# BAB 2
# LANDASAN TOERI

## 2.1 Pengertian Citra Digital

Citra diartikan secara harfiah adalah gambar pada bidang dwimatra (dua dimensi) yang disusun oleh banyak piksel yaitu bagian terkecil dari citra [2]. Dalam komputer, setiap piksel diwakili oleh dua buah bilangan bulat (*integer*) untuk menunjukkan lokasinya dalam bidang citra dan sebuah nilai dalam bilangan bulat (*integer*) untuk menunjukkan cahaya atau keadaan terang-gelap piksel tersebut.

Suatu piksel dapat ditunjukkan lokasinya dengan menggunakan koordinat (0,0) yang digunakan untuk posisi kiri atas dalam bidang citra. Tingkat pencahayaan suatu piksel dapat ditunjukkan dengan bilangan bulat yang besarnya 8 bit, dengan lebar selang nilai 0-255, di mana 0 untuk warna hitam, 255 untuk warna putih [1]. Citra digital yang diperoleh dari kamera digital akan menghasilkan intensitas pantulan yang menggambarkan terang dan gelap pada penampilan piksel-piksel penyusunannya selain itu juga akan memberikan informasi warna.

## 2.2 Pengolahan Citra

Pengolahan citra digital adalah pemrosesan citra, khususnya dengan menggunakan komputer sehingga citra itu kualitasnya menjadi lebih baik dan menghasilkan informasi nilai untuk tiap-tiap warnanya [1]. Tujuan utama pengolahan citra adalah agar citra yang mengalami gangguan mudah diinterpretasi oleh manusia maupun mesin (komputer).

Teknik pengolahan citra digital adalah mentransformasikan citra dua dimensi menjadi citra lain dengan kualitas yang lebih baik menggunakan komputer [2]. Proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra. Jadi masukannya berupa citra dan keluarannya juga berbentuk citra, dengan kualitas yang lebih baik dari citra masukan.

Beberapa contoh operasi pengolahan citra adalah pengubahan kontras citra, penghilangan derau (*noise*) dengan operasi penapisan (*filtering*), penajaman (sharpening), pemberian warna semu (pseudo*color*ing), dan sebagainya.

## 2.3 *Computer Vision*

Pengolahan citra adalah pengolahan suatu citra dengan menggunakan komputer, untuk menghasilkan suatu citra yang lain. Hubungan pengolahan citra dengan *computer vision* yaitu pengolahan citra merupakan proses awal (*preprocessing*) pada *computer Vision* [2]. *Computer vision* mencoba meniru cara kerja visual manusia (*Human Vision*). *Human Vision* sesungguhnya sangat kompleks, manusia melihat objek dengan indera penglihatan (mata) lalu objek citra diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek apa yang tampak dalam pandangan matanya. Hasil interpretasi ini mungkin digunakan untuk mengambil suatu keputusan.

*Computer vision* merupakan proses otomatis yang mengintegrasikan sejumlah besar proses untuk persepsi visual, seperti akuisisi citra, pengolahn citra, klasifikasi, pengenalan (*recognition*), dan membuat keputusan. *Computer vision* terdiri dari teknik-teknik untuk mengestimasi ciri-ciri objek dalam citra, pengukuran ciri yang berkaitan dengan geometri objek, dan menginterpretasi informasi geometri tersebut.

Proses-proses dalam *computer vision* dibagi dalam 3 (tiga) aktifitas :

1. Memperoleh atau mengakuisisi citra digital.
2. Melakukan teknik komputasi untuk memproses atau memodifikasi data citra. (operasi-operasi pengolahan citra).
3. Menganalisis dan menginterpretasi Citra menggunakan hasil pemrosesan untuk tujuan tertentu, misalnya memandu robot, mengontrol peralatan, memantau manufaktur dan lain-lain.

Tujuan dari *computer vision* adalah untuk mengekstrak informasi yang berguna dari gambar [9].

## 2.4 Konsep Citra *Gray* dan Warna

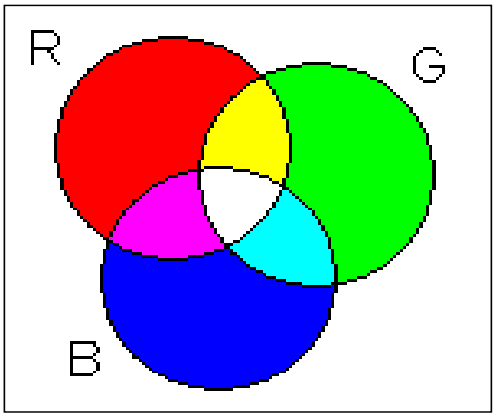
Warna adalah kesan yang diperoleh mata dari cahaya yang dipantulkan oleh benda-benda yang dikenainya. Warna yang diterima oleh mata dari sebuah objek ditentukan oleh warna sinar yang dipantulkan oleh objek tersebut. Warna-warna yang dapat ditangkap oleh mata manusia merupakan kombinasi cahaya dengan panjang berbeda.

Citra *Gray* merupakan warna-warna piksel yang berada pada rentang warna hitam dan putih. Jumlah warna pada citra *gray* adalah 256, karena citra *gray* jumlah bitnya adalah 8, sehingga jumlah warnanya adalah 28 = 256, nilanya berada pada jangkauan 0-255 [10] . Sehingga nilai intensitas dari citra *gray* tidak akan melebihi 255 dan tidak memungkinkan kurang dari 0.



**Gambar 2.1 Contoh Representasi *Grayscale***

Sedangkan jumlah warna untuk citra RGB (*Red*, *Green Blue*) adalah dengan mengalikan jumlah pada masing-masing komponennya, jumlah dari tiap komponennya, R=255 (8 bit) G=255(8 bit) dan B=255(8 bit) sehingga sering kali disebut citra dengan intensitas 24 bit.




**Gambar 2.2 Contoh Representasi RGB**

## 2.5 Pengolahan Warna RGB

Pengolahan warna menggunakan model RGB sangat mudah dan sederhana, karena informasi warna dalam komputer sudah dikemas dalam model yang sama [1]. Hal yang perlu dilakukan adalah melakukan pembacaan nilai-nilai R,G, dan B pada suatu piksel.

Salah satu cara untuk menghitung warna yaitu dengan cara normalisasi ketiga komponen tersebut. Normalisasi penting dilakukan terutama bila sejumlah citra ditangkap dengan penerangan yang berbeda-beda. Hasil perhitungan tiap komponen warna pokok yang telah dinormalisasi akan menghilangkan pengaruh penerangan sehingga setiap komponen warna dapat dibandingkan satu sama lainnya walaupun berasal dari citra dengan kondisi penerangan yang tidak sama. Cara melakukan normalisasi adalah sebagai berikut [1] :

$$r = \frac{R}{R+G+B} \tag{2.1}$$

$$g = \frac{G}{R+G+B} \tag{2.2}$$

$$b = \frac{B}{R+G+B} \tag{2.3}$$

Bila ketiga komponen warna yang telah dinormalisasi dengan masing-masing indeks warna merah (r), indeks warna hijau (g), dan indeks berwarna biru

(b), mempunyai nilai yang sama (1/3), maka objek tidak berwarna. Bila r lebih besar dari pada g dan b, maka objek berwarna merah, dan seterusnya.

## 2.6 Format RGB (*Red, Green, Blue*)

Format RGB adalah format dasar yang digunakan oleh banyak peralatan elektronik seperti monitor, LCD atau TV untuk menampilkan sebuah gambar [11]. Pada format RGB, suatu warna didefinisikan sebagai kombinasi (campuran) dari komponen *Red*, *Green*, dan *Blue*. Nilai-nilai yang terdapat pada format RGB 24 bit yang dijelaskan pada Tabel 2.1

**Tabel 2.1 Format RGB**

| Warna | R | G | B |
|---|---|---|---|
| Hitam | 0 | 0 | 0 |
| Merah | 255 | 0 | 0 |
| Hijau | 0 | 255 | 0 |
| Biru | 0 | 0 | 255 |
| Kuning | 255 | 255 | 0 |
| Magenta | 255 | 0 | 255 |
| Cyan | 0 | 255 | 255 |
| Putih | 255 | 255 | 255 |
| Abu-Abu | 127 | 127 | 127 |
| Orange | 255 | 110 | 0 |
| Ungu | 128 | 0 | 255 |
| Coklat | 128 | 25 | 0 |
| Pink | 255 | 190 | 220 |
| Navy | 0 | 0 | 120 |

## 2.7 *Color filtering*

*Color filtering* adalah satu teknik pengolahan citra yang berguna untuk menemukan sebuah warna yang terdapat pada sebuah gambar [12]. Dengan

ditemukannya warna yang di cari pada sebuah gambar, maka dapat menentukan proses apa yang selanjutnya harus dilakukan. Algoritma ini banyak digunakan dalam penelitian. Salah satunya pada *AForge.NET Framework* [13]. Pada dasarnya pencarian ini menggunakan kombinasi dari komponen *Red*, *Green*, dan *Blue*.

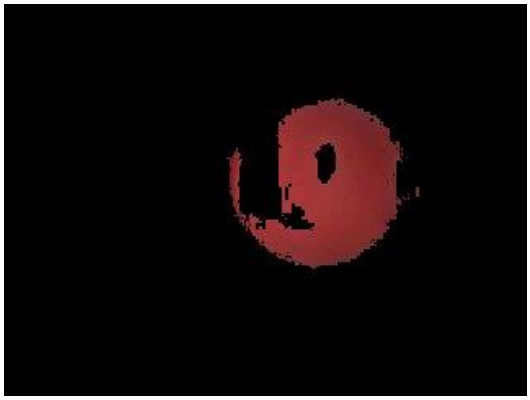
Cara kerjanya adalah dengan membandingkan komponen warna setiap piksel citra dengan warna spesifik. Apabila warnanya sesuai dengan warna spesifik komponen warna piksel tersebut maka dibiarkan saja. Namun, bila warnanya tidak sesuai dengan warna spesifik maka komponen warna piksel tersebut diubah menjadi warna *background*, biasanya menjadi warna hitam.

Warna yang digunakan dalam *Color Filtering* dapat direpresentasikan dalam berbagai ruang warna. Ada beberapa ruang warna yang dikenal, antara lain RGB (*Red*, *Green*, *Blue*), HSL (*Hue*, *Saturation*, *Luminance*), YCbCr, dan sebagainya. Dalam penelitian ini, peneliti menggunakan ruang warna RGB.

Terdapat penelitian yang dilakukan oleh Andrew Krillov yang menggunakan metode *color filtering* pada warna bola berwana merah. Contoh hasil penggunaan *color filtering* [13]:



**Gambar 2.3 Gambar Awal Sebelum Penggunaan *Color Filtering***



**Gambar 2.4 Gambar Akhir Setelah Penggunaan *Color Filtering***

## 2.8 *Blobs Processing*

*Blobs processing* adalah suatu teknik pengolahan citra yang digunakan untuk mengenali suatu objek secara utuh [13]. Berikut beberapa penggunaan *blobs processing*:

### 2.8.1 *Blob Counter*

*Blob counter* digunakan untuk mencari objek, menghitung objek, mengekstrak objek dan mendapatkan dimensi objek yang ada pada gambar, yang dipisahkan dari latar belakang. Sebagai contoh penggunaan *Blob counter* terdapat pada penjelasan di bawah ini:



**Gambar 2.5 Gambar Sumber**

Dengan menggunakan algoritma *blob counter* dapat menentukan dan mencari objek berdasarkan ukuran terbesar dan mengekstrak gambar sebagai *blob* terbesar:

```
// create an instance of blob Counter algorithm
BlobCounterBase bc = new BlobCounter( );
// set filtering options
bc.FilterBlobs = true;
bc.MinWidth  = 5;
bc.MinHeight = 5;
// set ordering options
bc.ObjectsOrder = ObjectsOrder.Size;
// process binary image
bc.ProcessImage( image );
Blob[] blobs = bc.GetObjectsInformation( );
// extract the biggest blob
if ( blobs.Length > 0 )
{
bc.ExtractBlobsImage( image, blobs[0], true );
}
```

Maka hasil dari algoritma *blob counter* tersebut, seperti gambar berikut:



**Gambar 2.6 Gambar hasil proses *Blob Counter***

### 2.8.2 *Extract* Bigest *Blob*

Jika pada Gambar 2.6 menjadi gambar sumber dan yang akan dikerjakan hanya mencari *blob* terbesar saja, *code* dapat disederhanakan dengan mengekstrak *blob* terbesar:

```
// create filter
ExtractBiggestBlob filter = new ExtractBiggestBlob( );
// apply the filter
Bitmap biggestBlobsImage = filter.Apply( image );
```



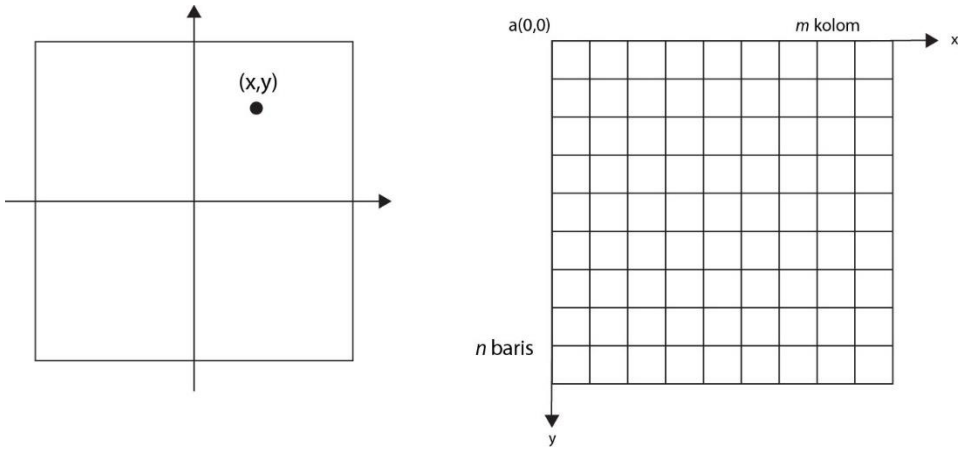
**Gambar 2.7 Hasil Ekstraksi *Blob* Terbesar**

Penggunaan *blob processing* sangat berguna bagi aplikasi *virtual drawing* yang akan dibangun, karena penentuan objek berbasis warna sangat berpengaruh pada penelitian ini.

## 2.9 Piksel

Sebuah piksel adalah sampel dari pemandangan yang mengandung intensitas citra yang dinyatakan dalam bilangan bulat [1]. Sebuah citra adalah kumpulan piksel-piksel yang disusun dalam larik dua dimensi. Indeks baris dan kolom (x,y) dari sebuah piksel yang dinyatakan dalam bilangan bulat. Piksel (0,0)

terletak pada sudut kiri atas pada citra, indeks bergerak ke kanan dan indeks bergerak ke bawah.




**Gambar 2.8 Hubungan antara koordinat pada citra dan indeks larik**

## 2.10 *Bitmap*

*Bitmap* adalah representasi dari citra grafis, seperti ukuran, format piksel yang merepresentasikan warna disimpan pada lokasi tertentu [2]. Saat ditampilkan pada layar komputer, setiap elemen digambarkan menjadi sebuah piksel tunggal.

## 2.11 Kamera dan Kamera *Video*

Kamera adalah kotak kedap sinar yang dipasang dengan lensa yang menyambung pada lubang lensa tempat gambar (objek) yang direkam dalam alat yang pekat cahaya. Kamera *video* merupakan sebuah kamera untuk merekam gambar pada rekaman *video* atau mengirimkan ke layar monitor.

Kamera sering dipakai dalam aktivitas fotografi. Nama kamera sendiri didapatkan dari kamera *obscura* bahasa latin untuk “ruang gelap” [14]. Kamera dapat digunakan seperti kamera *video*. Dalam beberapa hal kamera *video* pun disebut kamera seperti kamera yang mengambil foto dalam jangka waktu tertentu. Pada hal tersebut kamera dan kamera *video* tidak banyak perbedaan.

## 2.12 Kamera *Web* (*Webcam*)

Kamera *Web* atau *Web Camera* disingkat *Webcam* adalah kamera *video* yang terhubung dengan komputer yang diletakkan ditempat tertentu dan

tampilannya dapat dilihat pada sebuah jaringan (internet) [15]. Sesuai dengan sebutannya *Webcam* digunakan untuk aktivitas sebuah jaringan seperti komunikasi atau *broadcasting* secara langsung di internet. *Webcam* lebih sederhana jika dibandingkan dengan kamera *video* lainnya.

Sebuah *Webcam* sederhana terdiri dari sebuah lensa standar, dipasang di sebuah papan sirkuit untuk menangkap sinyal gambar dan memiliki sebuah lensa pada *casing* depan yang berguna untuk mengambil gambar, kabel pendukung, yang dibuat dari bahan yang fleksibel, salah satu ujungnya dihubungkan dengan papan sirkuit dan ujung satu lagi memiliki konektor.

Keuntungan dari *Webcam* harganya murah dan mudah dalam menggunakannya. Perkembangan teknologi dan minat penggunaan komunikasi secara *online* membuat tampilan komputer jadi lebih terintegrasi dengan kamera (*built-in*) seperti laptop.

## 2.13 *AForge.NET*

*AForge.NET* merupakan *open source Framework* yang dibangun oleh Andrew Kirilliov. *AForge.NET* merupakan *C# Framework* yang sangat membantu para developers dan penelitian khususnya di bidang *computer vision* dan kecerdasan buatan – pengolahan citra, *neural networks*, algoritma genetik, *machine learning*, robotika, dll.

### 2.13.1 *Library AForge.NET*

*Framework* ini terdiri dari berbagai *library* dan contoh dari *library AForge.NET* sebagai berikut [13] :

1. *AForge.Imaging* - *library* dengan pengolahan citra dan filter;
2. *AForge.Vision* - *library computer vision*;
3. *AForge.Video* - *library* untuk pemrosesan *video*;
4. *AForge.Neuro* - *library* untuk perhitungan jaringan saraf;
5. *AForge.Genetic* - *library* pemrograman evolusi;
6. *AForge*.Fuzzy - *library* perhitungan fuzzy;

7. *AForge.Robotics* - *library* yang membantu dukungan pada bidang robotika.

Keuntungan menggunakan *AForge.NET Framework* ini yaitu *class* yang disediakan oleh *AForge.NET Framework* dapat langsung digunakan sebagai API (*Application Programming Interface*) di *C#* program, sehingga membuat pengembangan aplikasi lebih cepat dan lebih mudah.

### 2.13.2 *Library* Pengolahan Citra *AForge.NET*

*Library* pengolahan citra pada *AForge.NET Framework* memiliki satu set filter pengolohan citra dan dibangun untuk berbagai macam tugas yang ditujukan pada *computer vision* dan analisis/pengolahan citra. *Library* pada filter pengolahan citra sangat banyak dipakai maka perkembangannya pun semakin banyak dan baru. Contoh berbagai macam set filter pengolahan citra pada *library* pengolahan citra pada *AForge.NET Framework* diantaranya :

1. *Color filtering* (*grayscale*, sepia, *invert*, *rotate channels*, *channel extrac*tion, *channel replacing*, *channel filtering*, *Color filtering*, *Euclidean Color filtering*, RGB *channels linear correction*).
2. HSL filter (*linear correction*, brightness, contrast, *Saturation*, *Hue* modifier, HSL *filtering*).
3. YCbCr *filter* (*linear correction*, YCbCr *filtering*, *channel extrac*tion/replacement).
4. Deteksi tepi (*homogeneity*, *difference*, *sobel*, *canny*).
5. *Blob counter* dan *connected components labeling filter*.
6. *Resize* dan *rotation* (*nearest neighbor, bilinear, bicubic*).

### 2.13.3 Piksel *Filtering* Berdasarkan Warna Pada *AForge.NET*

*AForge.NET Framework* memberikan pilihan filter pengolahan gambar, yang memungkinkan untuk menyaring piksel tergantung pada nilai-nilai warna. Filter pengolahan gambar ini dapat digunakan untuk menyimpan piksel, yang

warnanya berada di dalam atau di luar jangkauan tertentu, dan mengisi sisa piksel dengan warna tertentu [13]

Di bawah ini adalah contoh hasil piksel *filtering* berdasarkan warna dengan sumber gambar yang sama di bawah ini:



**Gambar 2.9 Gambar Sumber**

Filter pengolahan citra dengan menyaring piksel yang berada di dalam/luar dengan rentang tertentu pada warna RGB membuat piksel yang berada di luar jangkauan/sisanya diisi dengan warna tertentu.



**Gambar 2.10 *Color filtering***

## 2.14 *C#*

Bahasa pemrograman *C#* adalah salah satu bahasa menengah yang digunakan programmer untuk membuat program atau aplikasi. *C#* merupakan perkembangan dari bahasa pemograman C dan *C++* (diucapkan "C plus plus") [16].

Keuntungan menggunakan *C#* diantaranya yaitu:

1. Fleksibel

Program *C#* dapat dijalankan mesin yang sedang berkembang saat ini atau dapat ditransmisikan melalui *Web* dan dijalankan pada komputer lain.

2. *Powerful*

   *C#* memiliki dasar perintah yang sama seperti *C++* hanya saja lebih handal

3. *Easier to Use*

   *C#* lebih mudah dalam pencarian jika terjadi error daripada *C++*. Jadi lebih mempercepat waktu melihat error tersebut.

## 2.15 *Visual Studio 2012*

*Visual Studio 2012* adalah aplikasi pemrograman yang mempunyai fungsi yang dapat digunakan untuk keperluan proyek-proyek *C#* besar atau kecil. *Visual Studio 2012* dapat digunakan pada *Windows 7* dan *Windows 8*. *Visual Studio 2012* dapat menggabungkan modul yang ditulis dalam bahasa pemograman yang berbeda seperti *C++*, *Visual Basic*, dan F#. [17]

Keuntungan menggunakan *Visual Studio 2012* diantaranya yaitu:

1. *Visual Studio* mengotomatiskan langkah yang diperlukan untuk mengkompilasi *code* sumber tapi pada saat yang bersamaan memberikan kendali atas pilihan yang digunakan jika ingin mengubahnya.
2. *Visual Studio* teks editor disesuaikan dengan bahasa *Visual Studio* mendukung termasuk *C#* sehingga dapat mendeteksi kesalahan dengan cerdas dan menyarankan *code* jika diperlukan karena fitur ini disebut IntelliSense.
3. *Visual Studio* termasuk desainer untuk XAML ASP.NET, dan bahasa UI lainnya memungkinkan mendesain elemen UI dengan drag-and-drop.
4. *Visual Studio* memberikan fitur debugging yang dapat membantu mengembangkan proyek-proyek lalu di saat yang bersamaan dapat memantau dan mengawasi *code* intruksi yang sedang digunakan.

## 2.16 UML

*Unified Modeling* Language (UML) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan s*oft*ware berbasis OO (Object-Oriented) [18]. UML mulai diperkenalkan oleh Object Management Group, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM. UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi.

UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh grady Booch, Jim Rumbaugh dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

### 2.16.1 *Diagram* UML

UML menyediakan 10 macam *Diagram* untuk memodelkan aplikasi berorientasi objek, yaitu :

1. *Use Case Diagram* untuk memodelkan proses bisnis
2. *Conceptual Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi
3. *Sequence Diagram* untuk memodelkan pengiriman pesan (message) antar objek
4. Collaboration *Diagram* untuk memodelkan interaksi antar objek
5. State *Diagram* untuk memodelkan perilaku objek di dalam sistem
6. *Activity Diagram* untuk memodelkan perilaku *Use Case* dan Objek di dalam sistem
7. *Class Diagram* untuk memodelkan struktur kelas

8. Object *Diagram* untuk memodelkan struktur object
9. Component *Diagram* untuk memodelkan komponen object
10. *Deployment Diagram* untuk memodelkan distribusi aplikasi

Berikut akan dijelaskan macam- macam *Diagram* yang digunakan dalam pembangunan aplikasi berorientasi objek, yaitu [19]:

1. *Use Case Diagram,* untuk memodelkan proses bisnis dan merepresentasikan sebuah interaksi antara aktor dengan sistem
2. *Sequence Diagram*, untuk menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus 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 yakni nama (dan stereotype) , atribut , dan metoda.
3. Statechart *Diagram* , untuk memodelkan perilaku objek di dalam sistem atau menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada sistem. Pada umumnya statechart *Diagram* menggambarkan *class* tertentu.
4. *Activity Diagram* , untuk memodelkan perilaku *Use Case* dan objek di dalam sistem.
5. *Sequence Diagram* , untuk memodelkan pengiriman pesan (message) antar objek dan juga digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu.
6. Collaboration *Diagram* , untuk memodelkan interaksi antar objek seperti *sequence Diagram,* tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message
7. Component *Diagram* , untuk memodelkan komponen objek atau menggambarkan struktur dan hubungan antar modul yang berisi *code*, termasuk ketergantungan (dependency) di antaranya.
8. *Deployment Diagram,* untuk memodelkan distribusi aplikasi