II.2. Citra Digital

II.2.1. Defenisi Citra

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari sebuah objek. Citra sebagai keluaran suatu sistem perekamam data dapat bersifat optik berupa foto, bersifat *analog* berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpan. (T.Sutoyo, dkk ; 2009 : 9).

II.2.1.1. Defenisi Citra *Analog*

Citra *analog* adalah citra yang bersifat kontinu, seperti gambar pada minitor televisi, foto sinar x, foto yang tecetak dikertas foto, lukisan, pemandangan alam, hasil *CT scan*, gambar-gambar yang terekam pada pita kaset, dan lain sebagainya. Citra *analog* tidak dapat direpresentasikan dalam komputer sehingga tidak bisa diproses dikomputer secara langsung. Oleh sebab itu, agar citra dapat diproses dikomputer, proses konversi *analog* ke digital harus dilakukan terlebih dahulu. (T.Sutoyo, dkk ; 2009 : 9).

II.2.1.2. Defenisi Citra Digital

Citra digital adalah citra yang dapat diolah oleh komputer. Yang menghasilkan keluaran sebuah sistem perekaman data. (T. Sutoyo,dkk; 2009: 9). Sebuah citra digital dapat mewakili oleh sebuah matriks yang terdiri dari M kolom N baris, dimana perpotongan antara kolom dan baris disebut *pixel* (*pixel* = picture element), yaitu elemen terkecil dari sebuah citra. pixel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y)

adalah f(x,y), yaitu besar intensitas atau warna dari *pixel* di titik itu.Oleh sebab itu, sebuah citra digital dapat ditulis dalam bentuk matriks berikut. (T. Sutoyo, dkk; 2009 : 20).

$$f(x,y) = \begin{vmatrix} f(0,0) & f(0,1) & \dots & f(0,M\text{-}1) \\ f(1,0) & \dots & \dots & f(1,M\text{-}1) \\ \dots & \dots & \dots & \dots \\ f(N\text{-}1,0) & f(N\text{-}1,1) & \dots & f(N\text{-}1,M\text{-}1) \end{vmatrix}$$

Berdasarkan matriks tersebut, secara matematis citra digital dapat dituliskan sebagai fungsi intensitas f (x,y), dimana harga x (baris) dan y (kolom) merupakan koordinat posisi dan f(x,y) adalah nilai fungsi pada setiap titik (x,y) yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari *pixel* di titik tersebut. Pada proses digitalisasi (sampling dan kuantitas) diperoleh besar baris M dan kolom N hingga citra membentuk matriks ***M x N*** dan jumlah tingkat keabuan *pixel* G (T. Sutoyo, dkk; 2009 : 20).

Pengolahan citra digital adalah sebuah disiplin ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar (peningkatan kontras, transformasi warna, restorasi citra), transformasi gambar (rotasi, translasi, skala, transformasi geometrik), melakukan pemilihan citra ciri (*feature images)* yang optimal untuk tujuan analisis, melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung pada citra, melakukan kompresi atau reduksi data untuk tujuan penyimpanan data, transmisi data, dan waktu proses data. Input dari pengoalahan citra adalah citra, sedangkan outputnya adalah citra hasil pengolahan (T. Sutoyo, dkk; 2009: 5).

II.2.2. Format *File* Citra

Ada dua jenis format *file* citra yang sering digunakan dalam pengolahan citra, yaitu citra *bitmap* dan citra vektor. (T.Sutoyo, dkk ; 2009 : 25).

II.2.2.1. Format *File* Citra *Bitmap*

Citra *bitmap* sering disebut juga dengan citra raster. Citra *bitmap* menyimpan data kode citra secara digital dan lengkap (cara menyimpannya adalah per *pixel*). Citra *bitmap* dipresentasikan dalam bentuk matriks atau dipetakan dengan menggunakan bilangan biner atau sistem bilangan lain. Citra ini memiliki kelebihan untuk memanipulasi warna, tetapi untuk mengubah objek lebih sulit. Tampilan *bitmap* mampu menunjukkan kehalusan gradiasi bayangan dan warna dari sebuah gambar. Oleh karena itu, *bitmap* merupakan media elektronik yang paling tepat untuk gambar-gambar dengan perpaduan gradiasi warna yang rumit, seperti foto dan lukisan digital. Citra *bitmap* biasanya diperoleh dengan cara *scanner, camera* digital, *video, capture,* dan lain-lain. (T.Sutoyo, dkk ; 2009 : 25).

II.2.2.2. Format *File* Citra Vektor

Citra vektor dihasilkan dari perhitungan matematis dan tidak berdasarkan *pixel*, yaitu data tersimpan dalam bentuk vektor posisi, dimana yang tersimpan hanya informasi vektor posisi dengan bentuk sebuah fungsi. Pada citra vektor, mengubah warna lebih sulit dilakukan, tetapi membentuk objek dengan cara mengubah nilai lebih mudah. Oleh karena itu, bila citra diperbesar atau diperkecil, kualitas citra relatif tetap baik dan tidak berubah. Citra vektor biasanya dibuat

menggunakan aplikasi-aplikasi citra vektor, seperti *corelDraw, adobe Illustrator, macromedia Freehand, Autocad,* dan lain-lain. (T. Sutoyo, dkk; 2009 : 27).

II.2.3. Jenis-jenis Citra Digital

Beberapa jenis citra digital yang sering digunakan adalah citra biner, citra *grayscale*, dan citra warna. (T. Sutoyo, dkk; 2009 : 21).

II.2.3.1. Citra Biner (Monokrom)

Banyaknya warna 2, yaitu hitam dan putih. Dibutuhkan 1 bit di memori untuk menyimpan kedua warna ini. (T. Sutoyo, dkk; 2009 : 21).

bit 0 = warna hitam

bit 1 = warna putih

II.2.3.2. Citra *grayscale* (skala keabuan)

Banyaknya warna tergantung pada jumlah bit yang disediakan di memori untuk menampung kebutuhan warna ini. (T. Sutoyo, dkk; 2009 : 21).

Citra 2 bit mewakili 4 warna dengan gradiasi warna berikut :

Citra 3 bit mewakili 8 warna dengan gradiasi warna berikut :

Semakin besar jumlah bit warna yang disediakan di memori, semakin halus gradiasi warna yang terbentuk.

II.2.3.3. Citra Warna (*True color*)

Setiap *pixel* pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar (*RGB = Red Green Blue*). Setiap warna dasar menggunakan penyimpanan 8 bit = 1 byte, yang berarti setiap warna mempunyai gradiasi sebanyak 255 warna. Berarti setiap *pixel* mempunyai kombinasi warna sebanyak $2^8.2^8.2^8 = 2^{24}$ = 16 juta warna lebih. Itulah sebabnya format ini dinamakan *true color* karena mempunyai jumlah warna yang cukup besar sehingga bisa dikatakan hampir mencakup semua warna di alam. (T. Sutoyo, dkk; 2009 : 22).

Penyimpanan citra *true color* didalam memori berbeda dengan citra *grayscale*. Setiap *pixel* dari citra *grayscale* 256 gradiasi warna diwakili oleh 1 byte. Sedangkan 1 *pixel* citra *true color* diwakili oleh 3 byte, dimana masing-masing byte merepresentasikan warna merah (Red), hijau (Green), dan biru (Blue). Seperti ditunjukkan pada gambar II.1. dibawah ini :

Citra warna

Penyimpanan dalam memori

R=50 G=65 B=50	R=40 G=45 B=60	R=50 G=95 B=90	R=80 G=85 B=50	R=70 G=75 B=50
R=80 G=40 B=30	R=50 G=35 B=56	R=50 G=90 B=30	R=20 G=65 B=20	R=50 G=65 B=70
R=80 G=45 B=50	R=20 G=65 B=70	R=50 G=95 B=50	R=50 G=35 B=90	R=70 G=55 B=50
R=60 G=55 B=50	R=30 G=60 B=70	R=30 G=45 B=70	R=50 G=60 B=70	R=40 G=95 B=50
R=40 G=45 B=70	R=90 G=55 B=40	R=30 G=65 B=70	R=50 G=55 B=30	R=50 G=65 B=80

Gambar II.1. Contoh penyimpanan citra warna dalam

Sumber : Andi, 2009, 23.

II.2.4. Elemen-Elemen Citra Digital

Berikut adalah elemen-elemen yang terdapat pada citra digital :

1. Kecerahan (*Brightness*)

   Kecerahan (*Brightness*) merupakan intensitas cahaya yang dipancarkan *pixel* dari citra yang dapat ditangkap oleh sistem penglihatan. Kecerahan pada sebuah titik (*pixel*) di dalam citra merupakan intensitas rata-rata dari suatu area yang melingkupi.

2. Kontras (*Contrast*)

   Kontras (*Contrast*) menyatakan sebaran terang dan gelap dalam sebuah citra. Pada citra yang baik, komposisi gelap dan terang secara merata.

3. Kontur (*Contour*)

   Kontur (*Contour*) adalah keadaan yang ditimbulkan oleh perubahan intensitas pada *pixel-pixel* yang bertetangga. Karena adanya perubahan intensitas inilah mata mampu mendeteksi tepi-tepi objek di dalam citra.

4. Warna

   Warna sebagai persepsi yang ditangkap sistem visual terhadap panjang gelombang cahaya yang dipantulkan oleh objek.

5. Bentuk (*Shape*)

Bentuk (*Shape*) adalah properti intrinsik dari objek 3 dimensi, dengan pengertian bahwa bentuk merupakan properti intrinsik utama untuk sistem visual manusia (Sutoyo T, Mulyanto, dkk; 2009 : 24).

6. Tekstur (*Texture*)

Tekstur (*Texture*) dicirikan sebagai distribusi spasial dari derajat keabuan di dalam sekumpulan *pixel-pixel* yang bertetangga. Tekstur adalah sifat-sifat atau karakteristik yang dimiliki oleh suatu daerah tersebut. Tekstur adalah keteraturan pola-pola tertentu yang terbentuk dari susunan *pixel-pixel* dalam citra digital. (T. Sutoyo, dkk; 2009: 24).

II.3. Pemrograman Visual Basic

II.3.1. *Visual basic 2008*

Visual basic merupakan salah satu bahasa pemrograman yang andal dan banyak digunakan oleh pengembang untuk membangun berbagai macam aplikasi windows. *Visual basic 2008* atau visual basic 9 merupakan aplikasi pemrograman yang menggunakan teknologi .NET Framework. Teknologi .NET Framework merupakan komponen windows yang terintegrasi serta mendukung pembuatan, penggunaan aplikasi, dan halaman web. Teknologi .NET Framework mempunyai 2 komponen utama, yaitu CLR (Common Language Runtime) dan class library. CLR digunakan untuk menjalankan aplikasi yang berbasis .NET, sedangkan

Library adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi. (Wahana Komputer; 2010: 2).

II.3.2. Sistem *Visual basic 2008*

Sebelum menginstal *visual basic 2008*, komputer harus memenuhi beberapa persyaratan agar *visual basic 2008* dapat dijalankan dengan baik. Adapun, persyaratan (system requirments) yang harus dipenuhi dapat dilihat pada tabel II.1. berikut :

Tabel II.1. Sistem *Requirements Visual Basic 2008*

Sistem	Syarat minimal	Syarat yang direkomendasikan
Arsitektur	x86 dan x64	
Sistem Operasi	Microsoft Windows XP Service pack 2 Microsoft Windows Server 2003 Windows Vista	
Prosesor	CPU 1.6 GHz (Giga Hertz)	Windows XP dan Windows Server 2003: CPU 2,2 GHz atau yang lebih tinggi. Windows Vista: CPU 2,4 GHz.
RAM	Windows XP dan Windows Server 2003 384 MB (Mega Byte) Windows Vista: 768 MB.	RAM 1024 MB/1GB atau yang lebih besar.
Harddisk	Tanpa MSDN R Ruang kosong harddisk pada drive penginstalan 2 GB. Sisa ruang harddisk kosong 1 GB.	Kecepatan harddisk 7200 RPM atau yang lebih tinggi.
	Tanpa MSDN R Ruang kosong harddisk pada drive penginstalan 3,8 GB (MSDN diinstal full). 2,8 GB untuk menginstal MSDN default. Kecepatan harddisk 5400 RPM.	
Display Layar	1024x768 display	1280x1024 display

Sumber : Andi, 2010, 2.

II.3.3. Mengenal Area Kerja *Visual basic 2008*

Setelah berhasil menginstal *visual studio 2008* yang didalamnya terdapat *visual basic 2008*, maka selanjutnya adalah mencoba menjalankan dan mengenal lingkungan kerja *visual basic 2008*. Lingkungan kerja *visual basic* atau disebut *Integrated Development Environment* (*IDE*) adalah suatu lingkungan kerja tempat programmer melakukan pemrograman yang didukung oleh *compiler*, *editor* baik *editor grafis* maupun kode, dan lain sebagainya untuk memudahkan pemrograman. (Wahana Komputer; 2010: 3).

II.3.3.1. Membuka *IDE Visual basic 2008*

Langkah membuka lingkungan kerja *visual basic 2008* sebagai berikut :

1. Ikuti salah satu cara berikut untuk membuka *IDE* visual basic :

    - Klik ganda shortcut *visual basic 2008* pada desktop.

Gambar II.2. Ikon *shortcut* pada desktop
Sumber : Andi, 2010, 4.

    - Melalui menu *start* > *Microsoft Visual Studio 2008* > *Microsoft Visual Studio 2008.*

Gambar II.3. Membuka *visual studio 2008* melalui menu
Sumber : Andi, 2010, 4.

2. Pada saat pertama kali menjalankan *visual basic 2008*, akan muncul kotak dinformasi proses setting *IDE visual basic 2008*. Setelah proses setting selesai, kotak *splash screen* sesuai dengan gambar II.4. akan muncul :

Gambar II.4. *Splash screen* visual studio 2008.
Sumber : Andi, 2010, 4.

3. Setelah itu, akan terbuka lingkungan kerja atau *IDE* dari *visual basic 2008* yang ditunjukkan pada gambar II.5. berikut :

Gambar II.5. Area kerja *visual basic 2008* saat pertama kali dijalankan
Sumber : Andi, 2010, 5.

II.4. UML (*Unified Modelling Language*)

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia perkembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi perkembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (sharing) dan mengkomunikasikan rancangan dengan baik. (Munawar; 2005: 17).

UML merupakan kesatuan bahasa pemodelan yang dikembangkan oleh *Booch, Object Modeling Technique* (OMT) dan *Object Oriented Engineering* (OOSE). Metode *Booch* dari *Grady Booch* sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan desain ke dalam empat tahapan interatif, yaitu: identifikasi kelas-kelas dan objek-objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode *Booch* adalah pada detail dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh *Rumbaugh* didasarkan pada analisis terstuktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, desain sistem, desain objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari *Jacobson* lebih memberi penekanan dan *use case*. OOSE memiliki tiga tahapan yaitu membuat model *requirement* dan analisis, desain dan implementasi dan model pengujian (test Model). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi sederhana namun

mencakup seluruh tahapan dalam rekayasa perangkat lunak. (Munawar; 2005: 17).

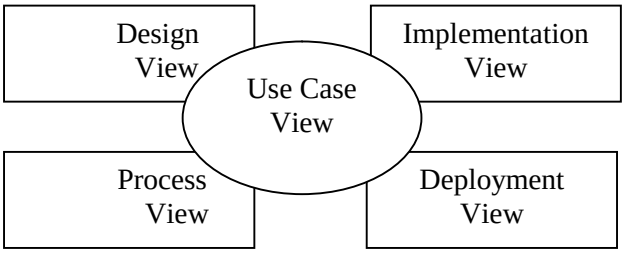
Dengan UML, metode *Booch*, OMT dan OOSE digabungkan dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam dari pada metode lainnya. Unsur-unsur yang membentuk UML ditunjukkan dalam Gambar II.6 : (Munawar; 2005: 18).

Gambar II.6 Unsur-unsur yang membentuk UML

Sumber : Graha Ilmu, 2005, 18

UML adalah hasil kerja dari konsorsium berbagai organisasi yang berhasil dijadikan sebagai standar baku dalam OOAD (*Object Oriented Analysis* dan *Design*). UML tidak hanya domain dalam penotasian dilingkungan OO tetapi juga populer di luar lingkungan OO. Ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan bahasa pemrograman. Sebagai sebuah sketsa UML bisa berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bisa diketahui informasi detail tentang *coding* program (*Forward engineering*) atau bahkan membaca program dan mengimplementasikannya kembali ke dalam diagram (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana kode program yang tidak terdokumentasi asli hilang atau bahkan belum perna dibuat sama sekali. Sebagai bahasa pemrograman, UML dapat diterjemahkan diagram yang ada di UML menjadi kode program siap untuk dijalankan. (Munawar; 2005: 18).

UML dibangun atas model 4+1 *view*. Model ini didasarkan pada fakta bahhwa struktur sebuah sistem dideskripsikan dalam *view* dimana salah satu diantaranya *use case view*. *use case view* ini memegang peran khusus untuk mengintegrasikan *content* ke *view* yang lain. Model 4+1 *view* ditunjukkan pada gambar II.7 : (Munawar; 2005: 20).



Gambar II.7. Model 4+1 *View*

Sumber : Graha Ilmu, 2005, 20

Kelima *view* tersebut tidak berhubungan dengan diagram yang dideskripsikan di UML. Setiap *view* berhubungan dengan perspektif tertentu dimana sistem akan diuji. *View* yang berbeda akan menekankan pada aspek yang berbeda dari sistem yang mewakili tentang sistem bisa dibentuk dengan menggabungkan informasi-informasi yang ada pada kelima *view* tersebut. (Munawar; 2005: 20).

Use case view mendefinisikan perilaku eksternal sistem. Hal ini menjadi daya tarik bagi *end user*, analis dan tester. Pandangan ini mendefenisikan kebutuhan sistem karena mengandung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari peran dan sering dikatakan yang mendrive proses perkembangan perangkat lunak. (Munawar; 2005: 20).

Design view mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di *use case*. *Design view* ini berisi definisi komponen program, *class-class* utama bersama-sama dengan spesifikasi data, perilaku dan interaksinya. Informasi yang terkandung di *view* menjadi pergantian para progremer karena menjelaskan secara detil bagaimana fungsionalitas sistem akan diimplementasikan. (Munawar; 2005: 20).

Implementasi *view* menjelaskan komponen-komponen visi yang akan dibangun. Hal ini berbeda dengan komponen logic yang dideskripsikan pada *design view*. Termasuk disini diantaranya *file exe*, *library* dan *database*. Informasi yang ada di *view* dan integrasi sistem. (Munawar; 2005: 21).

Proses *view* berhubungan dengan hal-hal yang berkaitan dengan *concurrency do* dalam sistem. Sedangkan *deployment view* menjelaskan

bagaimana komponen-komponen fisik didistribusikan ke lingkungan fisik seperti jaringan komputer dimana sistem akan dijalankan. Kedua *view* ini menunjukan kebutuhan non fungsional dari sistem seperti toleransi kesalahan dan hal-hal yang berhubungan dengan kinerja. (Munawar; 2005: 21).

II.4.1. *Use Case* Diagram

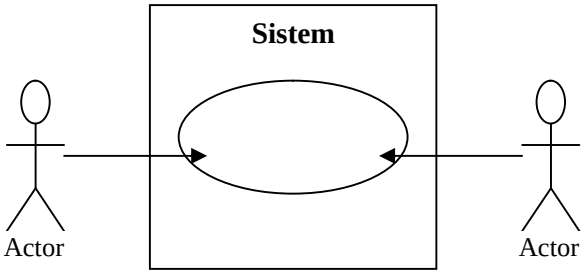
Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara deskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem yang disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras dan urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh pengguna tujuan umum pengguna. (Munawar; 2005: 63).

Dalam pembicaraan tentang *use case*, pengguna biasanya disebut dengan *actor*. *Actor* adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem. (Munawar; 2005: 63).

Model *use case* adalah bagai dari model *requirement*. Termasuk disini adalah problem domain object dan penjelasan tentang *user interface*. *Use case* memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari *persfectif user*. (Munawar; 2005: 63).

Notasi *use case* menunjukkan 3 aspek dari sistem yaitu *actor use case* dan *system / sub system boundary*. *Actor* mewakili peran orang, *system* yang lain atau

alat ketika berkomunikasi dengan *use case*. Ilustrasi *actor, use case* dan *system* ditunjukkan pada gambar II.8 : (Munawar; 2005: 64).

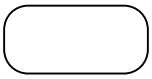
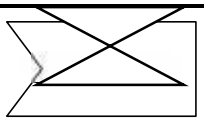


Gambar II.8. *Use Case* Diagram
Sumber : Graha Ilmu, 2005, 64

II.4.2. *Activity* Diagram

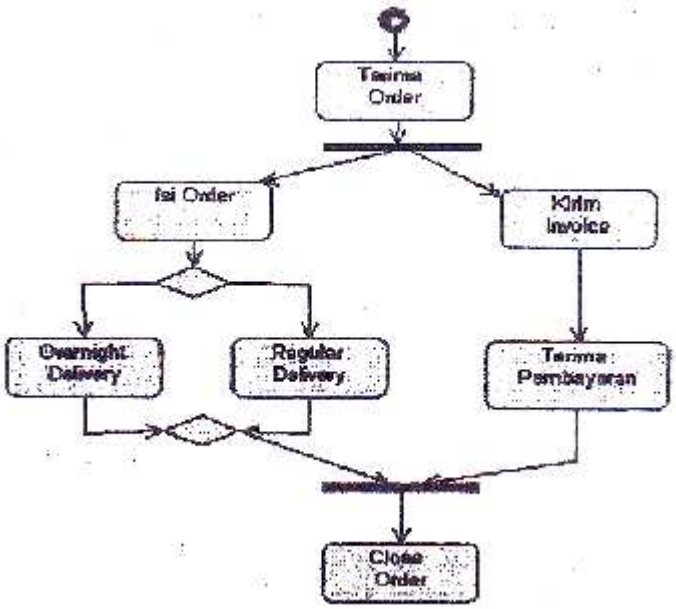
Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity* diagram mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity* diagram bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa. Berikut adalah simbol-simbol yang sering digunakan pada saat pembuatan *activity* diagram. Seperti yang ditunjukkan pada tabel II.5 berikut: (Munawar; 2005: 109).

Tabel II.2. Simbol-Simbol yang sering dipakai Pada *Activity* Diagram

Simbol	Keterangan
	Titik Awal
	Titik Akhir
	Activity
	Pilihan untuk mengambil keputusan
	Foks: Untuk menunjukkan kegiatan yang dilakukan secara paralel
	Rake: Menunjukkan adanya dekomposisi
	Tanda Waktu
	Tanda Penerimaan
	Aliran Akhir (Flow Final)

Sumber : Graha Ilmu, 2005, 109

Contoh *activity diagram* sederhana ditunjukkan pada gambar II.9. (Munawar; 2005: 111).



Gambar II.9. Contoh *Activity Diagram* Sederhana

Sumber : Graha Ilmu, 2005, 111

II.4.3. *Squence* Diagram

Squence diagram digunakan untuk menggambarkan perilaku pada sebuah sekenario. Diagram ini menunjukkan sebuah contoh objek dan pesan yang diletakkan diantara objek-objek ini didalam *use case*. (Munawar; 2005: 87).

Komponen utama *Squence* diagram terdiri dari atas objek yang dituliskan dengan kotak segiempat bernama. *Messege* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical*. (Munawar; 2005: 87).

1. Objek / *participant*

Objek diletakkan di dekat bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram. Setiap *participant* dihubungkan garis titik-titik yang disebut *lifeline.* Sepanjang *lifeline* ada kotak yang disebut *activation*. *Activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation*. *Activation* mewakili sebuah eksekusi operasi dari participant. Panjang kotak ini berbanding lurus dengan durasi *activation.* Bentuk *participant* dapat dilihat pada gambar II.10 : (Munawar; 2005: 87).

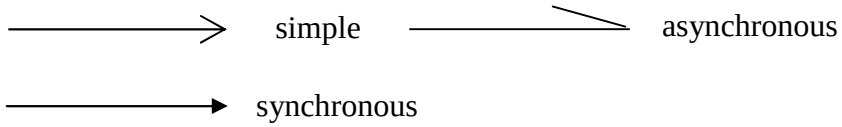
Gambar II.10 Bentuk *Participant*
Sumber : Graha Ilmu, 2005, 87

2. *Messege*

Sebuah *messege* bergerak dari suatu *participant* ke *participant* yang lain dan dari *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri. (Munawar; 2005: 88).

Sebuah *message* bisa jadi *simple*, *synchronous* atau *asynchoronous*. *Message* yang *simple* adalah sebuah perpindahan (transfer), contoh dari satu *participant* ke *participant* yang lainnya. Jika suatu *participant* mengirimkan sebuah *message* tersebut akan ditunggu sebelum di proses dengan urusannya. Namun jika *message asynchoronous* yang dikirimkan, maka jawabannya atas *message* tersebut tidak

perlu ditunggu. Simbol *message* pada *squnence* diagram dapat dilihat pada gambar II.14 : (Munawar; 2005: 87).



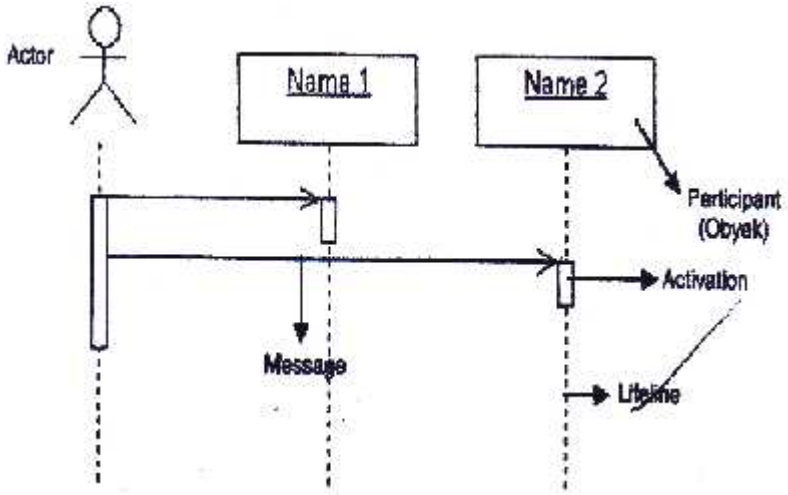
Gambar II.11 Bentuk *Messege*

Sumber : Graha Ilmu, 2005, 88

3. *Time*

Time adalah diagram yang mewakili waktu pada arah vertikal. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijalankan terlebih dahulu dibanding *message* yang lebih dekat kebawah. (Munawar; 2005: 88).

Terdapat dua dimensi pada *squence* diagram yaitu dimensi dari kiri ke kanan menunjukkan tata letak participant dan dimensi dari atas ke bawah menunjukkan lintasan waktu. Simbol-simbol yang ada pada *squence* diagram ditunjukkan pada gambar II.15 : (Munawar; 2005: 89).



Gambar II.12. Bentuk *Time*

Sumber : Graha Ilmu, 2005, 88