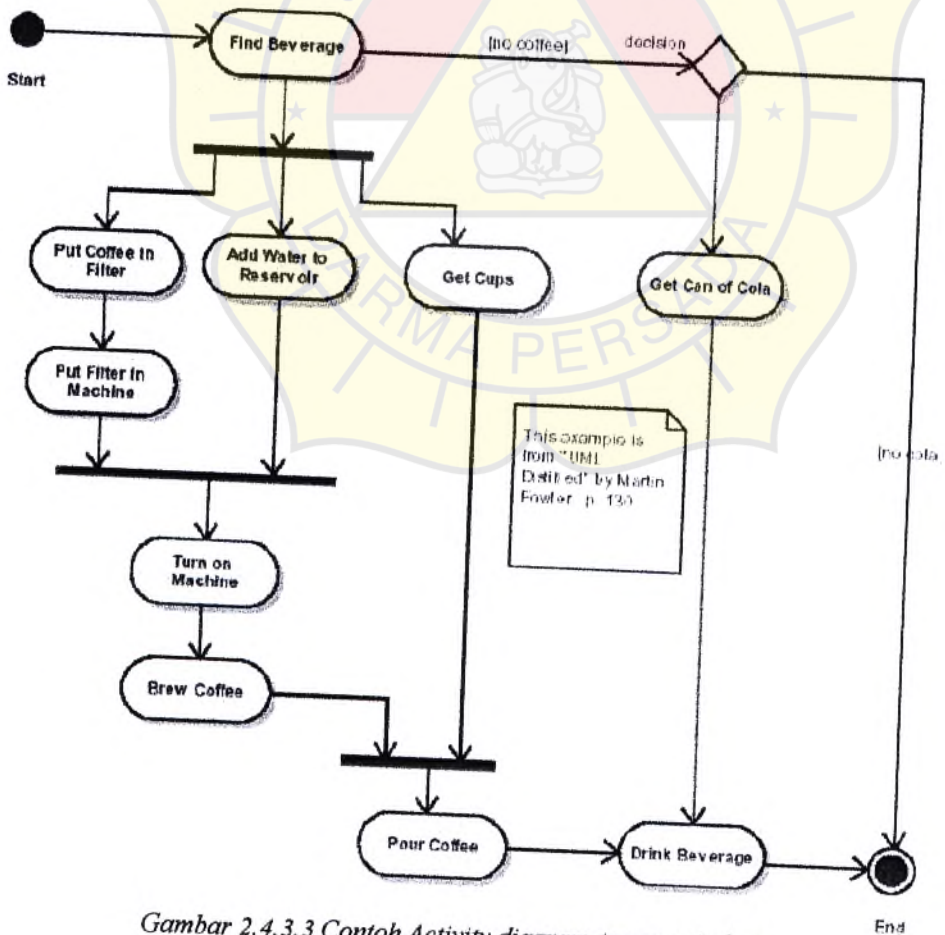
2.4.3.3 *Activity diagram*

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 2.4.3.3 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.

2.5 Konsep dasar pemograman

Menurut Subari (2008:21), kode pemograman adalah serangkaian tulisan perintah yang akan dilaksanakan jika suatu obyek dijalankan. Kode-kode program ini akan mengontrol dan menentukan jalannya suatu obyek.

Bahasa pemograman merupakan sarana komunikasi yang menghubungkan antara manusia dengan komputer. Proses pemograman computer bukan hanya sekedar menulis suatu instruksi yang harus dikerjakan oleh computer tetapi juga bertujuan untuk memecahkan suatu permasalahan serta membuat mudah pekerjaan atau hal-hal lainnya yang ingin digunakan oleh pengguna (*user*). Terdapat 3(tiga) factor penting didalam pemograman :

a. Sintaks adalah aturan penulisan bahasa tersebut(tata bahasanya).
b. Semantik adalah maksud yang terkandung didalam *statement* tersebut.
c. Kebenaran logika adalah berhubungan benar tidaknya urutan *statement*.

Untuk mendukung dalam penyusunan skripsi ini, penulis menggunakan *software* bahasa pemograman :

1. Visual Basic 6.0

Visual basic adalah sebuah bahasa pemrograman komputer.Bahasa pemrograman adalah perintah-perintah atau instruksi yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu.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 :

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

2. Microsoft Access

Suarna Nana(2005:11-12),Microsoft Acces adalah sebuah program aplikasi untuk mengolah data base *(basis data)* model relasional karena terdiri dari lajur kolom dan baris. Selain itu Microsoft Acces merupakan aplikasi program yang sangat mudah dan fleksibel dalam pembuatan dan perancangan sistem manajemen data base.

Microsoft Access saat ini banyak digunakan dalam pembuatan aplikasi program yang sangat sederhana dan mudah, diantara nya seperti :

a. Untuk membuat aplikasi program persediaan barang.

b. Untuk membuat aplikasi program gaji karyawan/pegawai.

c. Untuk membuat aplikasi program penjualan dan pembelian.

d. Untuk membuat aplikasi program administrasi pendidikan.

e. Untuk membuat aplikasi program antirian kunjungan berobat.

f. Dan lain-lain.