BAB 2

TINJAUAN PUSTAKA

2.1 Tinjauan Perusahaan

2.1.1 Sejarah Toko Sakinah Kerudung

Toko Sakinah Kerudung ini didirikan pada tahun 2007 oleh pasangan suami istri yaitu Bapak Muchammad Mudrik dan Ibu Dian Lestari. Toko ini bergerak dalam bisnis busana muslim dan kerudung. Usaha yang dijalani pasangan suami istri ini berawal dari memiliki konveksi kerudung rumahan pada tahun 2001. Mereka menjual kerudung hasil produksi ke beberapa pedagang eceran, maupun di titip ke toko-toko. Namun yang namanya bisnis pasti ada saatnya di puncak kesuksesan dan ada kalanya di bawah ambang kebangkrutan. Sempat mengalami kebangkrutan karena barang-barang yang tidak dibayar oleh para *reseller*. Sehingga pada tahun 2004-2005 , sempat berhenti berproduksi dan membangun kembali usahanya dengan melakukan semuanya dari awal.

Usaha busana muslim dan kerudung ini akhirnya bangkit kembali di tahun 2007 dengan mendirikan Toko Sakinah dan menjual sendiri hasil produksinya. Diawali dengan mengontrak sebuah toko kecil yang berukuran kira-kira 16 meter persegi, yang berlokasi di jalan raya Jatinangor nomor 131 (dekat UNPAD Jatinangor) Sumedang, sehingga usaha keluarga ini makin berkembang. Tidak hanya menjual busana muslim dan kerudung tetapi juga aneka aksesoris kerudung, walaupun lebih terkonsentrasi pada penjualan kerudung. Dengan mengikuti terus perkembangan model kerudung masa kini, toko ini semakin maju hingga mampu

menambah lima cabang di berbagai daerah di Bandung. Toko Sakinah Kerudung pada saat ini sudah memiliki enam cabang toko yang di antaranya bertempat di jalan raya Jatinangor nomor 131 (dekat UNPAD Jatinangor) Sumedang, jalan raya Sukamiskin nomor 134 Bandung, jalan Soekarno Hatta nomor 690A Riung Bandung, jalan Gegerkalong Girang nomor 41 Bandung, jalan Ahmad Nasution nomor 83A (dekat UIN Cibiru) Bandung, dan jalan Ahmad Yani nomor 674 Cicadas. Selain itu Toko Sakinah Kerudung juga mengembangkan usahanya melalui web yang telah mendukung transaksi secara *online* di alamat website www.sakinahkerudung.com.

2.1.2 Visi

Visi dari toko ini ialah menjadi agen pembaharuan di dalam masyarakat dan menyediakan produk – produk unggulan yang inovatif dan berorientasi pada pasar.

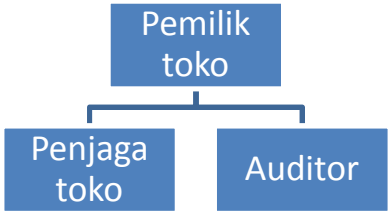
2.1.3 Misi

Toko ini memiliki misi memasyarakatkan kerudung di kalangan masyarakat indonesia dan memfasilitasi para wanita muslim untuk tetap tampil muslimah dan modis.

2.1.4 Struktur Organisasi

Struktur organisasi merupakan susunan yang terdiri dari fungsi-fungsi dan hubungan yang menggambarkan keseluruhan kegiatan untuk mencapai suatu sasaran. Secara fisik struktur organisasi dapat digambarkan dalam bentuk bagan yang memperlihatkan hubungan unit-unit organisasi dan garis-garis wewenang yang ada.

Beberapa keuntungan yang dapat diperoleh dari penggunaan bagan organisasi adalah dapat memperlihatkan karakteristik utama dari suatu perusahaan tersebut, tentang gambaran pekerjaan dan hubungan-hubungan yang ada di dalam perusahaan serta digunakan untuk merumuskan rencan kerja yang ideal sebagai pedoman untuk dapat mengetahui siapa bawahan dan atasannya. Struktur keorganisasian pada Toko Sakinah Kerudung dapat dilihat pada bagan struktur berikut ini :



Gambar 2. 1 Struktur Organisasi

2.1.5 Deskripsi Tugas

Deskripsi tugas digunakan untuk mengetahui tugas, wewenang, tanggung jawab dari masing-masing bagian. Deskripsi tugas yang ada di Toko Sakinah Kerudung adalah sebagai berikut :

1. Pemilik Toko
    a. Sebagai pengambil keputusan.
    b. Sebagai koordinator semua kegiatan yang telah dilaksanakan.
    c. Mengevaluasi semua kegiatan yang telah dilaksanakan.
    d. Menerapkan dan mengesahkan kebijakan yang menyangkut eksistensi toko.
2. Penjaga toko
    a. Melayani pembeli secara langsung.

b. Menghitung hasil penjualan dan bertanggung jawab atas uang yang masuk.
c. Melaporkan hasil penjualan secara harian, mingguan, bulanan, dan periodik
d. Melaporkan ke bagian auditor jika stok barang di cabang kurang atau habis.

3. Auditor
   a. Mengecek stok barang di tiap cabang.
   b. Mengirim stok barak ke tiap cabang.
   c. Mengecek laporan penjualan dari tiap cabang.
   d. Menjadi admin penjualan di web toko.

2.2 Landasan Teori

2.2.1 Konsep PBO (Pemrograman Berorientasi Objek)

Object Oriented Programming (OOP) atau Pemrograman Berorientasi Objek (PBO) adalah konsep pemrograman yang difokuskan pada penciptaan kelas yang merupakan abstraksi/*blueprint/prototype* dari suatu objek [3]. Sedangkan sistem berorientasi objek merupakan sebuat sistem yang dibangun berdasarkan metode berorientasi objek. Setiap komponen dalam sistem tersebut dapat mewarisi atribut dan sifat dari komponen lainnya, serta dapat berinteraksi satu sama lain.

Untuk membangun sebuah sistem berorientasi objek harus digunakan juga Pemprograman berorientasi objek (PBO) dapat menggunakan bahasa

pemrograman berorientasi objek seperti Java, C#, C++, Smalltalk, Eiffel dan PHP. Ada banyak cara untuk mengabstraksikan dan memodelkan objek – objek tersebut, mulai dari abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Pada saat mengabstraksikan dan memodelkan objek, data, dan proses – proses yang dipunyai oleh objek akan dienkapsulasi (dibungkus) menjadi satu kesatuan.

Dalam rekayasa perangkat lunak sendiri, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing - masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu. Berikut adalah komponen – komponen yang terdapat pada sistem berorientasi objek :

2.2.1.1 Kelas (*Class)*

Kelas adalah pemodelan dari objek yang berisi informasi (aturan) tentang sifat karakteristik (data) dan tingkah laku (*method*) yang dimiliki oleh objek tersebut [3]. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (metode), hubungan (*relationship)* dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

2.2.1.2 Objek (*object)*

Objek merupakan *instans* (perwujudan) dari suatu kelas. Maksudnya, objek merupakan peruwujudan dalam bentuk benda, baik berupa benda nyata, maupun

konsep, sesuai dengan deskripsi yang telah didefinisikan oleh kelas [3]. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan dimanipulasi dan dihancurkan.

2.2.1.3 Metode (*Method*)

Metode adalah prosedur atau fungsi yang dimiliki oleh sebuah objek. Sebuah kelas boleh memiliki lebih dari satu metode yang berfungsi untuk memanipulasi objek itu sendiri. Metode juga merupakan cara objek untuk berkomunikasi antar objek.

2.2.1.4 Atribut (*Attribute*)

Atribut adalah sifat karakteristik atau kondisi yang dimiliki oleh suatu objek [3]. Atribut dapat berupa nilai atau elemen – elemen data yang dimiliki oleh objek dalam kelas objek. Contoh manusia memiliki atribut status, berat badan, dan tinggi badan. Di dalam sebuah kelas atribut disebut variabel.

2.2.1.5 Enkapsulasi (*Encapsulation*)

Enkapsulasi ialah lapisan pelindung yang mencegah kode dan data yang secara acak diakses oleh kode lain yang didefinisikan di luar kelas [4]. Oleh karena itu dibutuhkan acces modifier dalam penggunaannya, yaitu public, private, dan protected.

2.2.1.6 Pewarisan (*Inheritance*)

Mekanisme yang memungkinkan saatu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

2.2.1.7 Antarmuka (*interface)*

Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

Sebuah kelas dapat mengimplementasikan lebih dari satu antarmuka dimana kelas ini akan mendeklarasikan metode pada antarmuka yang dibutuhkan oleh kelas itu sekaligus mendefinisikan isinya pada kode program kelas itu. Metode pada antarmuka yang diimplementasikan pada suatu kelas harus sama persis dengan yang ada pada antarmuka.

2.2.1.8 *Reusability*

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

2.2.1.9 Generalisasi dan Spesialisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil,motor dan kereta.

2.2.1.10 Polimorfisme (*Polymorfism)*

Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

2.2.1.11 *Package*

Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas - kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.2.2 Konsep UML *(Unified Modeling Language)*

Sebelum melangkah pada proses *coding* hal yang dilakukan yaitu memodelkan. Permodelan dalam dunia pembangunan perangkat lunak digunakan dalam proses merancang perangkat lunak sebelum melakukan pengkodean (*coding*) tujuannya untuk mempermudah langkah berikutnya dari pembangunan perangkat lunak sehingga lebih terencana dan semua kebutuhan pengguna dapat terpenuhi dengan lengkap dan tepat, termasuk faktor-faktor seperti *scalability*, *robustness*, *security*, dan sebagainya.

Unified Modelling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek [5]. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi perancangan berorientasi objek. Pada tahun 1995 dirilis draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG). Tahun 1997 UML versi 1.1 muncul, dan hingga saat ini telah ada versi 2.4.1 yang terdiri dari 4 macam spesifikasi, yaitu *Diagram Interchange Sfesification*, UML *Infratructure*,

UML *Super Structure* dan *Object Constraint Language* (OCL). Seluruh spesifikasi tersebut dapat diakses di http://www.omg.org .

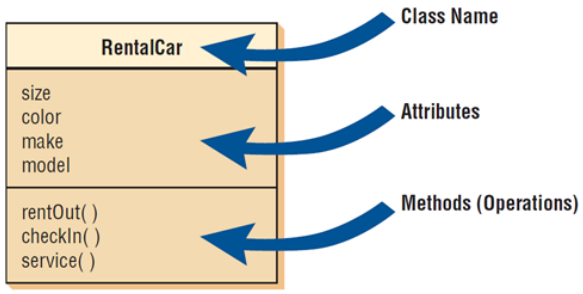
2.2.2.1 Diagram UML

Pada UML versi 2.4 terdapat 2 kategori diagram sebagai berikut:

1. *Structure Diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behaviour Diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

2.2.2.2 *Class Diagram*

Class Diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka [5]. *Class Diagram* bersifat statis di dalam *class diagram* digambarkan relasi dari masing - masing kelas tetapi tidak menggambarkan apa yang terjadi ketika kelas tersebut berelasi. Sebuah Kelas memiliki struktur sebagai gambar berikut:



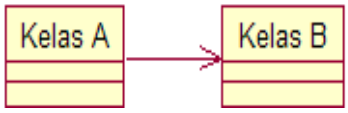
Gambar 2. 2 *Class Diagram* [6]

1. Nama Kelas/*Stereotype*
2. Atribut, yaitu variabel-variabel yang dimiliki suatu kelas.
3. Metode, yaitu fungsi-fungsi yang dimiliki oleh suatu kelas.

Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class Diagram* mempunyai 3 macam hubungan, sebagai berikut :

1. *Association*

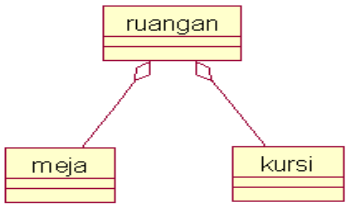
   Assosiation merupakan sebuah garis solid antara dua kelas ditarik dari kelas sumber ke kelas target [5]. Dalam notasi UML kita mengenal asosiasi 2 arah (bidirectional) dan 1 arah (undirectional).



Gambar 2. 3 Relasi *Association* [6]

2. *Aggregation*

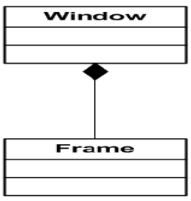
   Aggregation adalah *association* dimana salah satu kelasnya merupakan bagian dari suatu kumpulan. *Aggregation* memiliki titik pusat yang mencakup keseluruhan bagian.



Gambar 2. 4 Relasi *Aggregation* [6]

3. *Composition*

   Composition ialah *aggregation* dengan ikatan yang lebih kuat. Di dalam *composite aggregation*, siklus hidup *part class* sangat bergantung pada *whole class* sehingga bila objek *instance* dari *whole class* dihapus maka objek *instance* dari *part class* juga akan terhapus.



Gambar 2. 5 Relasi *Composition*[6]

4. *Depedency*

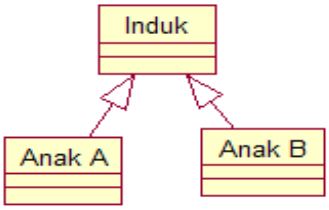
   Depedency ialah hubungan antar class di mana sebuah class memiliki ketergantungan pada class lainnya tetapi tidak sebaliknya.



Gambar 2. 6 Relasi *Depedency*[6]

5. *Generalization*

   Generalization diperlukan untuk memperlihatkan hubungan pewarisan (*inheritance*) antar unsur dalam diagram kelas. Pewarisan memungkinkan suatu kelas mewarisi semua atribut, operasi ,relasi, dari kelas yang berada dalam hirarki pewarisannya.



Gambar 2. 7 Relasi *Generalization*[6]

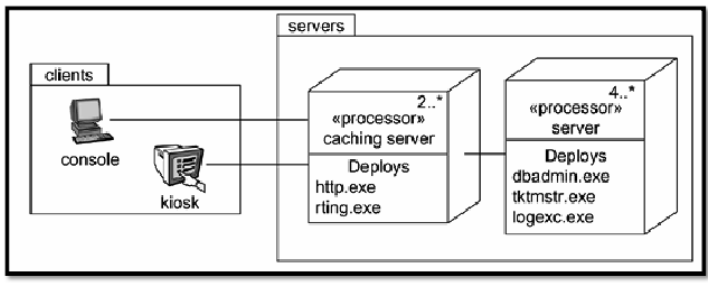
2.2.2.3 *Package Diagram*

Untuk mengatur pengorganisasian diagram *Class* yang kompleks, dapat dilakukan pengelompokan kelas-kelas berupa package (paket). *Package* adalah

sebuah bentuk pengelompokkan yang memungkinkan anda untuk mengambil setiap bentuk di UML dan mengelompokkan elemen-elemennya dalam tingkatan unit yang lebih tinggi [5].

2.2.2.4 *Deployment Diagram*

Deployment diagram menunjukkan susunan fisik sebuah sistem, menunjukkan bagian perangkat lunak mana yang berjlaan pada perangkat keras mana [5]. Hubungan antar node (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini, berikut ini contoh *deployment diagram*:



Gambar 2. 8 Contoh *Deployment Diagram*[6]

2.2.2.5 *Use Case Diagram*

Use case diagram merupakan teknik untuk merekam persyaratan fungsional sebuah sistem [5]. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibangun. *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu aktor dan *use case.*

1. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem yang akan dibangun.

2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor.

2.2.2.6 *Activity Diagram*

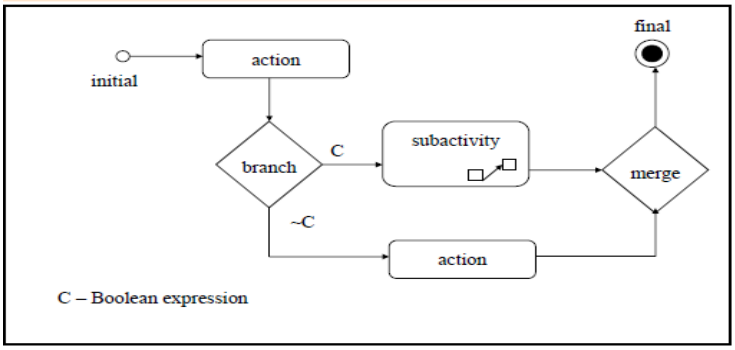
Activity diagram adalah teknik untuk menggambarkan logika, procedural, proses bisnis, dan jalur kerja [5]. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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 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.

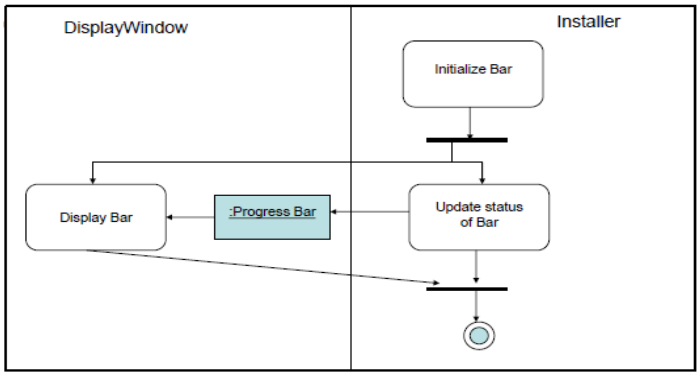
Seperti halnya *state*, standar UML menggunakan segi empat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Berikut ini contoh *activity diagram* :



Gambar 2. 9 Contoh *Activity Diagram*[7]

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu. Berikut ini adalah contoh *activity diagram* dengan *swimlane* :



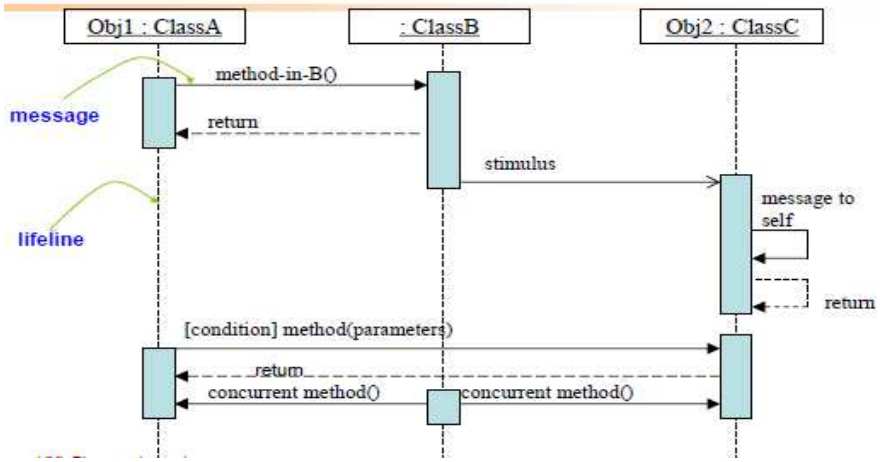
Gambar 2. 10 Object Swimelan [7]

2.2.2.7 Sequence Diagram

Sequence diagram secara khusus menjabarkan *behavior* sebuah skenario tunggal [5]. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah kejadian 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 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 simbol khusus untuk *objek boundary*, *controller* dan *persistent entity*. Contoh sequence diagram:



Gambar 2. 11 Contoh *Sequence Diagram*[7]

2.2.3 Pengolahan Citra

Data atau informasi tidak hanya disajikan dalam bentuk teks, tapi juga dapat berupa gambar, audio (bunyi, suara, musik) dan video. Keempat macam data atau informasi ini sering disebut multimedia. Citra istilah lain untuk gambar sebagai satu komponen multimedia memegang peranan penting sehingga bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi.

Pengolahan citra adalah pemrosesan citra menjadi citra yang lain dengan kualitas yang lebih baik, yaitu pemrosesan pada usaha untuk memanipulasi [8]. Citra yang telah menjadi gambar lain menggunakan algoritma atau teknik tertentu. Beberapa jenis operasi pengolahan citra adalah sebagai berikut :

2.2.3.1 Modifikasi Kecemerlangan (*Brightness Modification*)

Mengubah nilai keabuan/warna dari gelap menuju terang atau sebaliknya mengubah citra yang terlalu cemerlang/pucat menjadi gelap.

2.2.3.2 Peningkatan Kontras (*Contast Enhancement*)

Dengan peningkatan kontras maka titik yang cenderung gelap menjadi lebih gelap dan yang cenderung terang menjadi lebih cemerlang.

2.2.3.3 Negasi

Operasi untuk mendapatkan citra negatif (*negative image*)

2.2.3.4 Pengabuan (*grayscale)*

Merupakan proses konversi citra dengan warna sebenarnya (*true color*) menjadi citra keabuan (*grayscale*).

2.2.3.5 Pengambangan (*Thresholding*)

Operasi pengambangan digunakan untuk mengubah citra dengan format skala keabuan, yang mempunyai kemungkinan nilai lebih dari dua ke citra biner yang memiliki dua buah nilai (yaitu 0 dan 1).

2.2.3.6 Pencerminan (*Flipping*)

Pencerminan merupakan proses menggambar citra ke bentuk kebalikannya seperti ketika sedang bercermin.

2.2.3.7 Rotasi (*Rotating*)

Rotasi yaitu proses memutar koordinat citra sesuai derajat yang ditentukan.

2.2.3.8 Pemotongan (*Cropping*)

Memotong satu bagian dari citra sesuai kebutuhan.

2.2.3.9 Pengskalaan (*Scaling*)

Mengubah ukuran citra menjadi lebih besar atau lebih kecil.

2.2.3.10 Deteksi Tepi (*Edge Detection*)

Deteksi tepi pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra.

2.2.4 Metode *Haar Cascade Classifier*

Proses deteksi adanya citra wajah dalam sebuah gambar dapat menggunakan sebuah metode yang dipublikasikan oleh Paul Viola dan Michael Jones tahun 2001. Umumnya disebut metode *Haar Cascade Classifier*. Metode ini merupakan metode yang menggunakan statistik model (*classifier*). Pendekatan untuk mendeteksi wajah dalam gambar menggabungkan empat konsep utama :

1. *Training* data
2. Fitur segi empat sederhana yang disebut fitur Haar.
3. *Integral image* untuk pendeteksian fitur secara cepat.
4. Pengklasifikasi bertingkat (*Cascade classifier*) untuk menghubungkan banyak fitur secara efisien.

2.2.4.1 Training Data Pada *Haar*

Metode ini memerlukan 2 tipe gambar objek dalam proses *training* yang dilakukan, yaitu :

1. *Positive samples*

Berisi gambar obyek yang ingin dideteksi, apabila ingin mendeteksi mata maka *positive samples* ini berisi gambar mata, begitu juga objek lain yang ingin dikenali.

2. *Negative samples*

Berisi gambar objek selain objek yang ingin dikenali, umumnya berupa gambar *background* (tembok, pemandangan, lantai, dan gambar lainnya). Resolusi untuk sampel negatif disarankan untuk memiliki resolusi yang sama dengan resolusi kamera.

Training dari *Haar* menggunakan dua tipe sampel diatas. Informasi dari hasil *training* ini lalu dikonversi menjadi sebuah parameter model statistik.

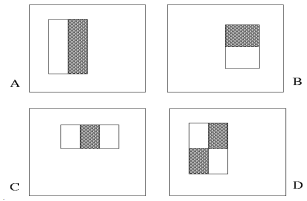
2.2.4.2 Sistem Kerja Algoritma *Haar Cascade Classifier*

Algoritma *Haar* menggunakan metode statistikal dalam melakukan pengenalan wajah. Metode ini menggunakan *simple haar-like features* dan juga *a cascade of boosted tree classifier*. *Classifier* ini menggunakan gambar berukuran tetap. Cara kerja dari algoritma haar dalam mendeteksi wajah adalah menggunakan teknik *sliding window* pada keseluruhan gambar dan mencari apakah terdapat bagian dari gambar yang berbentuk seperti wajah atau tidak. Haar juga memiliki kemampuan untuk melakukan *scaling* sehingga dapat mendeteksi adanya mata yang berukuran lebih besar ataupun lebih kecil dari gambar pada *classifier*. Tiap fitur dari *haar-like* fitur didefinisikan pada bentuk dari fitur, diantaranya koordinat dari fitur dan juga ukuran dari fitur tersebut.

2.2.4.3 Haar Feature

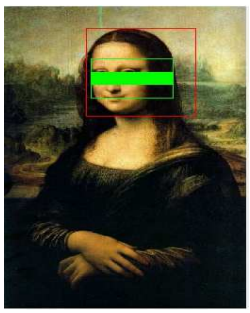
Haar Feature adalah fitur yang digunakan oleh Viola dan Jones didasarkan pada *Wavelet Haar*. *Wavelet Haar* adalah gelombang tunggal bujur sangkar (satu interval tinggi dan satu interval rendah) . Untuk dua dimensi, satu terang dan satu gelap. Selanjutnya kombinasi-kombinasi kotak yang digunakan untuk pendeteksian objek visual yang lebih baik. Ada tiga tipe kotak (*rectangular*) fitur pada umunya yaitu:

1. Tipe *two-rectangle feature* (horisontal/vertikal)
2. Tipe *three-rectangle feature*
3. Tipe *four-rectangle feature*



Gambar 2. 12 *Haar Feature*[9]

Adanya fitur Haar ditentukan dengan cara mengurangi rata-rata piksel pada daerah gelap dari rata-rata piksel pada daerah terang. Jika nilai perbedaannya itu diatas nilai ambang atau *threshold*, maka dapat dikatakan bahwa fitur tersebut ada.

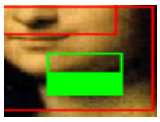


Gambar 2. 13 Contoh Penerapan Haar Feature

Nilai dari *Haar-like feature* adalah perbedaan antara jumlah nilai-nilai piksel *gray level* dalam daerah kotak hitam dan daerah kotak putih. Kotak *Haar-like feature* dapat dihitung secara cepat menggunakan *integral image.*

2.2.4.4 Integral Image

Integral Image digunakan untuk menentukan ada atau tidak fitur Haar pada sebuah gambar dan pada skala yang berbeda secara efisien. Pada umumnya, pengintegrasian tersebut berarti menambahkan unit-unit kecil secara bersamaan. Dalam hal ini unit-unit kecil tersebut adalah nilai-nilai piksel. Nilai integral untuk masing-masing piksel adalah jumlah dari semua piksel-piksel dari atas sampai bawah. Dimulai dari kiri atas sampai kanan bawah, keseluruhan gambar itu dapat dijumlahkan dengan beberapa operasi bilangan bulat per piksel. Berikut contoh gambar yang dapat dijadikan perhitungan *integral image* :



Gambar 2. 14 Contoh Bagian *Haar Feature*

Di bawah ini ada contoh perhitungan *integral image*. Nilai *integral image* dari kolom kesatu baris kesatu tetap, sedangkan *integral image* kolom kedua baris kesatu merupakan jumlah nilai piksel wajah di sebelah kirinya. Kolom kesatu baris kedua memperoleh nilai *integral image* dari jumlah nilai piksel kolom kesatu baris kesatu. Berikut ini perhitungan lengkapnya :

1	2	3
2	3	4
4	5	1

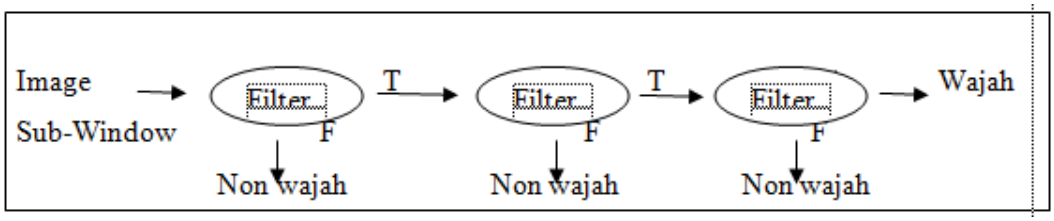
1	1+2	1+2+3
1+2	1+2+2+3	1+2+3+2+3+4
1+2+4	1+2+2+3+4+5	1+2+3+2+3+4+4+5+1

1	3	6
3	8	15
7	17	25

2.2.4.5 Cascade Classifier

Cascade classifier adalah sebuah rantai *stage classifier*, dimana setiap *stage classifier* digunakan untuk mendeteksi apakah didalam image *sub window* terdapat objek yang diinginkan (*object of interest*).

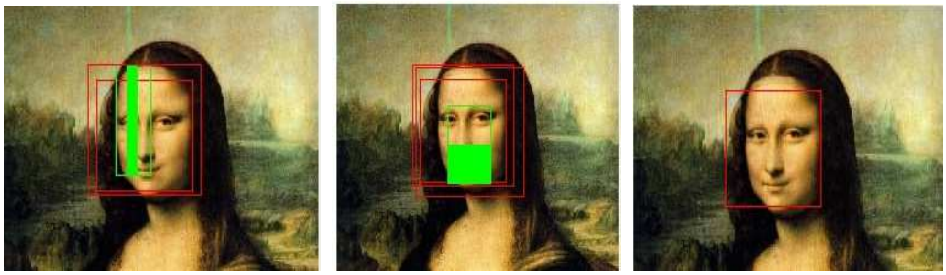
Stage classifier dibangun dengan menggunakan algoritma *adaptive-boost* (*AdaBoost*). Algoritma tersebut mengkombinasikan *performance* banyak *weak classifier* untuk menghasilkan *strong classifier*. *Weak classifier* dalam hal ini adalah nilai dari *haar-like feature*. Jenis *AdaBoost* yang digunakan adalah *Gentle AdaBoost*.



Gambar 2. 15 Proses *Cascade Classifier*

Nilai ambang batas pada setiap filter diatur setiap saat proses filter terjadi, sehingga nilai dari ambang batas bersifat dinamis. Filter pada setiap tingkat telah dilatih untuk mengklasifikasi citra yang telah melalui tahap sebelumnya. Saat proses pengklasifikasian, jika salah satu filter gagal terlewati maka citra dapat dikatakan sebagai daerah yang bukan wajah. Jika citra belum cukup untuk dikatakan sebagai *strong classifier* maka proses diulang sampai bobot terpenuhi dengan menaikkan nilai dari ambang batas. Ketika citra dapat melewati setiap filter yang ada didalam rantai, maka dapat dikatakan daerah tersebut merupakan wajah.

Proses *Cascade Classifier* dapat diilustrasikan pada contoh gambar 2.20 sebagai berikut :



Gambar 2. 16 Contoh Ilustrasi *Cascade Classifier*

2.2.5 *Virtual Reality*

2.2.5.1 Pengertian *Virtual Reality*

Virtual reality (VR) atau realitas maya adalah Suatu lingkungan yang dibangkitkan oleh komputer, bersifat tiga dimensional dan interaktif. Lingkungan-lingkungan ini bisa merupakan model-model dunia nyata ataupun imajiner [10]. Lingkungan *virtual reality* pada saat ini umumnya menyajikan pengalaman visual, yang ditampilkan pada sebuah layar komputer atau melalui sebuah penampil stereokopik, akan tetapi ada beberapa simulasi mengikutsertakan tambahan informasi hasil pengindraan, seperti suara melalui speaker atau headphone.

Bila diteliti lebih jauh, sebenarnya cara kerja virtual reality itu seperti berikut :

a. Pemakai melihat suatu dunia semu, yang sebenarnya berupa gambar-gambar yang bersifat dinamis.
b. Melalui perangkat headphone atau speaker, pemakai dapat mendengar suara yang realistis.

c. Melalui headset, glove, dan walker, semua gerakan pemakai dipantau oleh sistem dan sistem diberikan reaksi yang sesuai sehingga pemakai seolah merasakan sedang berada pada situasi yang nyata, baik secara fisik maupun psikologis.

2.2.5.2 Sejarah *Virtual Reality*

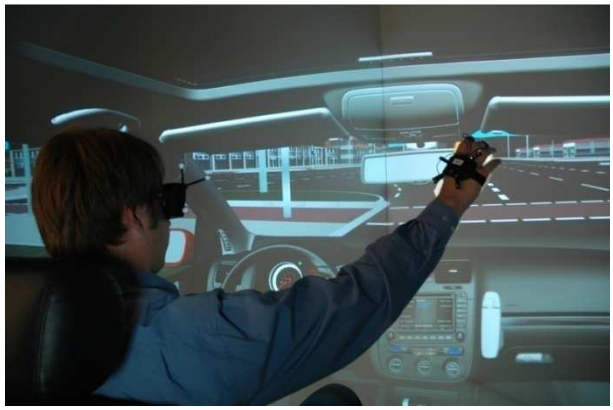
Morton Heilig menulis pada tahun 1950 tentang "Teater Pengalaman" yang dapat meliputi semua indera dengan suatu cara efektif, sehingga menarik penonton ke dalam kegiatan di layar. Kemudian Ia membangun suatu prototipe dari visinya yang dinamakan Sensorama pada 1962, bersama dengan lima film pendek untuk dipertunjukkan didalamnya dengan melibatkan berbagai indera (penglihatan, pendengaran, penciuman, dan sentuhan).

Pada tahun 1968, Ivan Shuterland dengan bantuan dari siswanya bernama Bob Sproull menciptakan apa yang secara luas dianggap sebagai pendahulu dari virtual reality dan sistem. Alat itu primitif, baik dalam kaitan dengan alat penghubung pemakai dan realisme, dan HMD untuk dikenakan oleh pemakai sangatlah berat sehingga harus digantungkan

Teknologi *virtual reality* yang lebih awal adalah Peta Bioskop Aspen, yang diciptakan oleh MIT pada tahun 1977. Programnya adalah suatu simulasi kasar tentang kota Aspen di Colorado. Di sana para pemakai bisa mengembara dalam salah satu dari tiga musim yaitu musim panas, musim dingin, dan musim semi.

Hingga pada saat ini *virtual reality* terus berkembang dalam kehidupan sehari-hari. Contoh aplikasi di bidang *virtual reality* saat ini yaitu manufaktur :

pengujian rancangan, prototipe semu (autocad, 3dmax), analisis ergonomik, simulasi semu dalam perakitan, produksi dan pemeliharaan. Bidang kedua yaitu arsitektur : perancangan gedung. Bidang ketiga yaitu militer : pelatihan (pilot, astronot, pengemudi), Simulasi Peperangan(Cry Engine 3). Bidang ketiga yaitu kedokteran : pelatihan pembedahan, terapi fisik. Bidang keempat yaitu penelitian / pendidikan : studi tentang topan, konfigurasi galaksi, pengujian matematika kompleks. Bidang kelima yaitu dalam hiburan : museum virtual, permainan balap, simulasi pertempuran udara, taman virtual, simulasi ski, dan lain sebagainya.



Gambar 2. 17 Contoh Penerapan *Virtual Reality*[11]

2.2.6 *Augmented Reality*

Menurut Professor Ronald T. Azuma, "*Augmented Reality (AR) is refers to a situation in which the goal is to supplement a user's perception of the real-world through the addition of virtual objects"* [12]. Paul Milgram dan Fumio Kishino pada tahun 1994, mendefinisikan *Milgram's Reality-Virtuality Continum*. Mereka menggambarkan sebuah kontinum yang membentang dari lingkungan nyata untuk lingkungan virtual murni. Mereka menyimpulkan bahwa AR lebih dekat ke lingkungan nyata dan *virtual reality* lebih dekat ke lingkungan virtual.

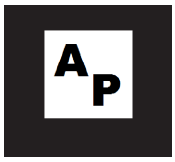
2.2.6.1 Macam-macam Teknik *Augmented Reality*

Augmented Reality memiliki berbagai jenis teknik yang semakin berkembang sampai saat ini. Berikut daftar teknik-teknik di *Augmented Reality* :

2.2.6.1.1 Teknik *Marker Tracking*

Teknik ini menggunakan *marker* untuk mendeteksi posisi tampilnya objek. *Marker* merupakan istilah yang dekat dengan dunia *augmented reality*. *Markers are the black and white squares which enable an application to track and detect the orientation and adjust the position of the 3D object accordingly* [13].

Marker yang dapat dideteksi oleh teknik *marker tracking* ini hanya yang berwarna hitam putih seperti gambar di bawah ini :



Gambar 2. 18 Contoh *Marker*[13]

SDK atau Engine yang dapat digunakan untuk teknik ini ialah :

a. ARToolkit
b. FLARToolkit
c. SLARToolkit

2.2.6.1.2 Teknik *Markerless Tracking*

Teknik ini masih menggunakan *marker* sebagai alat untuk mendeteksi. Namun *marker* yang digunakan dapat berbentuk apa saja tidak harus berwarna hitam putih, misalnya foto atau bahkan benda. SDK atau *Engine* yang dapat digunakan untuk teknik ini ialah:

a. osgART

b. Layar

c. D’Fusion Studio

d. IN2AR

2.2.6.1.3 Teknik GPS *Tracking*

Teknik GPS *Tracking* memanfaatkan lokasi *longitude* dan *latitude* sebagai alat untuk mendeteksi. Biasanya teknik ini digunakan di perangkat lunak untuk *smartphone*. SDK atau *Engine* yang dapat digunakan untuk teknik ini ialah :

a. Layar

b. Junaio

c. Mixare

d. KHARMA

2.2.6.1.4 Teknik *Face Tracking*

Teknik ini menggunakan wajah manusia untuk dideteksi dan pemanfaatannya bisa untuk *face tracking* maupun *face recognition*. SDK atau *Engine* yang dapat digunakan untuk teknik ini ialah:

a. Beyond Reality Face

b. Viewdle

c. SHORE

2.2.6.2 Sejarah *Augmented Reality*

Sejarah tentang *Augmented Reality* (AR) berasal dari *virtual reality* terlebih dahulu hingga Jaron Lanier pada tahun 1992 mengembangkan AR untuk melakukan perbaikan pada pesawat boeing, dan pada tahun yang sama, LB Rosenberg mengembangkan salah satu fungsi sistem AR, yang disebut *Virtual*

Fixtures, yang digunakan di Angkatan Udara AS Armstrong Labs, dan menunjukan manfaatnya pada manusia, dan pada tahun 1992 juga, Steven Feiner, Blair Maclntyre dan dorée Seligmann, memperkenalkan untuk pertama kalinya Major Paper untuk perkembangan Prototype AR. Pada tahun 1999, Hirokazu Kato, mengembangkan ArToolkit di HITLab dan didemonstrasikan di SIGGRAPH, pada tahun 2000, Bruce. H. Thomas, mengembangkan ARQuake, sebuah mobile games AR yang ditunjukan di international symposium on wearable komputers. Pada tahun 2008, witiude AR Travel Guide, memperkenalkan Android G1 telephone yang berteknologi AR, tahun 2009, Saqoosha memperkenalkan FLARToolkit yang merupakan perkembangan dari ArToolkit. FLARToolkit memungkinkan kita memasang teknologi AR di sebuah website, karena output yang dihasikan FLARToolkit berbentuk Flash. Ditahun yang sama, wikitude Drive meluncurkan sistem navigasi berteknologi AR di platform android. Tahun 2010 Acrossair menggunakan teknologi AR pada I-Phone 3GS.

2.2.7 *ActionScript*

Adobe *ActionScript* merupakan bahasa pemrograman yang bekerja di dalam *platform* Adobe Flash. *ActionScript* adalah menunjukkan koleksi set dari *action, function, event,* dan *event handler* yang memungkinkan dikembangkan oleh para *developer* untuk membuat flash *movie* yang lebih komplek dan lebih interaktif. *ActionScript* merupakan bahasa pemrograman berorientasi objek didasarkan pada ECMAScript-standar yang sama yang menjadi dasar JavaScript dan ditujukan untuk berjalan di atas sebuah *runtime environment* (seperti Flash Player, Adobe

AIR, atau Flash Lite). Mirip seperti prinsipnya Java, Flash pada dasarnya berjalan diatas sebuah AVM (*ActionScript Virtual Machine*).

2.2.7.1 Sejarah Perkembangan *ActionScript*

ActionScript sendiri telah memiliki beberapa versi dari satu sampai kini ada versi tiga. *ActionScript2* yaitu versi *ActionScript* yang telah digunakan dalam Flash Player 8 dan sebelumnya, tetap didukung dalam Flash Player 9 dan Flash Player 10. Hingga akhirnya pada tahun 2006 Adobe merilis *ActionScript* versi 3.0, yang merupakan bahasa pemprograman yang bisa didapatkan di Flash CS3 ke atas beda halnya dengan flash 8 dan sebelumnya, yang hanya mendukung *ActionScript2*.

Di dalam Flex *framework,* antarmuka pengguna dibuat menggunakan file MXML, sedangkan *ActionScript3* digunakan hanya untuk menciptakan file akhir SWF. Oleh karena itu *ActionScript 3.0* memiliki beberapa keuntungan dibandingkan versi pendahulunya diantaranya yaitu :

a) Memiliki lebih banyak laporan detail *error* yang ada. Hal tersebut dikarenakan *ActionScript* 3.0 mewajibkan penulisan tipe data dari variabel, *argument*, fungsi, dan yang lainnya. Berbeda dengan *ActionScript*2 yang menjadikan hal tersebut sebagai pilihan. Perubahan tersebut tentu dapat mempermudah *error checking*.

b) Perubahan sintak yang terjadi *ActionScript*3.0 seperti pada nama properti yang di awali *underscores* pada *ActionScript*2.0 dan tidak perlu lagi digunakan pada *ActionScript* 3.0.

c) Tampilan Arsitektur yang baru. Banyak metode sebelumnya untuk menambahkan sesuatu secara dinamis, sekarang memiliki proses yang lebih sederhana dan juga lebih mudah.

2.2.7.2 Dasar-dasar Pemrograman *ActionScript3*

2.2.7.2.1 Penggunaan Perintah *Trace*

Perintah *Trace* digunakan untuk menampilkan informasi pada panel *Output*. Panel *Output* hanya digunakan untuk menguji jalannya program saja, bukan sebagai tampilan utama program. Yang menjadi tampilan utama program adalah *Stage*, misalnya:

```
public function Main():void {
if (stage) init();
else addEventListener(Event.ADDED_TO_STAGE, init);
trace (2 + 5) // akan menampilkan bilangan 7
}
```

2.2.7.2.2 Pendeklarasian Variabel

Sebuah variabel (misalnya variabel x dalam pelajaran matematika yang menyajikan sebuah bilangan yang bisa diberi dengan suatu nilai) adalah sebuah penampung informasi yang bisa diambil atau pun diganti.

Variabel yang akan digunakan dalam *ActionScript* harus dideklarasikan terlebih dahulu. Beberapa tipe data/tipe variabel yang sering digunakan antara lain: *TextField, Number, String, Array, Sprite, Shape,* dan *MovieClip*. Perintah deklarasi variabel mempunyai bentuk umum:

```
var namaVariabel:TipeVariabel;
```

atau cara kedua berikut juga memberikan hasil yang sama:

```
var namaVariabel:TipeVariabel = new TipeVariabel();
```

Namun nama variabel harus memenuhi aturan-aturan berikut:

a. Hanya terdiri dari huruf, angka, karkater garis bawah, dan simbol $.
b. Tidak diawali dengan angka.
c. Tidak sama dengan kata khusus yang sudah dipakai oleh *ActionScript* (*reserved word*), seperti var, Sprite, dan lain-lain.
d. Harus unik

Untuk penulisan dalam penamaan variabel, dapat menggunakan mode KataKapital (*CamelCase*) yang merupakan proses penulisan beberapa kata tanpa tanda spasi namun dengan mengkapitalisasi (menulis dengan huruf besar) setiap kata pembentuknya, misalnya textSaya, radiusBola, dan tinggiAwalStage. Untuk tipe data kompleks (*movie Clip, button, textField*) yang ada dalam *stage* yang dibuat dengan Adobe Flash CS3 Profesional, anda dapat mendeklarasikannya dengan menuliskannya pada nama instan (*instance name*) pada bagian properti objeknya. Flash CS3 akan membuat variabel dengan nama sama dengan nama instan tersebut.

2.2.7.2.3 Tipe-tipe Data

ActionScript 3.0 mempunyai beberapa tipe data primitif seperti yang diberikan dalam Tabel 2.1. Selain itu *ActionScript* juga mempunyai beberapa tipe data kompleks yang dibangun dari tipe data primitif seperti yang diberikan dalam Tabel 2.2. Bahkan kita bisa membangun tipe data sendiri, dalam bahasa pemrograman objek sering dikenal dengan nama *class* objek.

Tabel 2. 1 Tipe Data Primitif

Tipe Data	Pengertian	Contoh
String	Sebuah karakter atau sederet karakter	var textSaya:String =”Hello World”
Boolean	Hanya bernilai benar (true) atau salah (false)	var sudahSelesai:Boolean = true
Number	Bilangan riil positif dan negative	var percepatan:Number = 0.01
Int	Bilangan bulat positif dan negatif (-2.147.483.648 s.d. 2.147.483.647)	var radiusBola:int = 10
Uint	Bilangan bulat positif (0 s.d. 4.294.967.295)	var warnaBola: uint = 0xFF00FF
*	Tipe data sembarang (biasanya digunakan untuk menampung data yang belum diketahui tipenya)	var dataBaru: * = dataSembarang

Tabel 2. 2 Tipe Data Kompleks

Tipe Data	Pengertian	Contoh
MovieClip	Sebuah simbol movie clip	var tokohSaya:MovieClip = new MovieClip()
Sprite	Sebuah Sprite	var kotak: Sprite = new Sprite()
TextField	Sebuah field teks yang bertipe dynamic atau field teks yang bertipe input (secara default akan bertipe dynamic)	var tulisan:TextField =new TextField() tulisan.text=”Tulisan Dinamis”
SimpleButton	Sebuah simbol button (tombol)	var start:SimpleButton=new SimpleButton()
Date	Data dalam format tanggal	var ultah:Date= new Date()
Array	Larik atau array yang dapat memuat lebih dari satu nilai	var arrayBilangan: Array = new Array() arrayBilangan = [2, 3, 15]

Tipe Data	Pengertian	Contoh
Objek	Struktur dasar dari setiap entitas ActionScript. Objek juga merupakan suatu bentuk variabel yang bisa diatur sendiri untuk menyimpan berbagai nilai sebagai alternatif dari Array	var gambar:Objek: new Object gambar.nama =”foto action” gambar.panjang=600

2.2.8 Adobe Flash Professional

Adobe Flash (dahulu bernama Macromedia Flash) adalah salah satu perangkat lunak komputer yang merupakan produk unggulan Adobe Systems. Adobe Flash digunakan untuk membuat gambar vektor maupun animasi gambar tersebut. Berkas yang dihasilkan dari perangkat lunak ini mempunyai file extension .swf dan dapat diputar di penjelajah web yang telah dipasangi Adobe Flash Player. Flash menggunakan bahasa pemrograman bernama *ActionScript* yang muncul pertama kalinya pada Flash 5. Sebelum tahun 2005, Flash dirilis oleh Macromedia. Flash 1.0 diluncurkan pada tahun 1996 setelah Macromedia membeli program animasi vektor bernama FutureSplash. Versi terakhir yang diluncurkan di pasaran dengan menggunakan nama 'Macromedia' adalah adalah Macromedia Flash 8. Pada tanggal 3 Desember 2005 Adobe Systems mengakuisisi Macromedia dan seluruh produknya, sehingga nama Macromedia Flash berubah menjadi Adobe Flash.

Adobe Flash merupakan sebuah program yang didesain khusus oleh Adobe dan program aplikasi standar authoring tool professional yang digunakan untuk membuat animasi dan bitmap yang sangat menarik untuk keperluan pembangunan situs web yang interaktif dan dinamis. Flash didesain dengan kemampuan untuk

membuat animasi 2 dimensi yang handal dan ringan sehingga flash banyak digunakan untuk membangun dan memberikan efek animasi pada website, CD Interaktif dan yang lainnya. Selain itu aplikasi ini juga dapat digunakan untuk membuat animasi logo, movie, game, pembuatan navigasi pada situs web, tombol animasi, banner, menu interaktif, interaktif form isian, e-card, screen saver dan pembuatan aplikasi-aplikasi web lainnya. Dalam Flash, terdapat teknik-teknik membuat animasi, fasilitas action script, filter, custom easing dan dapat memasukkan video lengkap dengan fasilitas playback FLV. Keunggulan yang dimiliki oleh Flash ini adalah ia mampu diberikan sedikit code pemograman baik yang berjalan sendiri untuk mengatur animasi yang ada didalamnya atau digunakan untuk berkomunikasi dengan program lain seperti HTML, PHP, dan Database dengan pendekatan XML, dapat dikolaborasikan dengan web, karena mempunyai keunggulan antara lain kecil dalam ukuran file outputnya.

Movie Flash memiliki ukuran file yang kecil dan dapat ditampilkan dengan ukuran layar yang dapat disesuaikan dengan keingginan. Aplikasi Flash merupakan sebuah standar aplikasi industri perancangan animasi web dengan peningkatan pengaturan dan perluasan kemampuan integrasi yang lebih baik. Banyak fiture-fiture baru dalam Flash yang dapat meningkatkan kreativitas dalam pembuatan isi media yang kaya dengan memanfaatkan kemampuan aplikasi tersebut secara maksimal. Fitur-fitur baru ini membantu kita lebih memusatkan perhatian pada desain yang dibuat secara cepat, bukannya memusatkan pada cara kerja dan penggunaan aplikasi tersebut. Flash juga dapat digunakan untuk mengembangkan secara cepat aplikasi-aplikasi web yang kaya dengan pembuatan

script tingkat lanjut. Di dalam aplikasinya juga tersedia sebuah alat untuk men-debug script. Dengan menggunakan *Code hint* untuk mempermudah dan mempercepat pembuatan dan pengembangan isi *ActionScript* secara otomatis.

2.2.9 *Library* Marilena

Library Marilena merupakan library yang mendukung deteksi wajah di flash dengan menggunakan *ActionScript*. Marilena merupakan *library* yang telah berjalan baik untuk mendeteksi wajah atau benda lain yang berdasarkan algoritma *haar feature*. Hanya saja kekurangan dari *haar feature* yang digunakan ialah tidak dapat mendeteksi wajah pada posisi miring atau selain posisi tegak lurus. Sebenarnya Marilena pada dasarnya berasal dari *library* OpenCV yang digunakan pada bahasa C, yang di konversi ke *ActionScript*. Seorang pria berkebangsaan Jepang bernama Ohtsuka Masakazu yang telah menciptakan Marilena. Baru-baru ini seorang pria bernama Mario Klingemann dari Quasimondo.com telah membuat beberapa perubahan untuk mengoptimasikan Marilena versi aslinya. Untuk mengunduh Marilena dapat menggunakan svn dengan memasukkan alamat http://www.libspark.org/svn/as3/Marilena/trunk/.

2.2.10 StarUML

StarUML adalah software permodelan yang mendukung UML (*UnifiedModeling Language*) [14]. Berdasarkan pada UML version 1.4 telah mendukung notasi UML 2.0 dan juga mendukung pendekatan MDA (*Model Driven Architecture*) dengan dukungan konsep UML. StarUML dapat

memaksimalkan pruduktivitas dan kualitas dari suatu *software project*. Berikut ialah konsep dasar dari StarUml.

2.2.10.1 Model, View, dan Diagram

StarUML membuat perbedaan konseptual yang lebih jelas antara *models*, *views*, dan *diagrams*. *Models* adalah elemen yang memuat informasi untuk model perangkat lunak [15]. *View* adalah suatu ekpresi visual dari informasi di dalam model [15]. Diagram adalah suatu koleksi dari elemen yang memberikan pemikiran *user* didalam mendesain secara spesifik [15].

2.2.10.2 *Project* dan Unit

2.2.10.2.1 *Project*

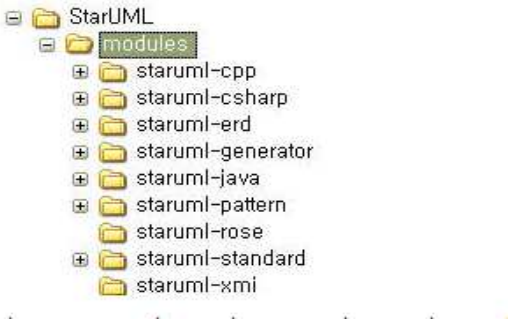
Project adalah unit manajemen dasar didalam StarUML [15]. Suatu *project* dapat mengatur satu atau lebih model perangkat lunak. *Project* merupakan *top-level package* yang selalu ada didalam beberapa model perangkat lunak. Secara umum, satu *project* disimpan dalam satu file.

2.2.10.2.2 File *Project*

File *project* disimpan kedalam format XML dengan extension ".UML" . Semua model,view, dan diagram yang dibuat dengan StarUML disimpan dalam satu file *project*.

2.2.10.2.3 Modul

Modul adalah suatu *package* yang menyediakan fungsi – fungsi baru dan fitur sebagai perluasan dari StarUML [15]. Modul dapat dibuat sebagai kombinasi dari beberapa elemen –elemen perpanjangan dan juga membuat beberapa jenis elemen – elemen didalam suatu modul.



Gambar 2. 19 gambar modul [16]