BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem merupakan salah satu yang terpenting dalam sebuah perusahaan yang dapat membentuk kegiatan usaha untuk mencapai kemajuan dan target yang dibutuhkan. Defenisi tentang sistem cukup banyak, untuk mengetahui lebih jelasnya tentang defenisi sistem ini diambil beberapa pernyataan dari beberapa ahli berikut ini:

Sistem adalah Entitas yang terdiri dari dua komponen atau lebih komponen atau sub sistem yang berinteraksi untuk mencapai suatu tujuan. Perlengkapan dan program yang terdiri dari instalasi komputer lengkap. Program dan prosedur terkait yang menjalankan suatu tugas dalam sebuah komputer (B.Romney, Dkk; 2006: 473).

II.3. Sistem Informasi

Sistem Informasi adalah sekumpulan *hardware, software, brainware, prosedure* atau aturan yang diorganisasikan secara integral untuk mengolah data menjadi informasi yang bermanfaat guna memecahkan masalah dan pengambilan keputusan.

Sistem Informasi adalah cara teratur untuk mengumpulkan, memproses, mengelola, dan melaporkan informasi agar organisasi dapat mencapai tujuan dan sasarannya (B.Romney, Dkk; 2006: 473).

Ada 2 (dua) jenis Sistem informasi yaitu:

1. Sistem informasi formal memiliki tanggung jawab jelas untuk memproduksi informasi.
2. Sistem informasi informal adalah sistem yang muncul dari adanya kebutuhan yang tidak dipuaskan oleh saluran formal. Sistem ini berjalan tanpa adanya penugasan formal tanggung jawab.

II.3. Sistem Informasi Akuntansi

Sistem Informasi Akuntansi adalah sekumpulan *hardware, software, brainware, prosedure* atau aturan yang diorganisasikan secara integral untuk mengolah data menjadi informasi yang bermanfaat guna memecahkan masalah dan pengambilan keputusan(Marshall B.Romney, Dkk; 2006: 473).

Sistem Informasi Akuntansi (SIA) terdiri dari lima (5) komponen yaitu:

1. Orang-orang yang mengoperasikan sistem tersebut dan melaksanakan berbagai fungsi.
2. Prosedur-prosedur baik manual maupun yang terotomatisasi, yang dilibatkan dalam mengumpulkan, memproses dan menyimpan data tentang aktifitas-aktivitas organisasi.
3. Data tentang proses-proses bisnis organisasi.
4. *Software* yang dipakai untuk memproses dan organisasi.

5. *Infrastruktur teknologi informasi*, termasuk komputer, peralatan pendukung (*Peripheral device*) dan peralatan untuk komunikasi jaringan.

Kelima komponen ini secara bersama-sama memungkinkan suatu SIA memenuhi tiga fungsi pentingnya dalam organisasi, yaitu:

1. Mengumpulkan dan menyimpan data tentang aktivitas-aktivitas yang dilaksanakan oleh organisasi, sumber daya yang dipengaruhi oleh aktivitas-aktivitas tersebut, dan para pelaku yang terlibat dalam berbagai aktivitas tersebut, agar pihak manajemen, dan pihak-pihak luar yang berkepentingan dapat meninjau ulang (*review*) hal-hal yang telah terjadi.
2. Mengubah data menjadi informasi yang berguna bagi pihak manajemen untuk membuat keputusan dalam aktivitas perencanaan, pelaksanaan, dan pengawasan.
3. Menyediakan pengendalian yang memadai untuk menjaga aset-aset organisasi, termasuk data organisasi, untuk memastikan bahwa data tersebut tersedia dibutuhkan, akurat dan andal.

Mempelajari SIA adalah hal yang penting dalam akuntansi, Dalam *Statement Of Financial Accounting Concepts no.2, Financial Accounting Standards Board* mendefenisikan akuntansi sebagai sistem informasi. Di dalam standar akuntansi keuangan tersebut juga disebutkan bahwa tujuan utama akuntansi adalah untuk menyediakan informasi yang berguna bagi para para pengambil keputusan. Oleh sebab itu, bukanlah hal yang mengherankan apabila *accounting education change commision* merekomendasikan bahwa kurikulum akuntansi harus menekankan bahwa akuntansi adalah suatu proses identifikasi, pengembangan, pengukuran dan

komunikasi informasi. Komisi tersebut menyarankan agar kurikulum akuntansi harus dirancang untuk memberi para mahasiswa sebuah pemahaman yang kuat atas tiga komponen dasar berikut:

1. Pemakaian informasi didalam pengambilan keputusan
2. Sifat, desain, pemakaian dan implementasi SIA
3. Pelaporan informasi keuangan

SIA juga merupakan aktivitas pendukung, jadi SIA dapat menambah nilai bagi organisasi dengan cara memberikan informasi yang akurat dan tepat waktu agar kelima aktivitas utama rantai nilai dapat dilaksanakan dengan lebih efektif dan efisien. SIA yang dirancang dengan baik dapat melakukan hal ini dengan cara :

1. Memperbaiki kualitas dan mengurangi biaya untuk menghasilkan produk atau jasa.
2. Memperbaiki efisiensi, SIA yang dirancang dengan baik dapat membantu memperbaiki efisiensi jalannya suatu proses dengan memberikan informasi yang tepat waktu.
3. Memperbaiki pengambilan keputusan, SIA dapat memperbaiki pengambilan keputusan dengan memberikan informasi dengan tepat waktu.
4. Berbagi pengetahuan, SIA yang dirancang dengan baik bisa mempermudah proses berbagi pengetahuan dan keahlian yang selanjutnya dapat memperbaiki proses operasi perusahaan dan bahkan memberikan keunggulan kompetetif.

Didalam organisasi SIA yang dirancang dengan baik juga dapat membantu meningkatkan laba organisasi dengan memperbaiki efisiensi dan efektivitas rantai persediaanya. (Marshall B.Romney, Dkk; 2006: 10).

II.4. Persediaan Perpetual

Sistem persediaan perpetual adalah suatu sistem akuntansi persediaan yang menggunakan catatan secara kontinu mengungkapkan besarnya persediaan (Soemarso S.R ; 2004: 366)

Dalam sistem persediaan perpetual, semua kenaikan dan penurunan barang dagang dicatat dengan cara yang sama seperti mencatat kenaikan dan penurunan kas. Pembelian dicatat dengan mendebit Persediaan Barang Dagang dan mengkredit kas atau utang usaha.

Persediaan Perpetual ada tiga (3) komponen diantaranya:

II.4.1. Metode FIFO (First-In, First-Out)

Metode Fifo merupakan suatu perhitungan persediaan barang dimana barang pertama masuk dan pertama keluar, jadi metode fifo dapat dikatakan konsisten dengan arus fisik atau pergerakan barang dagang.

Metode FIFO memberikan hasil-hasil yang sama dengan yang diperoleh melalui pengidentifikasian biaya khusus setiap item yang dijual dan ada dalam persediaan. Dengan menggunakan metode FIFO, biaya dimasukan dalam harga pokok penjualan sesuai dengan urutan terjadinya biaya itu. Sebagai ilustrasi:

Peraga memperlihatkan ayat jurnal untuk pembelian dan penjualan persediaan item 127B. Jumlah unit dalam persediaan setelah setiap transaksi, beserta biaya total dan biaya per unit, diperlihatkan dalam akun dimaksud. Kita

mengansumsikan bahwa semua unit dijual secara kredit masing-masing dengan harga $30.

Perhatikan bahwa setiap 7 unitdijual pada tanggal 4 januari, masih tersisa persediaan sejumlah tiga unit @$20. Sebanyak 8 unit yang dibeli pada tanggal 10 januari harganyaadalah $21 per unit, bukan $20. Oleh karena itu,persediaan setelah pembeliaan tanggal 10 januari dilaporkan dalam dua baris, yaitu 3 unit. (Soemarso S.R ; 2004: 366)

Peraga ayat jurnal dan akun Persediaan Perpetual (FIFO):

Tabel II.1. Metode Perpetual dengan Menggunakan Metode FIFO

Item 127B									
	Pembelian			**Harga Pokok Penjualan**			**Persediaan**		
Tanggal	Kuantitas	Biaya per Unit	Total biaya	Kuantitas	Biaya per Unit	Total biaya	Kuantitas	Biaya per Unit	Total biaya
1 jan							10	20	200
4				7	20	140	3	20	60
10	8	21	168				3	20	60
							8	21	168
22				3	20	60			
				1	21	21	7	21	147
28				2	21	42	5	21	105
30	10	22	220				5	21	105
							10	22	220

Sumber : (Soemarso S.R : 2004)

II.4.2. Metode LIFO (Last-In, First-Out)

Metode Lifo merupakan suatu perhitungan persediaan barang dimana barang terakhir masuk dan pertama keluar, Jika sebuah perusahaan menggunakan metode Lifo dalam persediaan perpetual, maka biaya dari unit yang dijual merupakan biaya pembelian paling akhir.

Sebagai ilustrasi:

Peraga memperlihatkan ayat jurnal untuk pembelian dan penjualan persediaan item 127B. Jumlah unit dalam persediaan setelah setiap transaksi, beserta biaya total dan biaya per unit, diperlihatkan dalam akun dimaksud yang dibuat berdasarkan LIFO. Jika anda membandigkan akun buku besar pada sistem perpetual FIFO dengan sistem perperpetual LIFO, anda akan menemukan bahwa akun-akun itu massih tetap sama sampai pembelian tanggal 10 januari.

Namun menurut LIFO, harga pokok dari 4 unit yang dijual pada tanggal 22 januari adalah harga pokok per unit dari pembelian tanggal 10 januari ($21per unit). Harga pokok 7 unit dalam persediaan setelah penjualan tanggal 22 januari adalah harga pokok 3 unit yang tersisa dari persediaan awaldan harga pokok 4 unit yang tersisa dari pembelian tanggal 10 januari. Sisa ilustrasi LIFO berikutnya dijelaskan dengan cara yang sama (Soemarso S.R ; 2004: 367).

Peraga ayat jurnal dan akun Persediaan Perpetual (LIFO):

Tabel II.2. Metode Perpetual dengan menggunakan Metode LIFO

Item 127B									
	Pembelian			**Harga Pokok Penjualan**			**Persediaan**		
Tanggal	Kuantitas	Biaya per Unit	Total biaya	Kuantitas	Biaya per Unit	Total biaya	Kuantitas	Biaya per Unit	Total biaya
1 jan							10	20	200
4				7	20	140	3	20	60
10	8	21	168				3	20	60
							8	21	168
22				4	21	84	3	20	60
							4	21	84
28				2	21	42	3	20	60
							2	21	42
30	10	22	220				3	20	60
							2	21	42
							10	22	`220

Sumber :(Soemarso S.R :2004)

II.4.3. Metode Avarage (Rata-rata)

Metode biaya rata-rata digunakan dalam sistem persediaan perpetual, biaya rata-rata per unit untuk masing-masing item dihitung setiap kali pembelian dilakukan. Untuk menentukan harga pokok setiap penjualan sampai pembelian berikutnya dilakukan dengan rata-rata baru dihitung (Soemarso S.R ; 2004: 368).

II.5. *Visual Basic.Net 2008*

Visual Basic berawal dari bahasa BASIC yang dikembangkan mulai dari tahun 1963. BASIC adalah singkatan dari *Beginner's All Purpose Symbolic Instruction Code.* Sesuai namanya, bahasa BASIC dibuat untuk tujuan memudahkan pengguna agar dapat dengan mudah mempelajari, membuat dan mengembangkan program komputer.

Visual basic 6.0 sangat populer dan masih banyak dipakai hingga saat ini. Sayangnya dukungan terhadap *Visual Basic 6* telah dihentikan oleh microsoft mulai bulan Maret 2008. Namun, program yang dibuat dengan Visual Basic 6 masih dapat dijalankan pada sistem operasi terbaru, seperti *windows server 2008* maupun *Windows Vista*. *Visual Basic .Net* diluncurkan Februari 2002, merupakan penerus dari *visual basic 6.0* dan menggunakan *platform .Net* yang berbeda dengan *Visual Basic* sebelumnya.

Visual Basic, merupakan bahasa dan aturan pemrograman yang harus ditaati dalam menuliskan perintah-perintah agar program dapat dikompilasi.

Visual.Basic 2008 adalah Aplikasi IDE (*Integrated Development Environment*) yang digunakan untuk mengembangkan *software*. Di dalam aplikasi IDE inilah tersedia berbagai fitur yang memudahkan pemrograman, seperti kompilasi *debugging*, pengaturan projek, mengedit antarmuka secara *visual*, dan lain-lain (Rachmad Hakim S; 2009: 2).

Visual Studio 2008 hadir dengan beberapa versi, yaitu:

1. ***Team System***, didesain untuk pemrograman dilingkungan korporasi dengan jumlah *proggramer* yang besar.

2. ***Professional Edition***, di desain untuk pemrograman yang melibatkan sedikit *proggrammer*.
3. ***Standard Edition***, di desain untuk pemrograman standar yang bukan *enterprise*.
4. ***Express Edition***, di desain untuk pemula yang baru belajar hobi dengan fasilitas yang sangat terbatas.

Pemrograman dengan *Visual Basic* dibuat dengan beberapa tahap berikut:

1. Menuliskan kode program dengan bantuan aplikasi IDE. Artinya, Anda menuliskan program menggunakan aplikasi *Microsoft Visual Basic 2008 Express Edition.*
2. Mengompilasi kode program tersebut menjadi program yang dapatdijalankan/dieksekusi. Hasil kompilasi adalah intruksi CIL (*Common Intermediate Language*) atau MSIL (*Microsof*t *Internediate Language*) yang hanya dimengerti oleh kompiler JIT*(Just In Time*) dan tidak dapat dieksekusi langsung padda komputer.
3. Penyebaran (distribusi) program di komputer dengan .Net Framework (Rachmad Hakim S; 2009: 3).

II.5.1. .NET Framework

.NET Framework, merupakan *library* dan *virtual machine* yang terus berkembang mengikuti teknologi terbaru. *Versi .NET Framework* dimulai dari versi 1.0, 1.1, 2.0, 3.0 dan 3.5. *Versi .NET* yang biasanya dirilis dengan perbaikanserta dukungan terhadap teknologi baru sehinggasemakin memudahkan pengembangan *software* (Rachmad Hakim S; 2009: 2).

.NET Framework, merupakan *software* kerangka kerja yang menghubungkan antara aplikasi .NET dengan sistem operasi, yang secara garis besar terdiri atas:

1. *Library*, berisi kode-kode siap pakai dan banyak dibutuhkan oleh programmer.
2. *Virtual machine*, berupa aplikasi yang digunakan untuk menjalankan program hasil kompilasi (Rachmad Hakim S; 2009: 5).

II.6. SQL Server 2005

SQL Server 2005 adalah Server basis data yang secara fungsional adalah proses menyediakan layanan basis data. *Client* berinteraksi dengan layanan basis data melalui antar muka komunikasi tertentu yang bertujuan untuk pengendalian dan keamanan. *Client* tidak mempunyai akses langsung ke data tetapi selalu berkomunikasi dengan server basis data (Stanek R. William; 2009: 4)

Dengan munggunakan *SQL Server* Setup, kita dapat membuat *instance-instance* baru dari *SQL Server*, menambahkan komponen-komponen, membangun kembali *registry SQL Server*, meng-uninstall *SQLServer 2005* akan digunakan dilingkungan Anda. Ketika Anda sudah memutuskan mengenairole atau peran yang akan dimiliki oleh *SQL Server*, Anda bisa membuat rencana peluncuran dan kemudian meluncurkan *SQL Server*.

SQL server dirancang sebagai sebuah *platform Business Intelligence* yang lengkap yang bisa digunakan untuk:

1. Ekstraksi, Transformasi, dan Loading *(Extraction, Transformation, and Loading- ETL)*
2. *Data warehous*e (gudang data) relasional
3. Database-database multi dimensi dan data mining (penambangan data)
4. *Analysis services* di SQL Server 2005
5. Pelaporan terkelola (*managed Reporting*) (Stanek R. William; 2009: 25)

II.7. UML (Unified Modeling Languages)

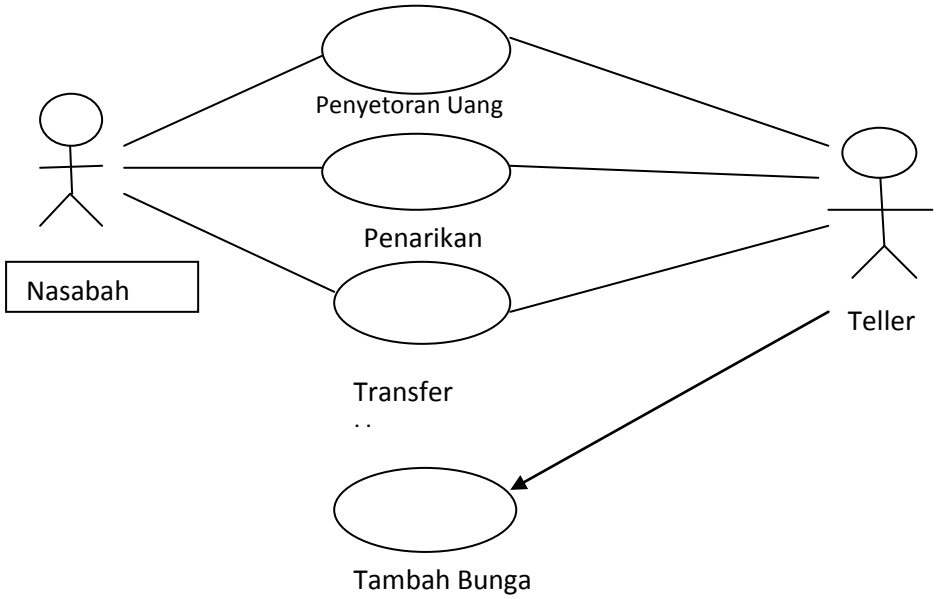
Unified Modeling Languages yang berarti bahasa pemodelan standar. (*Chonoles*; 2003: I) mengatakan sebagai bahasa, berarti UML memiliki sintaks dan sematik. Ketika kita membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti. UML bukan hanya sekedar diagram, tetapi juga menceritakan konteks (Prabowo Pudjo Widodo, Herlawati; 2011: 6)

Salah satu kontributor terhadap diagram use case dalam UML adalah Ivar Jacobsen. *Use Case* menggambarkan *External* view dari sistemyang akan kita buat modelnya *(Pooley*, 2003:15) mengatakan bahwa model *Use case* dapat dijabarkan dalam diagram *Use case*, tetapi yang perlu di ingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

Ada tiga(3)Komponen pembentukan digram *Use case* adalah:

1. Aktor (*actor*), menggambarkan pihak-pihak yang berperan dalam sistem.
2. *Use Case*, aktivitas/sarana yang disiapkan oleh bisnis/sistem.
3. Hubungan (*Link*), aktor mana saja yang terlibat dalam use case ini.

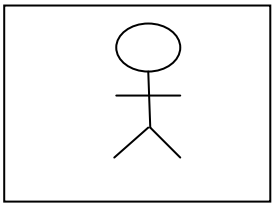
Gambar dibawah ini merupakan salah satu contoh bentuk diagram use case.



Gambar II.1. Diagram Use Case

Sumber: (Prabowo Pudjo Widodo, Herlawati; 2011)

II.7.1 Aktor



Gambar II.2. Simbol Aktor

Sumber: (Prabowo Pudjo Widodo, Herlawati; 2011)

Gambar 2.1 memperlihatkan diagram use case dengan dua aktor (nasabah dan teller) dan empat use case (Penyetoran uang, penarikan uang, transfer uang dan tambah bunga). Simbol aktor adalah gambar orang.

(*Chonoles*, 2003: bab 8) menyarankan sebelum membuat use case dan menentukan aktornya, agar mengidentifikasi siapa saja pihak yang terlibat dalam sistem kita. Pihak yang terlibat biasanya dinamakan *Stakeholder.*

Langkah awal yang baik adalah mempertimbangkan kebutuhan klien dan pelanggan sebelum menentukan use case. Setiap klien dan pelanggan sebelum membentuk use case. Setiap sistem memiliki *stakholde*r potensial yang harus dipertimbangkan karena berpengaruh terhadap kinerja bisnis/sistem tersebut.

Menurut (Whitten; 2004 :259) ada empat macam tipe aktor:

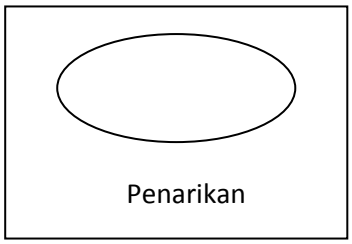
1. *Primary business actor* (Aktor bisnis utama) yaitu *stakeholder* yang terutama mendapatkan keuntungan dari pelaksanaan use case dengan menerima nilai yang terukur atau terobservasi.
2. *Primary sistem actor* (Aktor sistem utama) yaitu *stakeholder* yang secara langsung berhadapan dengan sistem untuk menginisiasi atau memicu kegiatan atau sistem.
3. *External server actor* (Aktor server eksternal) yaitu *stakeholder* yang melayani kebutuhan pengguna use case (misalnya biro kredit yang memiliki kuasa atau perubahan kartu kredit).
4. *External receiving actor* (Aktor penerima eksternal) yaitu stakeholder yang bukan pelaku utama, tapi menerima nilai yang terukur atau teramati *(output)* dari *use case.*

II.7.2 Use Case

Meneurut (Pilone; 2005: 7) *Use case* menggambarkan fungsi tertentu dalam suatu sistem berupa komponen,kejadian atau kelas. Sedangkan (Whitten;

2004: 258) mengartikan use casesebagai urutan langkah-langkah yang secara tindakan saling terkait(skenario), baik terotamisasi maupun secara manual, untuk tujuan melengkapi satu tugas bisnis tunggal.

Use Case digambarkan dalam bentuk Elips/Oval.



Gambar II.3. Simbol Use Case

Sumber: (Prabowo Pudjo Widodo, Herlawati; 2011)

Use case sangat menentukan karakteristik sistem yang kita buat, oleh karena itu (Chonoles; 2003: 8) menawarkan cara untuk menghasilkan use case yang baik, yaitu :

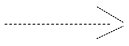
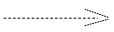
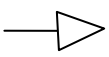
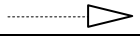
1. Pilihlah nama yang baik. *Use case* adalah sebuah behaviour (Perilaku), jadi seharusnya dalam frase kata kerja.
2. Ilustrasikan perilaku dengan lengkap. *Use case* dimulai dari inisiasi oleh aktor primerdan berakhir pada aktor dan menghasilkan tujuan.
3. Identifikasi perilaku dengan lengkap. Untuk mencapai tujuan dan menghasilkan tujuan danmenghasilkan nilai tertentu dari aktor, *Use case* harus lengkap.
4. Menyediakan *Use case* lawan (*inverse).* Kita biasanya membutuhkan use case yang membatalkan tujuan, misalnya pada use case pemesanan kamar, dibutuhkan pula *Use case* pembatalan pesanan kamar.

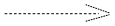
5. Batasi *Use case* hingga satu perilaku saja.
6. Nyatakan *Use case* dari sudut pandang aktor. Tulislah *Use case* dari sudut pandang aktor bukan dari sistem.

Menurut (Adi Nugroho; 2010 : 6) *Unified Modelling Language* (UML) adalah 'bahasa' pemodelan untuk sistem atau perangkat lunak yang berparadigma 'berorientasi objek'. Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

Relasi-relasi antar pengklasifikasi (*classifier*) yang dikenali UML (*Unified Modelling Language*) adalah asosiasi (*association*), generalisasi (generalization), aliran (*flow*), dan berbagai jenis kebergantungan (*dependency*), termasuk didalamnya realisasi (*realization*) dan penggunaan (*usage*).

Tabel II.3. Relasi-relasi dalam UML

Relasi	Fungsi	Notasi
Asosiasi (*association*)	Mendeskripsikan hubungan antar-*instance* suatu kelas.	
Kebergantungan (*dependency*)	Relasi antardua elemen model.	
Aliran (*flow*)	Relasi antardua versi suatu objek.	
Generalisasi (*generalization*)	Relasi antara pengklasifikasi yang memiliki deskripsi yang bersifat lebih umum dengan berbagai pengklasifikasi yang lebih spesifik, digunakan dalam struktur pewarisan.	
Realisasi	Relasi antara spesifikasi dan	

(*realization*)	implementasinya.	
Penggunaan (*usage*)	Situasi dimana salah satu elemen membutuhkan elemen yang lainnya agar dapat berfungsi dengan baik.	

Sumber : (Adi Nugroho; 2010 : 23)

Untuk membuat suatu model, UML memiliki diagram grafis yaitu sebagai berikut. (Sri D. dan Romi SW, 2003)

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.

2. *Class Diagram*

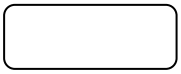
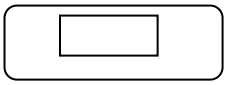
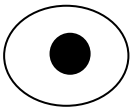
   Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class* memiliki tiga area pokok, yaitu: Nama (dan stereotype), Atribut, serta Metoda.

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*).

Tabel II.4. Jenis-Jenis *State*

Relasi	Fungsi	Notasi
State sederhana	*State* tanpa struktur apapun di dalamnya.	
State komposit konkuren	*State* yang dibagi menjadi 2 atau lebih *substate* konkuren.	
Initial state	*State* mengindikasikan awal rangkaian *state* dalam diagram *state*.	
Final state	*State* mengindikasikan akhir rangkaian *state* dalam diagram *state*.	

Sumber : (Adi Nugroho 2010 : 52)

4. *Activity Diagram*

Activity diagram 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.

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).

6. *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*.

7. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code.* Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil.

8. *Deployment Diagram*

Deployment 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 (Adi Nugroho; 2010 : 10)