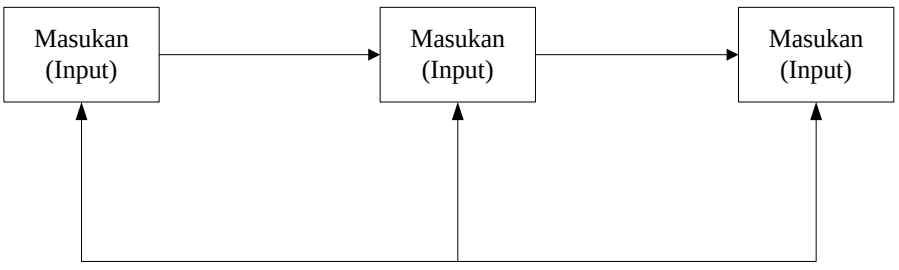
BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi

Sistem dapat diartikan sebagi suatu kumpulan atau himpunan dari unsure datau variabel-variabel yang saling terorganisasi, saling berinteraksi dan saling bergantung satu sama lain. Murdick dan Ross (1993) mendefinisikan sistem sebagai seperangkat elemen yang digabungkan satu dengan lainnya untuk suatu tujuan bersama. Sementara, definisi sistem dalam kamus Webester's Unbriged adalah elemen-elemen yang saling berhubungan dan membentuk satu kesatuan atau organisasi.

Menurut Scott (1996), sistem terdiri dari unsur-unsur seperti masukan (input), pengolahan (prosessing), serta keluaran (output). Cirri pokok sistem menurut Gaspert ada empat, yaitu sistem itu beroperasi dalam suatu lingkungan, terdiri atas unsure-unsur, ditandai dengan saling berhubungan, dan mempunyai satu fungsi atau tujuan utama.

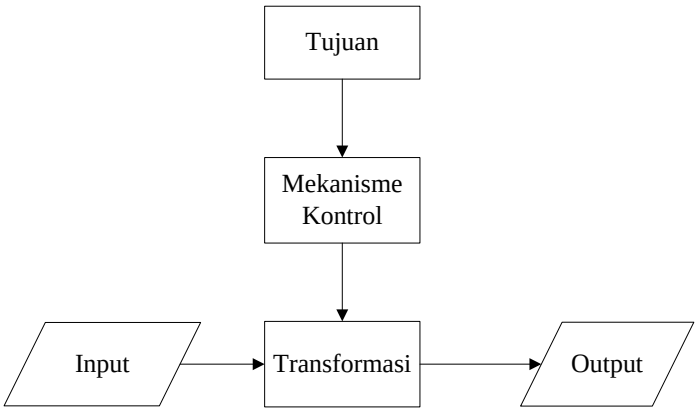


Gambar II.1. Model Sistem

(sumber : Hanif Al Fatta, 2007; 4)

Gambar diatas menunjukan bahwa sistem atau pendekatan sistem minimal mempunyai empat komponen, yakni masukan, pengolahan, keluaran dan balikan atau control.

Sementara Mc. Leod mendefinisikan sistem sebagai sekelompok elemen-elemen yang terintegrasi dengan maksud yangsama untuk mencapai suatu tujuan. Sumber daya mengalir dari elemen output dan untuk menjamin prosesnya berjalan denganbaik maka dihubngkan dengan mekanisme kontrol. Untuk lebih jelasnya elem sistem tersebut dapat digambarkan dengan model sebagai berikut :



Gambar.II.2. Model hubungan elemen-elemen sistem

(sumber : Hanif Al Fatta, 2007; 4)

II.1.2. Sistem Informasi Akuntansi

Akuntansi merupakan bahasa bisnis. Sebagai bahasa bisnis akuntansi menyediakan cara untuk menyajikan dan meringkas kejadian-kejadian bisnis dalam bentuk informasi keuangan kepada pemakainya.

Informasi akuntansi merupakan bagian terpenting dari seluruh informasi yang diperlukan oleh manajemen. Informasi akuntansi yang dihasilkan oleh suatu

sistem dibedakan menjadi dua, yaitu informasi akuntansi keuangan dan informasi akuntansi manajemen.

Pemakai informasi akuntansi pun terdiri dari dua kelompok, yaitu pemakai eksternal dan pemakai internal. Yang dimaksud dengan pemakai ekseternal mencakup pemegang saham, investor, kreditor, pemerintah, pelanggan, pemasok, pesaing, serikat kerja dan masyarakat. Sedangkan pemakai internal adalah pihak manajer dari berbagai tingkatan dalam organisasi bersangkutan.

Sistem Informasi Akuntansi (SIA) dapat didefinisikan sebagai sebuah sistem informasi yang merubah data transaksi bisnis menjadi informasi keuangan yang berguna bagi pemakainya.

Adapun tujuan Sistem Informasi Akuntansi adalah sebagai berikut:

1. mendukung operasi-operasi sehari-hari
2. mendukung pengambilan keputusan manajemen
3. memenuhi kewajiban yang berhubungan dengan pertanggungjawaban

II.1.2. Siklus Sistem Informasi Akuntansi

Sistem Informasi Akuntansi memiliki beberapa sistem-sistem bagian (*sub-system*) yang berupa siklus-siklus akuntansi. Siklus akuntansi menunjukkan prosedur

akuntansi mulai dari sumber data sampai ke proses pencatatan/pengolahan akuntansinya. Siklus akuntansi dibagi menjadi:

1. Siklus pendapatan
2. Siklus pengeluaran kas

3. Siklus konversi
4. Siklus manajemen Sumber Daya Manusia (SDM)
5. Siklus buku besar dan laporan keuangan

Siklus Pendapatan merupakan prosedur pendapatan dimulai dari bagian penjualan otorisasi kredit, pengambilan barang, penerimaan barang, penagihan sampai dengan penerimaan kas. Siklus pengeluaran kas merupakan prosedur pengeluaran kas mulai dari proses pembelian sampai ke proses pembayaran. Siklus konversi merupakan siklus produksi mulai dari bahan mentah sampai ke barang jadi. Siklus manajemen Sumber Daya Manusia melibatkan prosedur penggajian. Siklus buku besar dan pelaporan keuangan berupa prosedur pencatatan dan perekaman ke jurnal dan buku besar dan pencetakan laporan-laporan keuangan yang datanya diambil dari buku besar.

II.1.3. Karakteristik sistem

Untuk memahami atau mengembangkan suatu sistem, maka perlu membedakan unsure-unsur dari sistem yang membentuknya. Berikut adalah karakteristik sistem yang dapat membedakan suatu sistem dengan sistem yang lainya :

1. Batasan (*boundary*) : penggambaran dari suatu elemen atau unsur mana yang termasuk didalam sistem dan mana yang diluar sistem.
2. Lingkungan (*environment)* : segala sesuatu diluar sistem, lingkungan yang menyedikan asumsi, kendala dan input terhadap suatu sistem.

3. Masukan (*input*) : sumber daya (data, bahan baku, peralatan, energy) dari lingkungan yang dikonsumsi dan dimanipulasi oleh suatu sistem.
4. Keluaran (*output*) : sumber daya atau produk (informasi, laporan, dokumen, tampilan layer komputer, barang jadi) yang disediakan untuk lingkungan sistem oleh kegiatan dalam suatu sistem.
5. Komponen (*component*)
6. Penghubung (*interface*)
7. Penyimpanan (*storage*)

II.1.4. Metodologi Pengembangan Sistem

Beberapa alhi membagi proses-proses pengembangan sistem ke dalam sejumlah urutanm yang berbeda-beda. Tetapi semuanya akan mengacu pada proses-proses standar berikut :

a. Analisis
b. Desain
c. Implementasi
d. Pemeliharaan

Pada perkembangannya, proses-proses standar tadi dituangkan dalam metode yang dikenal dengan nama *Systems Depelopment Life Cycle* (SDLC) yang merupakan metodologi umum dalam pengembangan sistem yang menandai kemajuan usaha analisis dan desain. SDLC meliputi fase-fase sebagai berikut :

1. Identifikasi dan seleksi proyek
2. Inisiasi dan perencanaan proyek

3. Analisis
4. Desain
5. Implementasi
    a. Desain logical
    b. Desain fisikal
6. Pemeliharaan

Penjelasan :

1. Identifikasi dan seleksi proyek.

   Merupakan langkah pertama dalam SDLC keseluruhan informasi yang dibutuhkan oleh sistem : analisis, prioritas dan susun ulang. Dalam tahapan ini ada beberapa hal yang harus dilakukan, diantaranya :

   a. Mengidetifikasikan proyek-proyek yang potensial.
   b. Melakukan klasifikasi dan merangking proyek.
   c. Memilih proyek untuk dikembangkan.

2. Inisiasi dan Perencanaan Proyek

   Dalam tahapan ini, proyek SI yang potensial dijelaskan dan argumentasi untuk melanjutkan proyek dikemukakan. Rencana kerja yang matang juga sdisusun untuk menjalankan tahapan-tahapan lainya. Pada tahap ini ditentukan secara detil rencana kerja yang harus dikerjakan, durasi yang diperlukan masing-masing tahap, sumber daya manusia, perangkat lunak , perangkaty keras maupun financial diestimasi. Biasanya hal-hal tadi dituangkan dalam jadwal pelaksanaan proyek.

3. Tahapan Analisis

   Tujuan utama dari fase analisis adalah untuk memahami dan mendokumentasikan kebutuhan bisnis (*business need*) dan persyaratan dari sistem yang baru. Ada enam aktifitas utama dalam fase ini :

   a. Pengumpulan Informasi.
   b. Mendefinisikan sistem *requirement*
   c. *Memprioritaskan kebutuhan*
   d. *Menyusun dan mengevaluasi alternative*
   e. Mengulas kebutuhan dengan pihak manajemen

4. Tahapan Desain

   Tahapan desain adalah tahapan mengubah kebutuhna yang masih berupa konsep menjadi spesifikasi yang riil. Tahapan desain sistem dapat dibagi 2 tahap, yaitu desain logis (*logical desain*) dan tahpan desain fisik (*physical desain*).

5. Implementasi

   Pada tahapan kelima SDLC ini terdapat beberapa hal yang perlu dilakukan, yaitu :

   a. Testing, yaitu menguji hasil kode program yang telah dihasilkan dari tahapan fisik.
   b. Instalasi, setelah program lulus ujicoba, maka perangkat lunak dan perangkat keras akan diinstal pada organisasi perusahaan klien dan secara resmi mulai digunkan untuk menggantikan sistem lama.

6. Pemeliharaan

   Langkah akhir dari SDLC dimana pada tahapan in isistem secara sistematis diperbaiki dan ditingkatkan. Hasil dari tahapan in versi baru dari perangkat lunak yang telah dibuat.

II.2. Bunga Pinjaman

Bunga adalah imbalan jasa atas pinjaman uang. Imbalan ini merupakan suatu kompensasi kepada pemberi pinjaman atas manfaat kedepan dari uang pinjaman tersebut apabila diinvestasikan. Jumlah pinjaman tersebut disebut pokok utang (*principal*). Persentase dari pokok utang yang dibayarkan sebagai imbalan jasa (bunga) dalam satu periode disebut suku bunga. (*Nurul Diena Novalia, 2008 ; 1*)

Suku bunga pinjaman adalah besarnya persentase bunga pinjaman yang harus dibayarkan dari sejumlah uang yang dipinjam. Pada dasarnya ada dua jenis suku bunga yang berlaku dimasyarakat, yaitu suku bunga efektif dan suku bunga nominal. Suku bunga nominal adalah bilangan atau angka yang digunakan untuk menjelaskan tingkat suku bunga tahunan yang berlaku umum. Sedangkan suku bunga efektif adalah nilai aktual dari tingkat suku bunga tahunan yang dihitung pada akhir periode yang lebih pendek dari satu tahun dengan suku bunga majemuk (*Nurul Diena Novalia, 2008 ; 3*)

II. 4. Visual Basic 2010

Visual Basic 2010 merupakan lingkungan pengembangan terintegrasi atau biasa disebut IDE yang dikembangkan berbasis bahasa pemrograman *BASIC*. Bahasa *BASIC* sendiri sebenarnya sudah lama dikembangkan oleh *Microsoft Corporation* dengan nama *Microsoft Quick Basic*. Kesederhanaan sintaks dan fleksibelitas Bahasa Basic yang menyebabkan bahasa pemrograman in berlaku fenomenal sehingga banyak disukai dan pakai oleh *programmer* diseluruh dunia.

Berbekal kepopuleran tersebut *Microsoft* mengembangkan bahasa Basic in menjadi produk yang sangat terkenal di kalangan *programmer*, yaitu mulai *Microsoft Visual Basic 6.0* sampai sekarang, yaitu *Microsoft Visual Basic 2010*. Perkembangan teknologi dan penambahan banyak sekali fitur pada *Visual Basic 2010* tidak mengakibatkan adanya perubahan sintaks-sintaks dasar yang terdapat didalamnya. Sehingga dapat dikatakan untuki menjadi *programmer Visual Basic 2010* yang sebenarnya, anda diharuskan menguasai dan mengerti bagaimana menggunakan dan mengimplementasikansintaks dasar dalam bahasa *basic*. (*Wahana Komputer, 2010 ; 36*)

II. 5. Basis Data

Istilah basis data banyak menimbulkan interprestasi yang berbeda. Pada saat maraknya perangkat lunak dBASE II dan dBase II plus. Sebuah berkas (dengan ekstensi .DBF) biasa disebut basis data. Istilah yang tidak tepat ini meskipun telah merasuk ke sejumlah pemrograman, akhirnya diluruskan kembali oleh pencipta perangkat lunak bais data yang lain. Chou mendefinisikan basis data

sebagai kumpulan informasi bermanfaat yang diorganisasikan ke dalam tatacara yang khusus. Menurut Fabbri dan Schwab, basis data adalah sistem berkas terpadu yang dirancang terutama untuk meminimalkan pengulangan data. Menurut Date, basis data dapat dianggap sebagai tempat untuk sekumpulan berkas data terkomputerisasi.

Menurut Date, sistem basis data pada dasarnya sistem terkomputerisasi yang tujuan utamanya adalah memelihara informasi dan membuat informasi tersebut tersebut tersedia saat dibutuhkan. (Abdul Kadir; 2006; 9)

II. 5. 1. Kamus Data

Karena DBMS menyimpan kumpulan beberapa item data yang terpisah yang dapay digunakan pemakai pada beberapa aplikasi secara bersama-sama, adalah penting bahwa beberapa mekanisme digunakan untuk menyimpankan informasi mengenai mengenai beberapa item data bersangkutan. Itu adalah fungsi dari kamus data.

Kamus data adalah suatu file yang terpisah yang menyimpan informasi seperti :

a. Nama setiap item/jenis/kolom data.
b. Struktur data untuk tiap item.
c. Program yang menggunakan tiap item
d. Tingkat keamanan untuk setiap item

(Sumber : Zulkifli A.M; 2005; 382)

II. 5. 2. Normalisasi

Normalisasi adalah suatu proses untuk membuat data yang tidak normal menjadi data yang normal. Bentuk data yang tidak normal/ data mentah biasa disebut juga *Unnormalized Form*. Tujuan dari normalisasi adalah :

1. Menghindari inkonsistensi data
2. Menghindari terjadinya redudancy data.

(Sumber : Ema Utami dan Sukrisno; 2005; 73)

Pada dasarnya terdapat 7 level normalisasi , yaitu sebagai berikut :

1. First Normal Form (1NF)

   Suatu tabel dikatakan dalam keadaan *First Normal Form* (1NF) jika :

   a. Tidak ada perulangan record data dalam tabel
   b. Setiap sel memiliki satu nilai saja. Artinya tidak ada perulangan *group* dan *array*.
   c. Data yang diinputkan memiliki tipe yang sama dengan tipe data kolom dalam tabel.

2. Second Normal Form (2NF)

   Suatu tabel dikatakan dalam keadaan *Second Normal Form* (2NF) jika tabel tersebut sudah dalam keadaan First Normal Form (1NF) dan jika semua atribut yang bukan kunci tabel, baik *primary key* maupun *foreign key* tergantung pada semua kunci dalam tabel.

3. Third Normal Form (3NF)

   Suatu tabel dikatakan dalam keadaan *Third Normal Form* (3NF) jika tabel tersebut dalam keadaan Second Normal Form (2NF) dan jika tidak

terdapat ketergantungan yang transif. Artinya data –data yang mungkin diisikan berulang-ulang dapat dibuat sebuah tabel baru.

4. Fourth Normal Form (4NF)

   Suatu tabel dikatakan dalam keadaan Fourth Normal Form (4NF) jika tabel tersebut dalam keadaan *Boyce-Codd Normal Form* (BCNF) dan jika tidak terdapat ketergantungan nilai ganda.

5. Fiveth Normal Form (5NF)

   Suatu tabel dikatakan dalam keadaan Fiveth Normal Form (5NF) jika tabel tersebut dalam keadaan Fourth Normal Form (4NF) dan jika setiap ketergantungan dalam join pada tabel sudah konsekuen dengan kunci kandidat pada tabel tersebut.

   (Sumber : Ema Utami dan Sukrisno; 2005; 75-76)

II. 5. 3. ERD (*Entity Relational Database*)

ERD adalah gambar atau diagram yang menunjukan informasi dibuat, disimpan dan digunakan dalam sistem bisnis. Entitas biasanya menggambarkan jenis informasi yang sama. Dalam entitas digunakan untuk menghubungkan antar entitas digunakan untuk menghubungkan antar entitas yang sekaligus menunjukan hubungan antar data. Pada akhirnya ERD bisa juga digunakan untuk menunjukan aturan-aturan bisnis yangada pada sistem informasi yang akan dibangun. (Sumber : Hanif Al-Fattah : 2007; 121 -122)

II. 5. 4. MySQL Database

MySQL pertama kali dirintis oleh seorang programmer database bernama Michael Wildenius. MySQL server adalah RDBMS (*Relational Database Management System*) yang dapay menangani data yang bervolume besar. Meskipun begitu, tida menuntut *resource* yang besar. MySQL adalah database yang paling popular diantara database-database lain.

MySQL adalah program database yang mampu mengirim dan menerima data dengan sangat cepat dan multi *user*. MySQL memiliki dua bentuk license, yaitu *free software* dan *shareware*. (*Wahana Komputer, 2005 ; 5*)

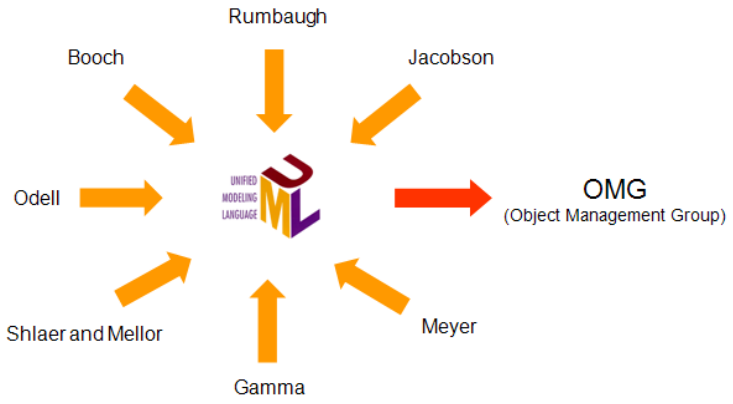
II.6. UML *(Unified Modelling Language)*

UML (*Unified Modelling Language*) adalah ssalah satu alat bantu yang sangat handal didalam dunia pengembangan sistem yang berorientasi obyek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

UML merupakan kesatuan dari bahasa pemodelan yang dikembangan oleh Booch, *Object Modelling Techniqu* (*OMT*) dan *Object Oriented Software Engineering* (OOSE). Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadi proses analisis dan desain ke dalam empat tahapan iterative, yaitu : identifikasi kelas-kelas dan

obyek-obyek, identifikasi semantic dari hubungan obyek dan kelas tersebut, perincian interface dan implementasi. Keunggulan metode Booch adalah pada detil dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan *entity-relationship.* Tahapan utama dalam metodologi ini adalah analisis, design sistem , design obyek dan implementasi. (Munawar, 2005 ; 17 - 19)

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambahkan dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspesif dan seragam daripada metode lainnya. Gambar berikut adalah unsur-unsur yang membentuk UML.



Gambar II.4. UML Sebagai Bahasa Standar Pemodelan Aplikasi OOP

Sumber : (Sri Dharwiyanti: 2006 ; 3)

II.6.1. Konsepsi Dasar UML

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya konsepsi dasar UML bisa kita rangkumkan dalam gambar dibawah.

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Gambar.II.5. Konsep Dasar UML

Sumber : (Sri Dharwiyanti, 2006 ; 3)

Seperti juga tercantum pada gambar diatas UML mendefinisikan diagram-diagram sebagai berikut:

1. use case diagram
2. class diagram
3. statechart diagram
4. activity diagram
5. sequence diagram
6. collaboration diagram
7. component diagram
8. deployment diagram

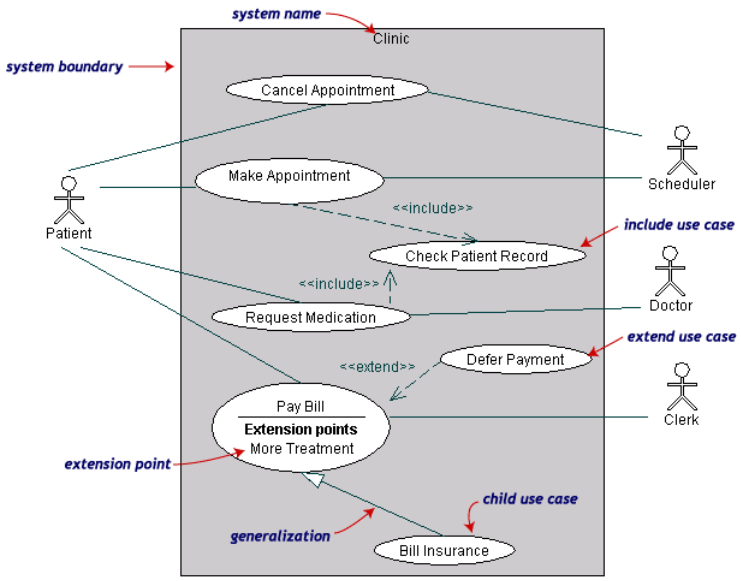
Dalam pembuatan skripsi ini penulis menggunakan diagram Use Case yang terdapat di dalam UML. Adapun maksud dari Use Case Diagram diterangkan dibawah ini.

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 sistem 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.

Contoh *use case diagram* :



Gambar.II.6.Use Diagram

Sumber : (Sri Dharwiyanti, 2006 ; 5)

2. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

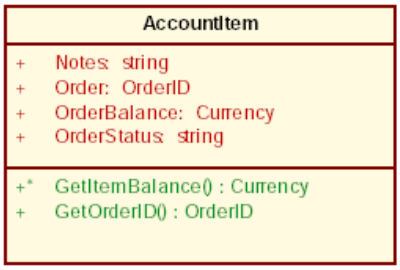
Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

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



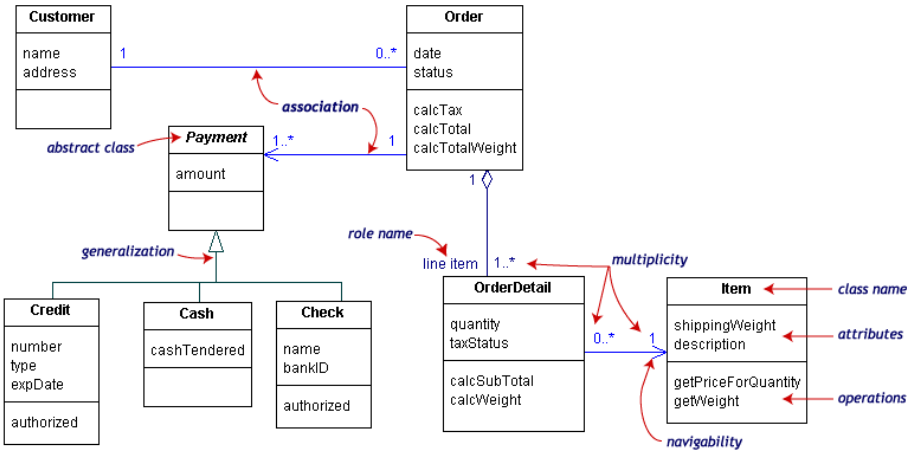
Gambar.II.7.Class Diagram

Sumber : (Sri Dharwiyanti, 2006 ; 5)

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*. (Sri Dharwiyanti, 2005; 5)

a. Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.



Gambar.II.8.Hubungan antar Class

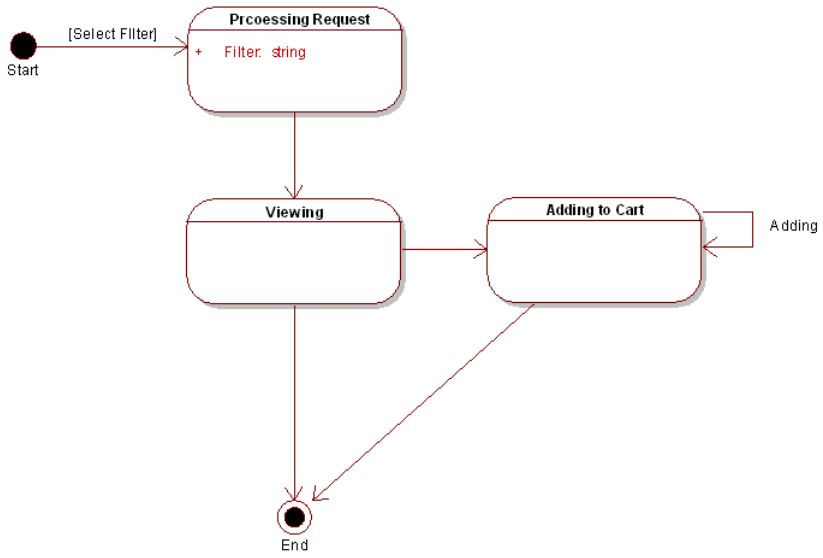
Sumber : (Sri Dharwiyanti, 2006 ; 6)

3. 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.



Gambar.II.9. State Diagram

Sumber : (Sri Dharwiyanti, 2006 ; 7)

4. Activity Diagram

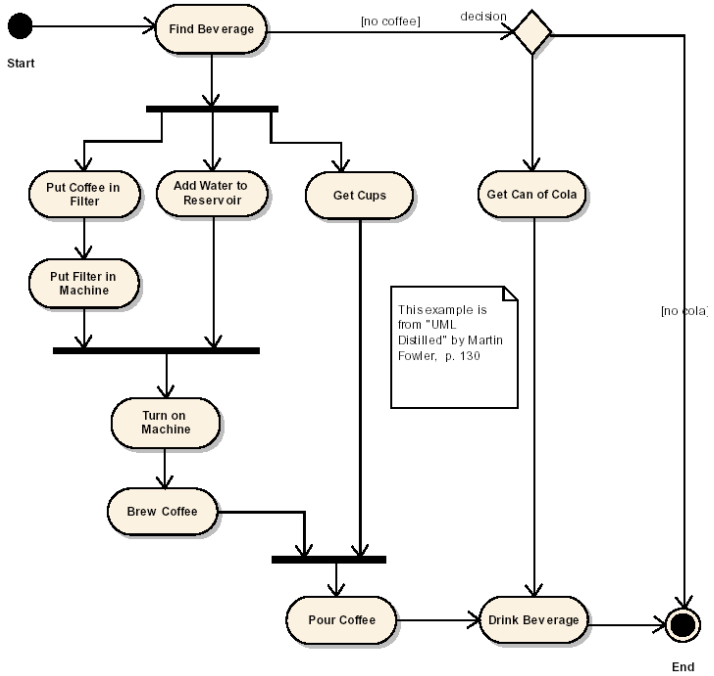
Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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.



Gambar.II.10. Contoh *Activity Diagram* State Diagram

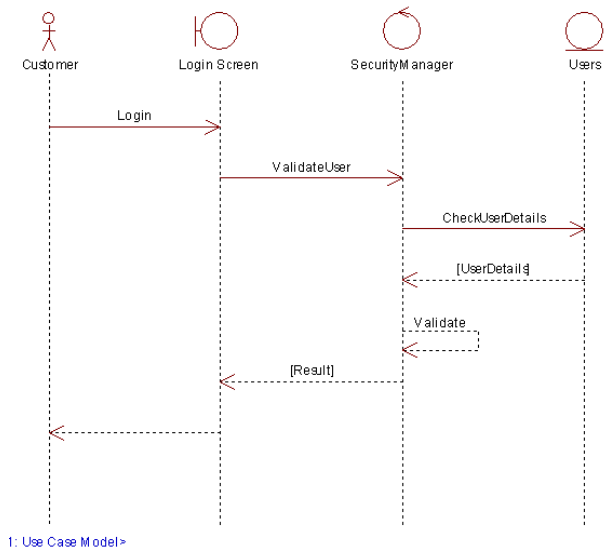
Sumber : (Sri Dharwiyanti, 2006 ; 7)

5. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai 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. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity*.



Gambar.II.11. Contoh *Activity Diagram* State Diagram

Sumber : (Sri Dharwiyanti, 2006 ; 8)