BAB II

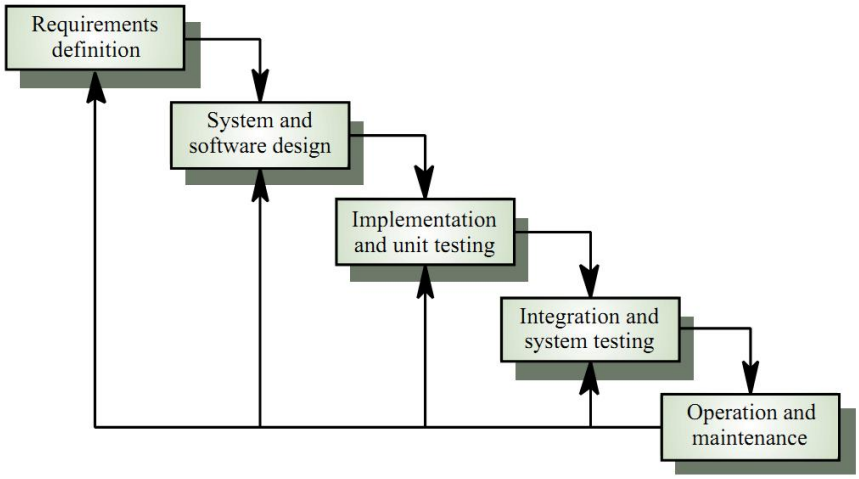
LANDASAN TEORI

II.1 Metode Pembangunan Perangkat Lunak

Dalam proses pembangunan perangkat lunak digunakan beberapa model. Adapun model-model tersebut yaitu :

II.1.1 Waterfall Model

Model proses *waterfall* merupakan dasar proses aktifitas pembangunan suatu perangkat lunak. Model proses *waterfall* mempunyai beberapa tahapan yang berurutan seperti yang dapat dilihat pada gambar berikut ini :



Gambar II-1 Waterfall Model

Setiap tahapan mempunyai arti tersendiri dalam proses pembangunan suatu perangkat lunak serta tahapan-tahapan tersebut harus dilakukan secara berurutan. Berikut ini merupakan penjelasan dari setiap tahapan yang ada dalam *waterfall* model :

a. *Requirements analysis and definition*

Tahap ini adalah tahap untuk mengumpulkan kebutuhan secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh program yang akan dibangun. Tahap ini harus dikerjakan secara lengkap untuk bisa menghasilkan desain yang lengkap.

b. *System and software design*

Tahap ini adalah tahap mendesain sistem dan perangkat lunak yang dikerjakan setelah kebutuhan selesai dikumpulkan secara lengkap.

c. *Implementation and unit testing*

Tahap ini mendesain sistem dan program maka pada tahap ini desain diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji baik secara unit.

d. *Integration and system testing*

Pada tahap ini unit-unit program disatukan kemudian diuji secara keseluruhan (system testing).

e. *Operation and maintenance*

Tahap ini program dioperasikan pada lingkungannya dan dilakukan pemeliharaan, seperti penyesuaian atau perubahan karena adaptasi dengan situasi sebenarnya. [4]

II.1.2 Object Oriented Programing (OOP)

Object Oriented Programming atau yang biasa disebut sebagai OOP merupakan pola pikir manusia dalam proses pembangunan perangkat lunak. Semua yang berhubungan dengan interaksi sistem dalam OOP diprensentasikan dalam objek [5]. Terdapat beberapa komponen dalam OOP, berikut ini merupakan komponen-komponennya :

1. Objek

   Objek merupakan struktur yang dapat melakukan serangkaian kegiatan di dalam sebuah perangkat lunak.

2. Class

   Class merupakan rancangan awal yang menjelaskan kemampuan yang dapat dilakukan oleh objek. Class terdiri dari kumpulan variabel, properti dan method yang merupakan identitas dari suatu class.

3. Enkapsulasi

   Enkapsulasi merupakan pembungkusan *method* yang mempunyai hak akses *private* dalam suatu *class,* sehingga orang lain yang ingin menggunakan class tersebut tidak perlu tahu detailnya, hanya cukup mengetahui method dan properti yang ada dalam class tersebut serta kegunaanya.

4. Asosiasi

   Asosiasi merupakan relasi antara dua buah *class* dengan cara membuat objek dari *class* lain sebagai atribut dari *class* tersebut.

5. Komposisi

   Komposisi merupakan relasi antara dua buah *class* dimana relasi antara dua buah *class* kuat, sehingga jika salah satu *class* tidak ada maka *class* yang lain pun tidak ada.

II.1.3 UML (Unified Modeling Language)

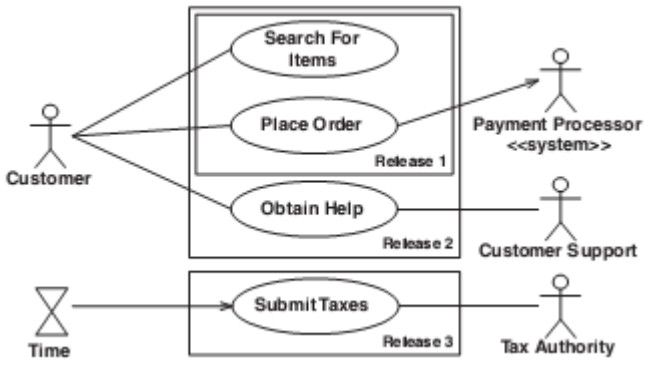
UML merupakan bahasa standar untuk merancang dan mendokumentasikan perangkat lunak dengan cara yang berorientasi objek. Ada beberapa diagram yang digunakan proses pembuatan perangkat lunak. Berikut adalah diagram-diagram tersebut :

1. *Use Case* Diagram

   Use case diagram merupakan diagram hubungan antara aktor dan *use case*, dimana *use case* menunjukkan layanan yang diberikan oleh sistem kepada aktor. Terdapat dua hal utama yang diperlukan dalam pembentukan suatu *use case* diagram yaitu aktor dan *use case.*

   a. Aktor merupakan orang, benda maupun sistem lain yang berinteraksi dengan sistem yang akan dibangun.

   b. *Use case* merupakan fungsionalitas atau layanan yang disediakan oleh sistem.



Gambar II-2 Contoh Use Case Diagram [6]

2. *Activity* Diagram

Activity diagram merupakan diagram yang menggambarkan fungsionalitas yang ada di dalam sistem. Setiap *use case* yang telah dibentuk digambarkan aktifitasnya dalam *Activity diagram*, mulai dari peran aktor, peran sistem, dan *decision.*

3. *Sequence* Diagram

Sequence diagram digunakan untuk menggambarkan fungsionalitas yang terjadi antar *object* yang ada didalam class.

4. *Class* Diagram

Class diagram menunjukkan interaksi dan relasi antar *class* yang ada di dalam sistem. Sebuah class memiliki nama *class, attributes, methods(operations).*

Nama *Class*
Attributes
Methods (*operations*)

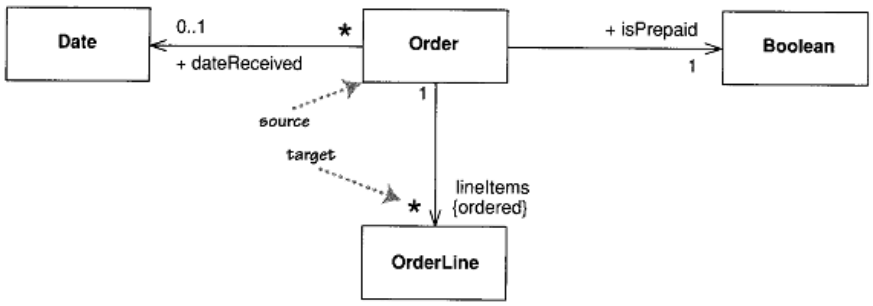
Gambar II-3 Class

a. Nama *Class* merupakan nama dari sebuah class yang digunakan dalam sebuah sistem.
b. *Attributes* merupakan variabel-variabel yang dimiliki oleh suatu *class*.
c. *Methods* merupakan fungsionalitas atau pekerjaan yang dilakukan oleh suatu class.

Class diagram menggambarkan relasi atau hubungan antar *class* dari sebuah sistem. Berikut ini beberapa gambaran relasi yang ada dalam *class* diagram.

1. Asosiasi

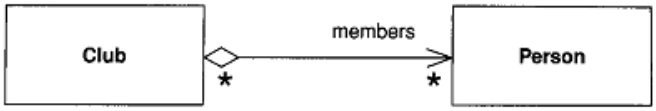
   Asosiasi merupakan hubungan antar *class* yang statis. Class yang mempunyai relasi asosiasi menggunakan class lain sebagai atribut pada dirinya. Asosiasi digambarkan dengan garis panah lurus.



Gambar II-4 Asosiasi [7]

2. Agregasi

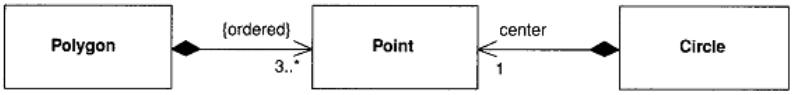
   Agrerasi merupakan relasi yang membuat class yang saling berelasi terikat satu sama lain namun tidak terlalu berkegantungan.



Gambar II-5 Agregasi [7]

3. *Composition*

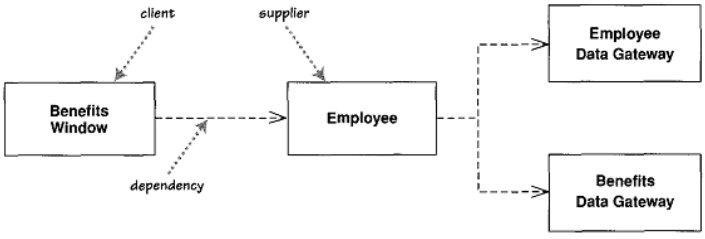
   Composition merupakan relasi agregasi dengan mengikat satu sama lain dengan ikatan yang sangat kuat dan saling berkegantungan.



Gambar II-6 *Compostion* [7]

4. *Dependency*

   Dependency merupakan hubungan antar-*class* di mana class yang memiliki relasi *dependency* menggunakan *class* lain sebagai *attribute* pada *method.*

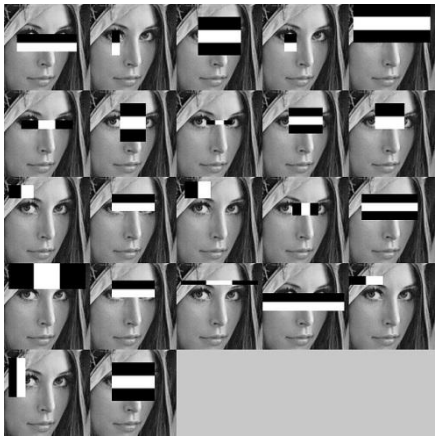


Gambar II-7 *Dependency* [7]

II.2 Deteksi Wajah (*Face Detection)*

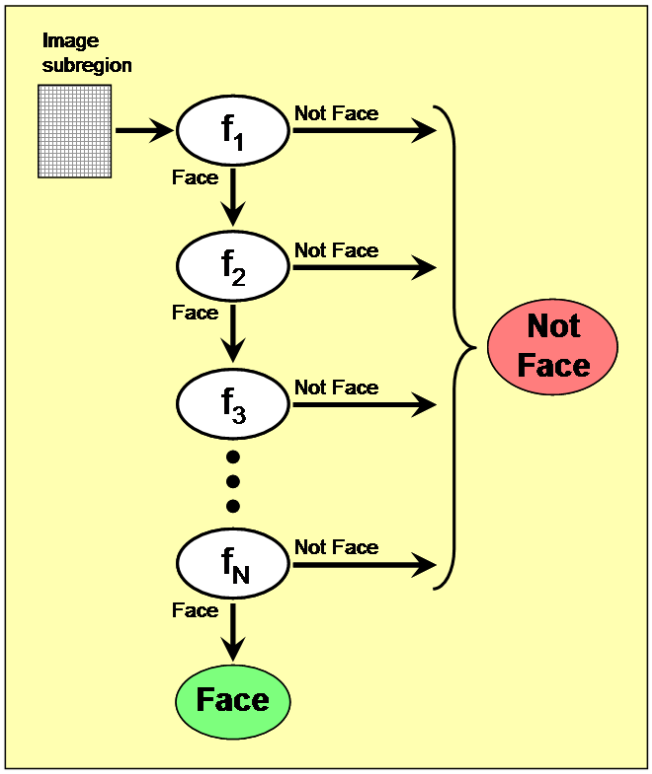
1. Metode *Viola Jones*

Metode *Viola Jones* merupakan metode yang paling banyak digunakan dalam proses pengenalan wajah. Metode *Viola Jones* menggunakan data latih yang digunakan dalam proses pengklasifikasian citra. Klasifikasi citra dilakukan berdasarkan nilai dari sebuah fitur. Fitur-fitur ini merupakan gambaran dari wajah manusia yang biasa disebut sebagai *Haar Feature*.



Gambar II-8 Haar *Feature*

Untuk menghitung nilai dari setiap fitur *Haar* maka digunakan citra integral. Setiap fitur mempunyai nilai yang berbeda sehingga dibutuhkan cara untuk mendapatkan fitur yang memiliki nilai yang paling baik. Algoritma *Adaboost* digunakan untuk memilih setiap fitur yang memiliki nilai terbaik dengan cara memfilter setiap gambar masukkan dan membandingkannya dengan ambang batas yang bersifat dinamis.



Gambar II-9 Algoritma adaboost

2. Metode *Roberts Cross Edge Detection*

Metode Roberts Cross *Edge Detection* merupakan satu dari metode pengenalan wajah yang pertama dan yang mulanya diusulkan oleh Lawrence Roberts pada tahun 1963 *[8]*. Metode *Roberts Cross Edge Detection* menggunakan teknik segmentasi warna, teknik segmentasi citra dan algoritma *Roberts Cross Edge detection* sebagai cara untuk mendeteksi wajah. Teknik segmentasi warna merupakan teknik yang mengubah gambar normal yang berformat RGB menjadi gambar yang mempunyai format YCbCr.



Gambar II-10 Segmentasi Warna

Teknik ini digunakan sebagai cara untuk mendeteksi warna kulit, sehingga gambar yang dihasilkan merupakan gambar yang berwarna abu-abu.

Teknik segmentasi citra merupakan teknik yang mengubah gambar abu-abu yang telah dihasilkan oleh teknik segmentasi warna menjadi gambar biner yang merupakan gambar hitam-putih.



Gambar II-11 Segmentasi Citra

Algoritma Roberts Cross Edge detection merupakan salah satu algoritma yang digunakan untuk mendeteksi tepi. Algoritma Roberts Cross Edge detection mengubah gambar menjadi gambar dua dimensi dan menentukan gradien gambar.



Gambar II-12 *Roberts Cross Edge detection*

II.3 Pengenalan Wajah *(Face Recognition*) dengan menggunakan metode Principal Component Analysis (PCA)

Dalam konteks pengenalan wajah, PCA mencari eigenface yang merupakan kumpulan dari eigenvector. Eigenface adalah principal component (ciri-ciri penting) dari distribusi citra wajah yang didapatkan dari eigenvector dengan eigenvalue terbesar. Untuk mendapatkan eigenface, PCA melakukan perhitungan matrik kovarian dari kumpulan citra wajah latih setelah dilakukan normalisasi. Eigenface tersebut akan menjadi dasar perhitungan face space yang merepresentasikan nilai bobot individu yang mewakili satu atau lebih citra wajah. Nilai bobot inilah yang digunakan untuk mengenali citra wajah uji dengan mencari jarak nilai bobot citra wajah uji dengan nilai bobot citra wajah latih [9].

II.4 Pembangun Perangkat Lunak

Dalam membangun perangkat lunak dibutuhkan alat-alat pembangun yang berupa software dan library. Berikut ini merupakan alat-alat yang digunakan dalam proses pemgbangunan perangkat lunak :

1. C# (C Sharp)

C# dibaca C Sharp merupakan bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft yang merupakan bagian dari .Net dan kemudian disetujui oleh ECMA sebagai bagian dari bahasa pemrograman. C# merupakan bahasa pemrograman yang dikembangkan oleh Anders Hejlsberg dari C++ pada tahun 1999 [10].

2. Emgu CV

Emgu CV merupakan salah satu library pengembangan Open CV yang digunakan pada bahasa pemrograman C#. Emgu CV merupakan library yang *multi platform* sehingga library ini dapat digunakan di platform yang mendukung Mono, termasuk Linux, Mac OS X, IOS dan Android [11].

3. Visual Studio 2010

Visual Studio 2010 merupakan alat untuk membangun sistem yang dibangun oleh Microsoft dengan berbasis .Net *Framework*. Visual Studio 2010 dapat digunakan sebagai alat pengembangan perangkat lunak dan sistem berbasis desktop, mobile dan web.