BAB II

LANDASAN TEORI

2.1 Sejarah Keraton Kasepuhan

Menurut Dinas Pariwisata dan Budaya Provinsi Jawa Barat salah satu museum tertua yang ada di Kota Cirebon adalah Keraton Kasepuhan Cirebon. Kota yang dibangun pada masa perkembangan Islam, pusat kota Cirebon berada di sekitar alun-alun. Di sebelah selatan alun-alun terdapat Keraton Kasepuhan. Secara administratif keraton ini berada di wilayah Kampung Mandalangan, Kelurahan Kasepuhan, Kecamatan Lemah Wungkuk tepatnya pada koordinat 06º 43' 559" Lintang Selatan dan 108º 34' 244" Bujur Timur. Seluruh kompleks keraton luasnya sekitar ± 185.500 m^2 yang dibatasi oleh Jl. Kasepuhan di sebelah utara, timur Jl. Mayor Sastraatmaja, selatan Kali Kriyan, dan di sisi barat terdapat pemukiman penduduk.

Keraton Kasepuhan dibangun sekitar tahun 1529 oleh Pangeran Cakrabuana, merupakan perluasan dari Keraton tertua di Cirebon, Pakungwati. Keraton Pakungwati atau yang dikenal juga Dalem Agung Pakungwati merupakan cikal bakal Keraton Kasepuhan. Keraton Pakungwati yang terletak di sebalah timur Keraton Kasepuhan, dibangun oleh Pangeran Cakrabuana (Putera Raja Padjajaran) pada tahun 1452, berati bersamaan dengan pembangunan Tajug Pejlagrahan yang berada di sebelah timurnya. Pada tahun 1479 keraton ini diperluas dan dilebarkan. Luas situs pertama di Cirebon ini sekitar 4900 m^2, mempunyai tembok keliling sendiri, keadaan

bangunannya sekarang tinggal reruntuhannya saja. Di lokasi tersebut terdapat sisa-sisa bangunan, gua buatan, sumur dan taman.

Pada abad ke-16 Sunan Gunung Jati mangkat, digantikan oleh cicitnya yang bernama Pangeran Emas Zaenal Arifin dan bergelar Panembahan Pakungwati I. Pada tahun 1529 beliau membangun keraton baru di sebelah barat daya keraton lama. Keraton baru ini juga dinamai Keraton Pakungwati, mengabdikan nama puteri Pangeran Cakrabuana atau buyut sultan, yang gugur pada tahun 1549 ketika ikut memadamkan kobaran api yang membakar Mesjid Agung Sang Cipta Rasa.

Pada tahun 1969 Kesultanan Cirebon dibagi dua, yaitu Kesultanan Kanoman dan Kasepuhan. Kesultanan Kanoman dipimpin oleh Pangeran Kartawijaya dan bergelar Sultan Anom I, sementara Kesultanan Kasepuhan dipimpin oleh Pangeran Martawijaya yang bergelar Sultan Sepuh I. Kedua sultan ini kakak beradik, dan masing-masing menempati Keraton sendiri. Sultan Sepuh I menempati Keraton Pakungwati, yang kemudian berganti nama menjadi Keraton Kasepuhan.

Pintu gerbang utama Keraton Kasepuhan terletak di sebelah utara dan pintu gerbang kedua berada di selatan kompleks. Gerbang utara disebut Kreteg Pangrawit berupa jembatan, sedangkan di sebelah selatan disebut lawang sanga (pintu sembilan). Setelah melewati Kreteg (jembatan) Pangrawit akan sampai di bagian depan keraton. Di bagian ini terdapat dua bangunan yaitu Pancaratna dan Pancaniti.

Bangunan Pancaratna berada di kiri depan kompleks arah barat berdenah persegi panjang dengan ukuran 8 x 8 m. Lantai tegel, konstruksi atap ditunjang empat sokoguru di atas lantai yang lebih tinggi dan 12 tiang pendukung di permukaan lantai yang lebih rendah. Atap dari bahan genteng, pada puncaknya terdapat mamolo. Bangunan ini berfungsi sebagai tempat seba atau tempat yang menghadap para pembesar desa atau kampong yang diterima oleh Demang atau Wedana. Secara keseluruhan memiliki pagar terali besi.

Bangunan Pangrawit berada di kiri depan kompleks menghadap arah utara. Bangunan ini berukuran 8 x 8 m, berantai tegel. Bangunan ini terbuka tanpa dinding. Tiang-tiang yang berjumlah 16 buah mendukung atap sirap. Bangunan ini memiliki pagar terali besi. Nama Pancaniti berasal dari panca berarti jalan dan niti berarti mata atau raja atau atasan. Bangunan ini berfungsi sebagai tempat perwira melatih prajurit dalam perang-perangan, tempat istirahat, dan juga sebagai tempat pengadilan. (http://www.disparbud.jabarprov.go.id).

2.2 Pengenalan Objek

Pengenalan objek (*Object Recognition*) adalah menemukan objek pada dunia asli dari sebuah gambar menggunakan model (Ramesh Jain, 1995). Hal ini berfungsi ketika kita akan mengetahui sebuah informasi yang terdapat pada objek tersebut, baik informasi yang berupa teks, suara maupun video. Dalam pengimplementasiannya diperlukan sebuah cara dengan

menyisipkan suatu algoritma khusus yang berfungsi untuk pengenalan pada objek, dalam hal ini menggunakan teknik berbasis fitur. Dan algoritma pengenalan objek ini bergantung pada pencocokan, diantara algoritma yang cukup populer dalam pengenalan objek dan pendekatannya yang berbasis *feature* antara lain seperti *FAST, SURF dan SIFT*.

2.3 Benda Bersejarah

Sejarah sendiri diartikan secara sederhana sebagai ilmu tentang asal usul dan perkembangan peristiwa yang telah terjadi. Menurut Taufik Abdullah sejarah dapat dilihat dalam beberapa sisi, yaitu sejarah dapat digunakan sebagai nasehat, misalnya dengan mengutip kata-kata Sukarno "jangan sekali - sekali melupakan sejarah" ini berarti sejarah adalah sebuah kearifan yang dapat membimbing kita dalam mengarungi hidup saat ini dan merintis hari depan. Sejarah dapat juga dimaknai sebagai "guru" seperti "Sejarah telah mengajarkan pada kita bahwa....". Dalam bidang filsafat, Hegel mengatakan bahwa "sejarah adalah proses ke arah cita kemanusian yang tertinggi".

Berbicara tentang sejarah sangat erat hubungannya dengan benda – benda peninggalan, benda yang dibuat pada zaman dulu dan memiliki cerita sejarah yang tinggi. Benda peninggalan purbakala dijadikan sebagai warisan leluhur merupakan sebuah bukti bahwa sejarah berguna untuk memupuk kepribadian bangsa, baik untuk sekarang maupun masa yang akan datang. Warisan budaya tersebut akan senantiasa menjadi sumber inspirasi daya cipta

kehidupan bangsa, sekaligus menjadi landasan kesadaran nasional dalam pembangunan (Widodo, 1992:32). Oleh karena itu, diperlukan upaya untuk memahami nilai serta budayanya dalam melestarikan benda bersejarah tersebut.

Adapun pengertian budaya (sering juga disebut kebudayaan) adalah keseluruhan sistem gagasan, tindakan, dan hasil karya manusia dalam rangka kehidupan masyarakat yang dijadikan milik diri manusia dengan belajar (Koentjaraningrat, 1984: 9; dan 1986: 180). J.J. Honigmann dalam The World of Man (1959) membedakan adanya tiga gejala kebudayaan, yaitu ideas, activities, dan artifacts. Sejalan dengan hal tersebut Koentjaraningrat mengemukakan bahwa kebudayaan dapat digolongkan dalam tiga wujud. Pertama, wujud kebudayaan sebagai suatu kompleks ide-ide, gagasan, nilai-nilai, norma-norma, dan peraturan. Wujud pertama merupakan wujud ideal dari kebudayaan yang bersifat abstrak (tidak dapat diraba, dipegang, atau difoto), berada di alam pikiran warga masyarakat di mana kebudayaan yang bersangkutan hidup. Kebudayaan ideal ini disebut pula tata kelakukan yang berfungsi mengatur, mengendalikan, dan memberi arah kepada tindakan, kelakuan, dan perbuatan manusia dalam masyarakat.

Para ahli antropologi dan sosiologi menyebut wujud ideal dari kebudayaan ini dengan sistem budaya (*cultural system*) yang dalam bahasa Indonesia dikenal dengan istilah adat atau adat istiadat (dalam bentuk jamak). *Kedua,* wujud kebudayaan sebagai suatu kompleks aktivitas serta tindakan

berpola dari manusia dalam masyarakat. Wujud kebudayaan tersebut dinamakan sistem sosial (*social system*).

Wujud kebudayaan ini dapat diobservasi, difoto, dan didokumentasi karena dalam sistem sosial ini terdapat aktivitas-aktivitas manusia yang berinteraksi satu dengan yang lain. Sistem sosial merupakan perwujudan kebudayaan yang bersifat konkret dalam bentuk perilaku dan bahasa. *Ketiga*, wujud kebudayaan sebagai benda-benda hasil karya manusia; bersifat paling konkret dan berupa benda-benda atau hal-hal yang dapat dilihat, diraba, dan difoto. Wujud kebudayaan yang ketiga ini disebut kebudayaan fisik (*material culture*).

Ketiga wujud kebudayaan dalam realitas kehidupan masyarakat tentu tidak dapat dipisahkan antara satu dengan yang lain. Kebudayaan ideal mengatur dan memberi arah kepada tindakan dan karya manusia. Ide-ide dan tindakan menghasilkan benda-benda yang merupakan kebudayaan fisik. Sebaliknya kebudayaan fisik membentuk lingkungan hidup tertentu yang dapat mempengaruhi tindakan dan cara berpikir masyarakat (Koentjaraningrat, 1984: 5-6; dan 1986: 186-188).

2.4 Pengolahan Citra

Citra adalah gambar yang terdiri dari sekumpulan titik, bidang, garis serta warna sehingga dapat menciptakan suatu objek. Menurut jenisnya, citra terdiri atas citra analog yaitu sesuatu hal yang nampak/ terlihat dan citra digital yaitu sebuah objek yang berdasarkan melalui proses digital. Dengan

berkembangnya teknologi, citra ini banyak dilakukan sebuah inovasi serta pengembangan kearah yang lebih baik, hal tersebut dipengaruhi dari sisi kebermanfaatan dari sebuah citra.

Pengolahan citra (*Image Processing*) adalah pemprosesan sinyal dimana yang menjadi masukan adalah gambar seperti foto atau *frame* dari sebuah *video*. Dengan keluaran atau hasil dari pemprosesannya bisa menjadi sebuah gambar atau sebuah karakteristik atau parameter yang berhubungan dengan gambar tersebut. (Putra D, 2010). Untuk saat ini banyak sekali manfaat daripada pengolahan citra yang sudah dikembangkan, misalnya dalam ilmu kedokteran yaitu penganalisisan hasil scan, bidang fotografi digunakan dalam pengolahan hasil foto dan bahkan dalam bidang militer misalnya digunakan untuk pelacakan yang saat ini pengolahan citra tersebut sudah lebih dikembangkan ke arah computer vision, yaitu dengan menggunakan metode pencocokan yang dapat mendeteksi suatu objek.

2.5 Algoritma

Dalam buku Algoritma dan Pemrograman yang ditulis oleh Rinaldi Munir (2011), Algoritma adalah urutan langkah – langkah untuk memecahkan suatu masalah. Terdapat beberapa definisi lain dari Algoritma, tetapi pada prinsipnya senada dengan definisi yang diungkapkan diatas, yang kita kutip dari berbagai literatur, antara lain; Dalam buku *Introduction to The Design and Analysis of Algorithms,* Algoritma adalah deretan instruksi yang jelas untuk memecahkan masalah, yaitu untuk memperoleh keluaran yang

diinginkan dari suatu masukan dalam jumlah waktu yang terbatas. Serta menurut buku yang ditulis oleh Thomas H Cormen (1989), algoritma adalah prosedur komputasi yang terdefinisi dengan baik yang menggunakan beberapa nilai sebagai masukan dan menghasilkan beberapa nilai yang disebut keluaran. Jadi, algoritma adalah deretan langkah komputasi yang mentransformasikan masukan menjadi keluaran.

Contoh Algoritma:

Diberikan dua buah bilangan bulat tak negative m dan n {m ≥ n}. Algoritma Euclidean mencari pembagi Bersama terbesar, gcd, dari kedua bilangan tersebut, yaitu bilangan bulat posistif terbesar yang habis membagi m dan n.

DESKRIPSI

1. Jika n = 0 maka

   m adalah jawabannya;

   stop.

   tetapi jika n ≠ 0,

   lanjutkan ke langkah 2.
2. Bagilah m dengan n dan misalkan r adalah sisanya.
3. Ganti nilai m dengan *n* dan nilai n dengan nilai *r*, lalu ulang kembali ke langkah 1.

2.6 Augmented Reality

Pada tahun 1997, Azuma menggagaskan adanya penggabungan antara suatu objek nyata dengan objek virtual yang memiliki bentuk 3 Dimensi (3D)

dengan lingkungan luar yang nyata secara real-time yang saat ini dikenal dengan sebutan *Augmented Reality* (AR). Dengan kata lain, *Augmented Reality* adalah teknologi yang mampu menggabungkan objek dua nyata dengan objek maya dalam dua dimensi (2D) atau tiga dimensi (3D), kemudian memproyeksikan objek tersebut secara *real-time*. Perkembangan teknologi ini telah memberikan banyak kontribusi ke dalam berbagai bidang. Bidang – bidang tersebut meliputi periklanan dan pemasaran, arsitektur hiburan, konstruksi, wisata maupun museum.

Dalam buku yang ditulis oleh Anggi Andriyadi (2011), ada beberapa bentuk pengaplikasian dari *Augmented Reality*. Dan Bidang – Bidang yang pernah menerapkan teknologi *Augmented Reality* adalah:

a. Kedokteran *(Medical)*

   Teknologi pencitraan sangat dibutuhkan di dunia kedoteran, seperti misalnya untuk simulasi operasi, simulasi pembuatan vaksin virus, dan lain sebagainya. Untuk itu, bidang kedokteran menerapkan *Augmented Reality* pada visualisasi penelitian mereka.

b. Hiburan *(Entertainment)*

   Augmented Reality sekarang sudah dipakai di dunia entertainment. Bentuknya beragam ada yang dipakai untuk efek perfilman, permainan untk di smartphone, majalah, dan lain sebagainya. Biasanya *Augmented Reality* ini bisa dijadikan sebagai nilai jual yang tinggi di dunia *Entertainment.*

c. Latihan Militer *(Military Training)*

Militer telah menerapkan *Augmented Reality* pada latihan tempur mereka. Sebagai contoh, militer menggunakan *Augmented Reality* untuk membuat sebuah permainan perang, dimana prajurit masuk kedalam dunia *game* tersebut dan seolah – olah seperti melakukan perang sesungguhnya.

d. *Engineering*

Dunia *Augmented Reality* juga telah mencakup dunia *Engineering.* Biasanya *Augmented Reality* digunakan untuk latihan para *Engineering* untuk bereksperimen. Misalnya ahli *Engineering* Mesin menggunakan *Augmented Reality* untuk memperbaiki mobil yang rusak.

e. *Robotics dan Telerobotics*

Dalam bidang robotika, seorang operator robot menggunakan pencitraan visual dalam mengendalikan robot itu. Jadi, penerapan *Augmented Reality* dibutuhkan di dunia robot.

f. *Consumer Design*

Berkembangnya zaman, *Augmented Reality i*ni telah banyak dikembangkan dan digunakan dalam mempromosikan produk. Sebagai contoh seorang pengembang menggunakan brosur dalam bentuk virtual untuk memberikan informasi yang lengkap secara 3D, sehingga pelanggan dapat mengetahui secara jelas produk yang ditawarkan.

2.7 Marker

Didalam buku Augmented Reality with ArToolkit yang ditulis oleh Anggi Andriyadi (2011), Marker adalah sebuah gambar berpola khusus yang sudah dikenali oleh Template Memory ArToolkit. Marker ini berfungsi untuk dibaca dan dikenali oleh kamera lalu dicocokan dengan template ArToolkit. Marker standar yang sering digunakan adalah Marker Hiro dan Marker Kanji. Dapat dilihat pada gambar 2.1.

Gambar 2.1. Contoh Marker Hiro dan Kanji

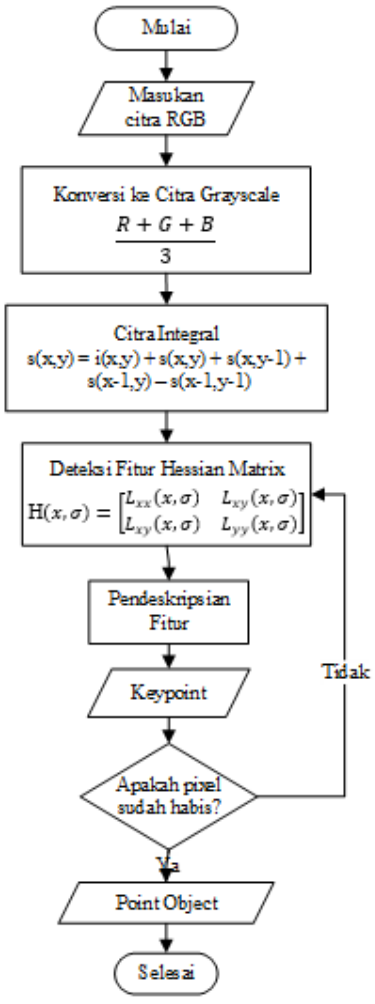
Pola marker tersebut sudah ada di dalam template memory ArToolkit. Pada umumnya, marker yang bisa dikenali ArToolkit hanya marker dengan pola berbentuk kotak dengan bingkai hitam didalamnya, namun seiring banyaknya pengembang AR, pola tersebut sudah jarang digunakan

2.8 Algoritma *Speeded Up Robust Features* (SURF)

SURF (*Speeded Up Robust Features*) adalah pedeteksi fitur lokal yang kuat, Algoritma SURF pertama kali dipublikasikan oleh peneliti dari ETH Zurich, Herbert Bay pada tahun 2006. yang dapat digunakan dalam tugas-tugas visi komputer seperti pengenalan objek atau rekonstruksi 3D.

Dalam pengembangannya Herbert Bay juga dibantu oleh dua rekannya yaitu Tinne Tuytelaars dari Katholieke Universiteit Leuven dan Luc

Van Gool. SURF mampu mendeteksi fitur lokal suatu citra dengan handal dan cepat. Algoritma SURF menggunakan penggabungan algoritma citra integral (*integral image*) dan *blob detection* berdasarkan determinan dari matrik Hessian. Algoritma ini terinspirasi dari *Scale Invariant Feature Transform* (SIFT) yang lebih dulu muncul pada 1999, terutama pada tahap *scale space representation*. Versi standar SURF beberapa kali lebih cepat daripada SIFT dan lebih kuat terhadap transformasi gambar yang berbeda dari SIFT. Adapun alur kerja implementasi algoritma ini dapat dilihat pada gambar 2.2.



Gambar 2.2. *Surf Features Detection Algorithm*

Speeded Up Robust Features adalah algoritma yang paling canggih untuk gambar mosaicking. Ini adalah pendeteksi fitur lokal yang tangguh. Algoritma ini menggunakan respon wavelet 2D Haar dan membuat penggunaan gambar integral yang efisien. Algoritma ini bekerja secara efektif dengan adanya noise dan variasi minor lainnya. Juga algoritma ini adalah skala dan iluminasi invarian. Algoritma ini dapat digunakan untuk aplikasi waktu nyata karena versi standar SURF beberapa kali lebih cepat daripada SIFT.

Dalam implementasinya, algoritma SURF dibagi menjadi beberapa tahapan sebagai berikut:

a. Integral Image

Tahap awal dari algoritma SURF ini adalah mempersiapkan citra masukan. Citra masukan adalah citra dengan format *grayscale*. Pada tahap ini, citra dari hasil kamera direpresentasikan menjadi citra integral yang kemudian akan menghasilkan representasi citra. Dengan rumus:

$$\sum = A + D - (C + B)$$

b. Interest Point Detection

Deteksi titik perhatian (*interest point*) digunakan untuk memilih titik yang mengandung banyak informasi dan sekaligus stabil terhadap gangguan lokal atau global dalam citra digital. Dalam algoritma SURF, dipilih detektor titik perhatian yang mempunyai sifat invarian terhadap skala, yaitu *blob detection*. Blob merupakan area pada citra digital yang memiliki sifat yang konstan atau bervariasi dalam kisaran tertentu. Untuk

melakukan komputasi *blob detection* ini, digunakan determinan dari matriks Hessian (DoH) dari citra pada rumus berikut ini.

$$\mathrm{H}(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix}$$

c. *Feature Description*

Fitur didefinisikan sebagai bagian yang mengandung banyak informasi suatu citra, dan fitur ini digunakan sebagai titik awal untuk algoritma deteksi objek. Tujuan dari proses deteksi fitur ini adalah untuk mendapatkan deskripsi dari fitur-fitur dalam citra yang diamati. Langkah pertama adalah melihat orientasi yang dominan pada titik perhatian yang terdapat dalam citra, kemudian membangun suatu area yang akan diambil nilainya dan mencari fitur korespondensi pada citra pembanding

d. *Feature Matching*

Dalam tahap ini yaitu membandingkan fitur hasil perhitungan proses sebelumnya tetapi hanya bila terdapat perbedaan kontras, yang dideteksi melalui tanda dari *trace* matriks Hessian. Dengan cara ini, biaya komputasi dari algoritma SURF bisa dikatakan sangat minim.

Dan sebagai algoritma pendeteksian dan pendeskripsian fitur dari citra, maka pada dasarnya Algoritma ini dipilih penulis karena memiliki beberapa faktor keunggulan sebagai berikut:

a. Dapat mendeteksi fitur-fitur dengan cepat (diklaim lebih cepat dari SIFT) dengan respresentasi citra integral (*integral image*).

b. Dapat mendeskripsikan fitur-fitur yang terdeteksi secara unik (*distinctive*).
c. Memiliki ketahanan (*Invariant*) terhadap transformasi citra seperti:
    - rotasi citra (*rotation*),
    - perubahan skala (*scaling*),
    - perubahan pencahayaan (*illumination*),
    - dan perubahan sudut pandang (*viewpoint*) yang relatif kecil.
d. Tahan terhadap gangguan (*noise*) dengan intensitas tertentu.

2.9 UML (*Unified Modelling Language*)

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang aakn dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang sama tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat di mengerti oleh banyak orang.

Pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak.

UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks – teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi beroritentasi objek.

Menurut Booch (2005:7) UML adalah Bahasa standar untuk membuat rancangan software. UML biasanya digunakan untuk menggambarkan dan membangun, dokumen artifak dari software – *intensive system.*

Menurut Widodo (2011:10), "Beberapa literatur menyebutkan bahwa UML menyediakan sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa diagram yang digabung, misanya diagram komunikasi, diagram urutan dan diagram pewaktuan digabung menjadi diagram interaksi". Namun demikian model-model itu dapat dikelompokkan berdasarkan sifatnya yaitu statis atau dinamis.

Adapun beberapa jenis jenis UML yang digunakan, diantaranya yaitu (*use case diagram, activity diagram, sequence diagram dan class diagram*).

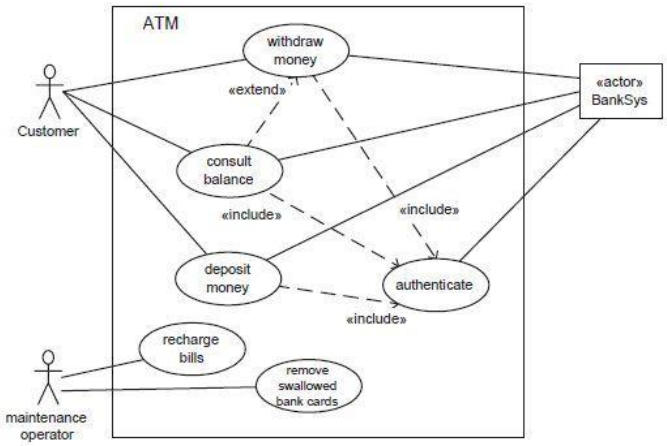
a. Use Case Diagram

Menurut para ahli, pengertian *use case diagram* adalah sebagai berikut:

*"*Diagram untuk menunjukkan peran dari berbagai pengguna dan bagaimana peran-peran menggunakan sistem." –Satzinger, Jackson dan Burd (2009, p242)– *"*Representasi visual yang mewakili interaksi

antara pengguna dan sistem informasi dalam UML.*”* -Shelly dan Rosenblatt (2012, p151)

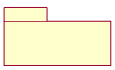
Menurut Widodo (2011:10), Use case diagram yaitu salah satu jenis diagram pada UML yang menggambarkan interaksi antara sistem dan aktor, use case diagram juga dapat men-deskripsikan tipe interaksi antara si pemakai sistem dengan sistemnya. Gambar dapat dilihat pada Gambar 2.3.



Gambar 2.3. *Usecase Diagram*

Dengan penjelasan simbol dapat dilihat pada Tabel 2.1. berikut ini.

Tabel 2.1. *Usecase Diagram*

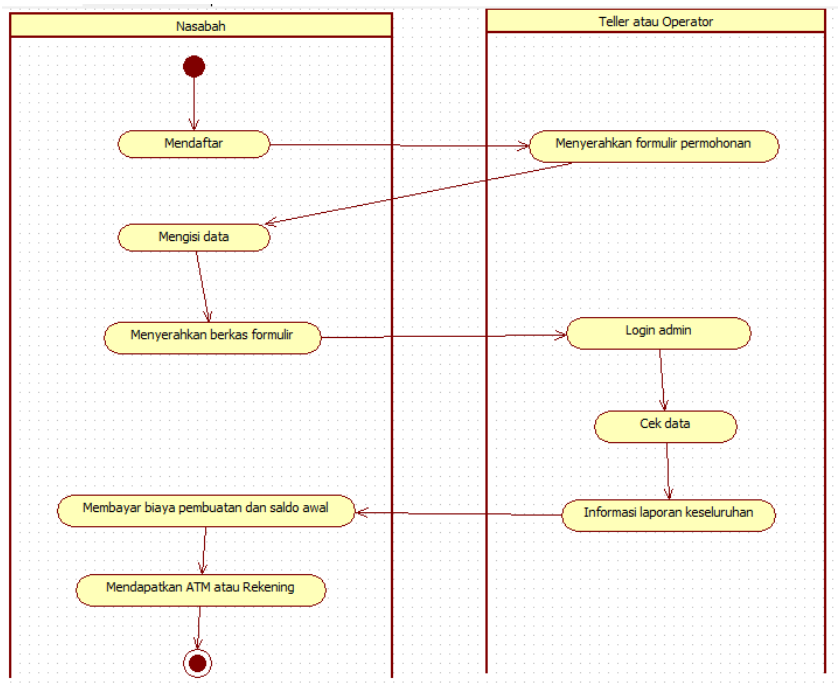
Gambar	**Nama**	**Fungsi**
	Package	Menambahkan paket baru dalam diagram
	Actor	Menambah aktor dalam diagram
	Use case	Menambahkan *use case* pada diagram
	Unidirectional association	Menggambarkan relasi antara aktor dengan *use case*

	Dependencies or Instantiates	Menggambarkan kebergantungan *(dependencies)* antar item dalam diagram
	Generalization	Mengggambarkan relasi lanjut antar *use case* atau menggambarkan struktur pewarisan antar aktor.

b. Activity Diagram

Activity diagram menurut Martin *Fowler (2005: h,163)* adalah "*teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja".* Dalam beberapa hal, *activity* diagram memainkan peran mirip diagram alir, tetapi perbedaan prinsip antara notasi diagram alir adalah *activity* diagram mendukung behavior paralel. *Node* pada sebuah *activity* diagram disebut sebagai action, sehingga diagram tersebut menampilkan sebuah *activity* yang tersusun dari *action.*

Menurut Widodo (2011:10), menyediakan analis dengan kemampuan untuk memodelkan proses dalam suatu sistem informasi. Activity diagram dapat digunakan untuk alur kerja model, use case individual, atau logika keputusan yang terkandung dalam metode individual. Activity diagram juga menyediakan pendekatan untuk proses pemodelan paralel. Pada dasarnya, diagram aktivitas canggih dan merupakan diagram aliran data yang terbaru. Secara teknis, diagram aktivitas menggabungkan ide-ide proses pemodelan dengan teknik yang berbeda termasuk model acara, statecharts. Gambar dapat dilihat pada Gambar 2.4.

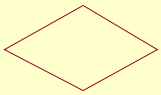


Gambar 2.4 *Activity Diagram*

Dengan penjelasan simbol dapat dilihat pada Tabel 2.2. berikut ini.

Tabel 2.2. Activity Diagram

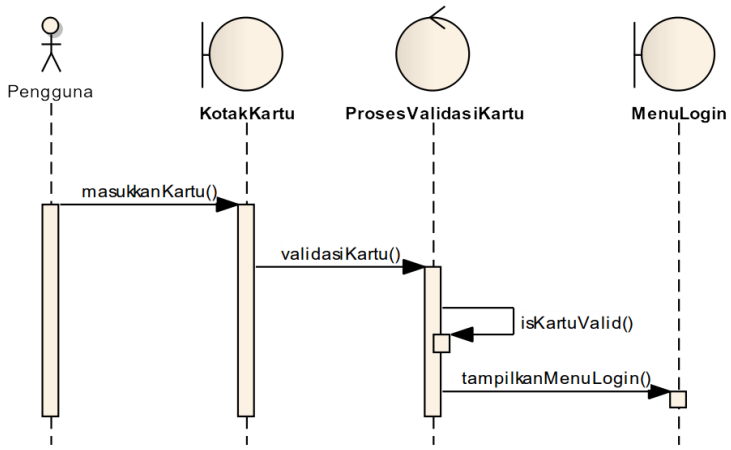
Gambar	Nama	Fungsi
	State	Menambahkan *state* untuk suatu objek
	Activity	Menambahkan aktivitas baru pada diagram
	Start state	Memperlihatkan dimana aliran kerja berawal.
	End state	Memperlihatkan dimana aliran kerja berakhir.
	State transition	Menambah transisi dari suatu aktivitas ke aktivitas yang lainnya.
	Transition to self	Menambah transisi rekursif.

Gambar	Nama	Fungsi
	Horizontal synchronization	Menambahkan sinkronisasi *horizontal* pada diagram.
	Vertical synchronizations	Menambahkan sinkronisasi *vertikal* pada diagram.
	Decisions points	Menambahkan titik keputusan pada aliran kerja.
 **NewSwimlane**	Swimlane	Menambahkan s*wimlane* (sering digunakan pada pemodelan bisnis).

c. Sequence Diagram

Sequence diagram menurut Munawar (2005: h,187) adalah "grafik dua dimensi dimana obyek ditunjukkan dalam dimensi horizontal, sedangkan *lifeline* ditunjukkan dalam dimensi vertikal*"*.

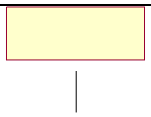
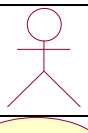
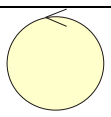
Menurut Widodo *Sequence (2011:10), diagram* menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya sequence diagram adalah gambaran tahap demi tahap yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan use case diagram. Gambar dapat dilihat pada Gambar 2.5.

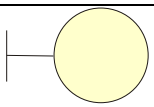
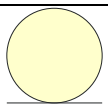


Gambar 2.5. *Sequence Diagram*

Selanjutnya, untuk pengenalan simbol dapat dilihat pada tabel 2.3. berikut.

Tabel 2.3. Sequence Diagram

Gambar	Nama	Fungsi
	Object	Menambahkan objek baru pada Diagram.
	Object message	Menggmbar pesan *(message)* antar dua objek.
	Message to self	Menggambar pesan *(message)* yang menuju dirinya sendiri.
	Return message	Menggambarkan pengembalian dari pemanggilan prosedur.
	Destruction marker	Memperlihatkan saat objek tertentu dihancurkan.
	Actor	Menggambarkan aktor pada diagram kelas
	Use case	Menggambarkan *use case* pada diagram kelas
	Control	Menggambarkan unsur kendali pada diagram.

Gambar	Nama	Fungsi
	Boundary	Menambahkan kelas batasan *(boundary)* pada diagram.
	Entity	Menambahkan kelas entitas *(entity)* pada diagram.
	Domain	Menambahkan domain baru pada diagram.
	Domain package	Menambahkan paket domain baru pada diagram.

d. Class Diagram

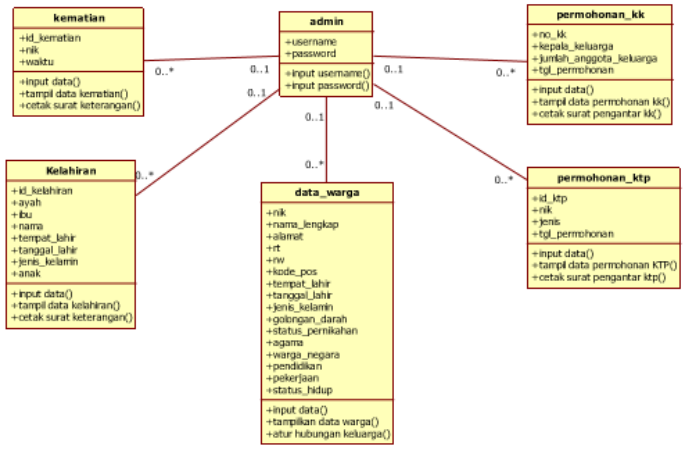
Class diagram menurut Munawar (2005: h, 28) adalah "*merupakan himpunan dari objek-objek yang sejenis".* Sebuah objek memiliki keadaan sesaat (*state*) dan perilaku *(behavior). State* sebuah objek adalah kondisi objek tersebut yang dinyatakan dalam *attribute/ properties*. Sedangkan perilaku suatu objek mendefinisikan bagaimana sebuah objek bertindak/beraksi dan memberikan reaksi. Menurut Widodo (2011:10), Tujuan utama dari class diagram adalah untuk menciptakan sebuah kosa kata yang digunakan oleh analis dan pengguna. Class diagram biasanya merupakan hal-hal, ide-ide atau konsep yang terkandung dalam aplikasi. Misalnya, jika sedang membangun sebuah aplikasi penggajian, ini mungkin akan berisi kelas yang mewakili hal-hal seperti karyawan, cek dan pendaftaran gaji. Class diagram menggambarkan hubungan antara kelas. Gambar dapat dilihat pada Gambar 2.6.

Class diagram memiliki 3 area pokok:

1) *Name*

2) *Attribute*

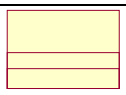
3) *Method*



Gambar 2.6 *Class Diagram*

Dan untuk pengenalan simbol dapat dilihat pada tabel 2.4. berikut ini.

Tabel 2.4. Class Diagram

Gambar	**Nama**	**Fungsi**
	Class	Menambahkan kelas barupada Diagram.
	Interface	Menambahkan kelas antarmuka *(interface)* pada diagram
	Association	Menggambar relasi asosiasi
	Association class	Menghubungkan kelas asosiasi *(association class)* pada suatu relasi asosiasi
	Generalization	Menggambarkan relasi generalisasi
	Realize	Menggambarkan relasi realisasai
	Aggregation	Menggambarkan relasi agregasi

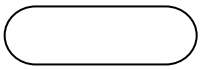
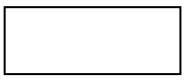
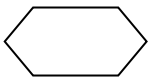
2.10 Flowchart

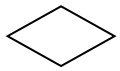
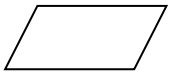
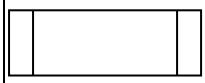
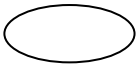
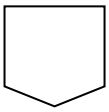
Notasi algoritma yang paling banyak digunakan adalah flowchart, karena flowchart memiliki bentuk yang sederhana dan mudah dipahami. Dalam buku bahasa pemrograman (Suprapto, 2008) Flow chart (diagram alir) adalah penggambaran secara grafik dari langkah-langkah pemecahan masalah yang harus diikuti oleh pemroses. Flow chart terdiri atas sekumpulan simbol, dimana masing-masing symbol menggambarkan suatu kegiatan tertentu.

Dan menurut (Heri Sismoro, 2005), Flowchart terbagi menjadi dua, flowchart sistem yaitu bagan yang menggambarkan suatu prosedur dan proses suatu file dalam suatu media menjadi file dalam media yang lain dalam suatu sistem data. Sedangkan flowchart program yaitu bagan yang menggambarkan urutan logika dari suatu prosedur pemecahan masalah.

Untuk simbol – simbol yang digunakan dapat dilihat pada tabel 2.5.

Tabel 2.5 Simbol-Simbol Dalam Flowchart.

Simbol	Deskripsi
	***Terminator***, simbol untuk menunjukkan awal atau akhir dari aliran proses.
	***Processing,*** menujukan pengolahan aritmatika dan pemindahan data.
	***Preparation,*** memberikan nilai awal pada suatu variabel atau counter.
	Connector, tanda panah yang menunjukkan arah aliran dari satu proses ke proses yang lain.

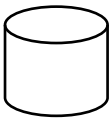
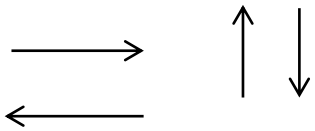
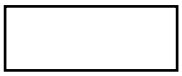
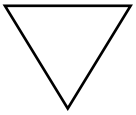
Simbol	Deskripsi
	***Decision***, menunjukan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban ya / tidak.
	***Input/Output,*** simbol untuk menunjukkan data yang menjadi input atau output proses.
	***Sub-process,*** simbol untuk menunjukkan bahwa dalam langkah yang dimaksud terdapat *flowchart* lain yang menggambarkan langkah tersebut lebih rinci.
	***Connector,*** penghubung pada halaman yang sama.
	***Off page connector,*** penghubung pada halaman yang berbeda.

2.11 Flowmap

Menurut Ladjamudin bin Al-Bahra Flowmap adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah- langkah penyelesaian suatu masalah. Flowmap merupakan cara penyajian dari suatu algoritma.

FlowMap mempunyai fungsi sebagai mendefinisikan hubungan antara bagian (pelaku proses), proses manual/berbasis komputer) dan aliran data (dalam bentuk dokumen keluaran dan masukan). Flow Map dapat dikatakan sebuah aliran data berbentuk dokumen atau formulir di dalam suatu sistem informasi yang merupakan suatu aktivitas yang saling terkait dalam hubungannya dengan kebutuhan data dan informasi. Proses aliran dokumen ini dapat terjadi dengan entitas di luar sistem. Berikut dapat dilihat daftar simbol pada Flowmap pada tabel 2.6.

Tabel 2.6 Simbol Flowmap

Simbol	Keterangan
	**Dokumen** input dan output pada proses manual dan proses barbasis komputer.
	**Proses manual** menunjukan proses yang dilakukan secara manual.
	**Penyimpanan magnetik** menunjukan media penyimpanan data/ informasi file pada proses berbasis komputer.
	**Arah alir dokumen** menunjukan arah aliran dokumen antar bagian yang terkait pada suatu sistem keluar ataupun dari luar ke sistem dan antar bagian diluar sistem
	**Penghubung** menunjukan alir dokumen yang terputus atau terpisah pada halaman alir dokumen yang sama
	**Proses Komputer** menunjukan proses yang dilakukan secara komputerisasi
	**Pengarsipan** menunjukan simpanan data non-komputer/informasi file pada proses manual. Dokumen dapat disimpan pada lemari, arsip, map file dan lain-lain
	**Input keyboard** menunjukan input yang dimasukan melalui keyboard
	**Penyimpanan manual** menunjukan media penyimpanan data/informasi secara manual.

2.12 Unity

Teknologi merupakan sebuah alat, alat yang dapat mengubah sesuatu hal yang biasa menjadi luar biasa. Unity 3D adalah teknologi baru yang berusaha membuat hidup lebih baik dan lebih mudah untuk game pengembang. Unity adalah mesin permainan atau sebuah game authoring yang memungkinkan orang – orang kreatif suka untuk membangun sebuah game. Karena dengan menggunakan software Unity, kita dapat membuat sebuah game lebih cepat dan mudah daripada sebelumnya. Selain itu ada beberapa alasan lain yang bermanfaat ketika menggunakan Unity dibanding mesin permainan lainnya, diantaranya:

- Kemampuan untuk memprogram dengan C#, JavaScript, Boo, atau kombinasi dari ini bahasa dalam game yang sama. (Sementara Unity mendukung script dalam dukungan yang berbeda bahasa (C#, JavaScript dan Boo) dalam *game* yang sama, namun dalam praktik terbaik adalah memilih satu bahasa pemrograman dan menggunakannya selama dalam permainan.
- Kemampuan untuk menguji permainan bermain dalam tampilan terpisah (jendela) tanpa harus mem-*build*-nya terlebih dahulu.
- Kemampuan untuk membuat perubahan pada game saat dimainkan dan tidak memiliki perubahan tersebut berdampak pada versi yang disimpan. Hal ini cocok untuk bereksperimen dan menguji.
- Kemampuan untuk mengembangkan dan kemudian menyebarkan ke ponsel/ smartphone (*iOS, Android*, *Windows Phone 8* dan *BlackBerry 10)*,

desktop (Mac, Windows dan *Linux), Web (Safari, Firefox, Chrome* dan *Internet Explorer)* dan *perangkat konsol (Xbox One, Xboxx 360, PlayStation 3, PlayStation 4, PlayStation Vita* dan *Wii U*).

Selain itu, alasan lain menggunakan Unity dalam *game engine* ini adalah sebagian besar bersifat gratis.

2.13 Blender

Blender adalah sebuah software 3D *suite* yang dapat dikatakan salah satu software terlengkap diantara *software – software open source*. *Tools* yang disediakan sederhana, namun sudah mencakup seluruh kebutuhan untuk pembuatan film animasi. Untuk animasi *character* contohnya, blender ini menyediakan fasilitas bone walau tidak secanggih *software – software* kelas komersial seperti Maya atau 3dsMax. Satu kelebihan utama blender adalah *game engine* yang terintegrasi dan dengan game engine tersebut anda dapat menciptakan *software* interaktif baik itu game, presentasi atau web interaktif, tanpa menuntut anda memiliki pengetahuan tentang programming yang mendalam. Bahkan untuk *game* yang sederhana atau presentasi yang sederhana (seperti *walkthrough* interaktif) anda bahkan tidak perlu memerlukan pengetahuan *programming* sama sekali.

Untuk pencahayaan blender menyediakan fasilitas *radiosity*. Dengan *radiosity*, anda dapat menciptakan efek pencahayaan yang realitstik, menyerupai dengan dunia nyata. Walaupun implementasinya pada blender masih terbilang sesderhana dan masih jauh dari sempurna, namun *radiosity*

andalah fasilitas yang absen pada beberapa software animasi komersil bahkan yang bernama besar.

Selain itu blender tersedia untuk berbagai macam OS diantaranya; *Windows, Linux, Max OS X, FreeBSD, Irix* dan *Solaris*. Blender juga tidak menuntut kemampuan komputer yang tinggi, karena pada dasarnya *software* ini berhubungan dengan sebuah gambar maupun yang berhubungan dengan desain grafis.

2.14 Android

Android merupakan salah satu Sistem Operasi dalam smartphone yang popular saat ini. Menurut Nazruddin Safaat H (2014), dalam bukunya Android Pemrograman Aplikasi *Mobile Smartphone* dan Table PC Berbasis Android, pengertian android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Dan android ini juga menyediakan *platform mobile* terbuka, *platform* yang memberikan pengembang untuk melakukan pengembangan sesuai dengan yang diharapkannya.

Telepon pertama yang memakai sistem Android adalah *HTC Dream*, yang dirilis pada 22 Oktober 2008. Pada penghujung 2010 diperkirakan hamper vendor seluler didunia menggunakan Android sebagai operati sistem. Dan adapun versi – versi Android yang pernah dirilis diantaranya sebagai berikut:

a. Android versi 1.1

Pada 9 Maret 2009, Google merilis Android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam, alarm, *voice search*, pengiriman pesan dengan gmail dan pemberitahuan email.

b. Android versi 1.5 *(Cupcake)*

Pada pertengahan Mei 2009, Google kembali merilis telepon selular dengan menggunakan Android dan SDK *(Software Development Kit)* dengan versi 1.5 *(Cupcake)*. Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengupload video ke youtube dan gambar ke picasa langsung dari telepon, dukungan Bluetooth A2DP, kemampuan terhubung secara otomatis ke headset Bluetooth, animasi layar dan keyboard pada layar yang dapat disesuaikan dengan sistem.

c. Android versi 1.6 *(Donut)*

Versi ini rilis pada bulan September dengan menampilkan proses pencarian yang lebih baik dari sebelumnya. Fitur lainnya adalah galeri memunginkan pengguna untuk memilih foto yang akan dihapus; kamera, camcorder dan galeri yang diintegrasikan.

d. Android versi 2.0/ 2.1 (*Éclair*)

Rilis pada 3 Desember 2009, perubahan yang dilakukan adalah pengoptimalan hardware, peningkatan google maps 3.1.2, perubahan UI

dengan browser baru dan dukungan HTML5, daftar kontak yang baru, dukungan flash untuk kamera 3.2 MP, *digital zoom* dan Bluetooth 2.1.

e. Android versi 2.2 (*Froyo: Frozen Yoghurt*)

   Sistem operasi ini diresmikan pada Bulan Mei 2010.

f. Android versi 2.3 (*Gingerbread*)

   Android versi 2.3 diluncurkan pada Desember 2010.

g. Android versi 3.0 (*Honeycomb*)

   Dirilis pada Bulan Februari 2011 sebagai android 3.0 revisi 1 serta android versi 3.0 revision 2 telah dirilis pada Juli 2011.

h. Android versi 3.1

   Dirilis pada Mei 2011, sedangkan Android 3.1 revisi 2 juga dirilis pada Bulan Mei 2011, serta android 3.1 revision 3 dirilis pada Juli 2011.

i. Android versi 3.2

   Dirilis pada Bulan Juli 2011.

j. Android versi 4.0

   Dirilis pada November 2011.

k. Android versi 4.1

l. Android versi 4.2

m. Android versi 4.3

Android versi 3.0 ke atas adalah generasi platform yang digunakan untuk tablet pc. Sementara versi 4.0 sudah merupakan platform yang bisa dipakai di smartphone dan tablet pc. Demikian beberapa ersi android yang sudah dirilis sampai dengan buku ini ditulis, kemungkinan besar versi

tersebut akan terus erkembang seiring dengan kebutuhan yang sangat kompleks dibidang penggunaan smartphone. Dan berdasarkan website resmi (www.developer.android.com) baru – baru ini bulan April- Mei kemarin Android memperkenalkan versi terbarunya Android versi 8.0 - 8.1 Oreo.

2.15 Rational Rose

Rational Rose adalah alat (*tools*) pemodelan visual untuk pengembangan sistem berbasis objek yang sangat handal untuk digunakan sebagai bantuan bagi para pengembang dalam melakukan analisis dan perancangan sistem. Rational Rose digunakan untuk melakukan pemodelan sistem sebelum pengembang menulis kode – kode dalam bahasa pemrograman tertentu. Rational Rose ini juga mendukung pemodelan bisnis, yang membantu para pengembang untuk memahami sistem secara komprehensif. Ia juga membantu analisis sistem dengan cara pengembang membuat diagram use case untuk melihat fungsionalitas sistem secara keseluruhan sesuai dengan harapan dan keinginan pengguna. Kemudian, rational rose juga menuntuk pengembang untuk mengembangkan *Interaction Diagram* untuk melihat bagaimana objek – objek saling bekerja sama dalam menyediakan fungsionalitas yang diperlukan. (Adi Nugroho, 2005)

Dalam Rational Rose, pemodelan adalah cara melihat sistem dari berbagai sudut pandang. Ia mencakup semua diagram yang dikenal dengan UML, aktor – aktor yang terlibat dalam sistem, *use- case*, objek- objek, kelas – kelas, komponen – komponen, serta simpul – simpul penyebaran

(*deployment node*). Model juga mendeskripsikan rincian yang diperlukan sistem dan bagaimana ia akan bekerja, sehingga para pengembang dapat menggunakan model itu sebagai *blueprint* untuk sistem yang akan dikembangkan. Selain itu, Rational Rose juga memungkinkan pengembang untuk mendokumentasikan kegiatan – kegiatannya. Adapun beberapa fungsi dari dokumentasi adalah sebagai berikut:

- Para pengembang dapat menggunakan diagram *use case* untuk mendapatkan pemahaman yang komprehensif tentang sistem dan lingkungan luar yang melingkupi sistem.
- Manager proyek dapat menggunakan diagram *use case* untuk memagi sistem secara keseluruhan menjadi bagian – bagian yang dapat dikelola dengan seksama.
- Analisis sistem dan para pengembang dapat melihat *Sequence Diagram* dan *Collaboration Diagram* untuk memahami bagaimana logika sistem berjalan, objek – objek yang terlibat dalam sistem, serta pesan – pesan (*message*) yang dikirimkan suatu objek ke objek – objek yang lainnya.
- Para penulis teknik dapat melihat diagram *use case* untuk menulis panduan – panduan (*manual*) sistem dan merancang pelatihan – pelatihan.

Selain hal – hal diatas, Rational Rose juga akan membantu para pengembang dengan menghasilkan kode – kode inti dalam beberapa bahasa pemrograman bergantung instalasi *Rational Rose Enterprise Edition*.

Namun, pada dasarnya Rational Rose ini digunakan dalam sebuah pemodelan UML.

2.16 C#

Bahasa pemrograman C# dirancang oleh *Microsoft Corp*. sebagai bahasa pemrograman yang berdaya guna, aman serta mudah digunakan. Sebagai bagian dari platform .NET, bahasa pemrograman C# dirancang untuk bekerja sangat baik di atas framework.NET, yang mampu digunakan untuk menulis perangkat lunak handal demi layanan yang cepat. Bahasa pemrograman C# juga dapat digunakan untuk mengembangkan aplikasi – aplikasi sarana bergerak (*mobile application*), aplikasi berbasis Web (*Web-based applications*), serta aplikasi berskala besar (*Enterprise*) (Andrew Troelsen, 2007). Bahasa pemrograman C# merupakan gabungan dari kecanggihan bahasa keluarga C (C, C++, Objective-C, Java dan sebagainya) dengan kemudahan bahasa pemrograman Visual Basic. Dalam hal ini spesifikasi bahasa pemrograman C# pada awalnya ditulis oleh Anders Hejsberg dan Scoott Wiltamuth dari Microsoft Corp. (Ridi Ferdiana, 2006).

Dengan menggunakan bahasa pemrograman C# kita dapat mengembangkan logika aplikasi yang rumit seperti yang dapat dilakukan menggunakan bahasa keluarga C, namun juga dapat membuat antarmuka pengguna yang ramah terhadap pengguna aplikasi (user friendly) serta dapat mengakses sistem basis data relasional dengan cara yang relative mudah.

Secara mudah bahasa pemrograman C# memiliki karakteristik umum seperti tertulis dibawah ini (Faraz Rasheed, dkk., 2006):

a) Tidak ada alokasi memori secara manual menggunakan pointer (hal ini mirip dengan bahasa pemrograman Java).
b) Manajemen memori otomatis menggunakan salah satu fiturnya yang dinamakan garbage collection (hal ini juga mirip dengan bahasa Java).
c) Mendukung konstruksi kelas, antarmuka, struktur dan enumerasi seperti bahasa pemrograman berorientasi objek lain (misalnya C++ atau Java).
d) Mendukung pemrograman berbasis atribut (*attribut based programming*).
e) Mendukung LINQ (*Language Integrated Query*) yang memungkinkan aplikasi yang ditulis menggunakan bahasa pemrograman C# mampu berinteraksi dan bekerja sama dengan berbagai jenis format data dimana hal ini sangat penting saat kita membuat aplikasi bahasa C# yang mengakses sitem basis data relational (RDBMS – *Relational Database Management System*).
f) Mendukung tipe data dan kelas generic (mirip dengan C++ dan Java).
g) Mendukung operator delegasi (= >)

2.17 Metode Pengujian Perangkat Lunak

Metode pengujian perangkat lunak digunakan untuk menguji program yang dibuat oleh penulis yaitu menggunakan metode *white box testing* dan *black box testing*.

a. Pengujian Kotak Putih (*White-Box Testing)*

Menurut Rogers S. Pressman (2002:551) Pengujian *White Box* (*glass box*) adalah pengujian yang di dasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian.

Tujuan pengujian perangkat lunak adalah:

a. Menilai apakah perangkat lunak yang dikembangkan telah memenuhi kebutuhan pemakai.
b. Menilai apakah tahap pengembangan perangkat lunak telah sesuai dengan metodologi yang digunakan.
c. Membuat dokumentasi hasil pengujian yang menginformasikan kesesuaian perangkat lunak yang diuji dengan spesifikasi yang telah ditentukan.

Klasifikasi white *box* testing mencakup beberapa pengujian (Janner Simarmata, 2010), yaitu:

1. Pengujian unit. Pengujian ini berada pada tingkat yang sangat dasar seperti ketika unit kode dikembangkan atau fungsi tertentu dibangun. Pengujian unit berkaitan dengan unit secara keseluruhan.
2. Analisis statis dan dinamis. Analisis statis dilibatkan melalui kode untuk mengetahui segala kemungkinan cacat dalam kode, sedangkan

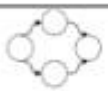
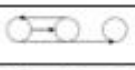
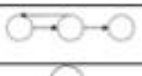
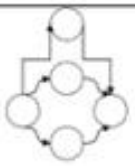
analisis dinamis akan melibatkan pelaksanaan kode dan penganalisisan hasilnya.

3. Cakupan pernyataan. Dalam hal ini jenis pengujian kode dijalankan dengan setiap pernyataan dari aplikasi yang dijalankan minimal sekali. Hal tersebut membantu dalam memastikan semua pernyataan untuk dijalankan tanpa efek samping.
4. Cakupan cabang. Pengujian ini membantu pemvalidasian semua cabang di dalam kode dan memastikan bahwa tidak ada yang mengarah ke percabangan perilaku abnormal dari aplikasi.
5. Pengujian mutasi. Pada pengujian ini aplikasi diuji untuk kode yang telah dimodifikasi setelah pemasangan bug/cacat tertentu. Hal ini juga membantu dalam menemukan kode dan strategi pengodean yang dapat membantu dalam mengembangkan fungsi secara efektif.

Teknik *white box* sebagai berikut:

1. Pengujian basis *path*. Metode ini memungkinkan penguji dapat mengukur kompleksitas logis dari desain procedural dan menggunakannya sebagai pedoman untuk menetapkan himpunan basis dari semua jalur eksekusi.
    a. Notasi Diagram Alir. Notasi yang digunakan untuk menggambarkan jalur eksekusi adalah notasi diagram alir (atau grafik program), yang menggunakan notasi lingkaran (simpul atau *node*) dan anak panah (*link* atau *edge*). Untuk notasi diagram alir dapat dilihat pada tabel 2.7.

Tabel 2.7. Notasi Diagram Alir (Janner Simarmata, 2010)

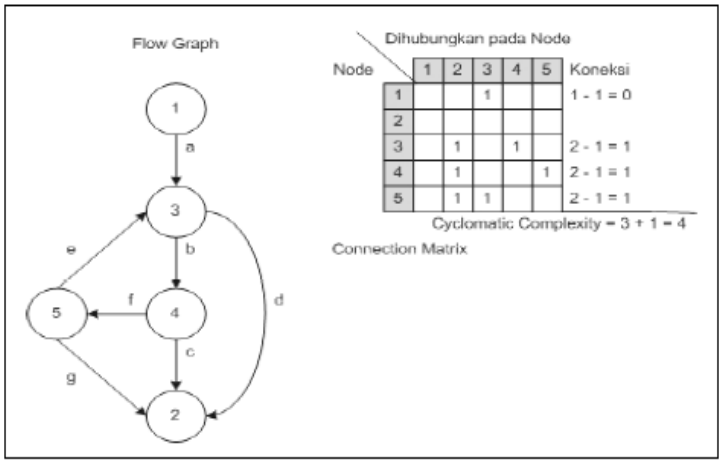
Notasi	Arti
	Skema Sequence
	Skema If
	Skema While (...) DO (...)
	Skema Repeat (...) Until (...)
	Skema Case (...) Of

b. Kompleksitas Siklomatis adalah metrics perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program, nilai yang didapat akan menentukan jumlah jalur independen dalam himpunan path, serta akan memberi nilai batas atas bagi jumlah pengujian yang harus dilakukan, untuk memastikan bahwa semua pernyataan telah dieksekusi sedikitnya satu kali. Untuk menentukan tingkat kompleksitas suatu *methode/ procedure*, digunakan rumus sebagai berikut:

$$V(G) = E – N + 2$$

c. Matriks Grafis (Graph Matrik). Bentuk struktur data yang sering digunakan untuk mengambarkan pengujian adalah dengan matriks grafis. Matriks grafis adalah matriks bujursangkar yang berukuran sama dengan jumlah simpul pada grafik alir. Inputan dalam matriks

harus bersesuaian dengan arah sisi dengan simpul. Bentuk Matriks Grafis dapat dilihat pada gambar 2.7.



Gambar 2.7. Matriks Grafis (Janner Simarmata, 2010)

Rumus untuk menentukan nilai kompleksitas siklomatis:

2. Pengujian Struktur Kontrol. Teknik pengujian basis path merupakan salah satu dari sejumlah teknik untuk pengujian struktur control, namun pengujian basis path tidak memadai untuk beberapa kasus uji.
   a. Pengujian Kondisi. Pengujian kondisi menggunakan kondisi logis sederhana yang terdapat dalam program. Bila suatu kondisi tidak benar, maka akan terdapat paling tidak satu komponen dari kondisi yang salah, sehingga tipe kesalahan pada suatu kondisi meliputi:
      1. Kesalahan operator Boolean
      2. Kesalahan variable Boolean
      3. Kesalahan tanda kurung Boolean
      4. Kesalahan operator relasional
      5. Kesalahan persamaan aritmatika

b. Pengujian Aliran Data. Metode pengujian aliran data melakukan pengujian dengan mengggunakan definisi variable dalam program, efektif digunakan untuk melindungi kesalahan, tetapi akan memiliki cakupan pengukuran dan pemilihan jalur uji yang kompleks.

c. Pengujian *Loop*. *Loop* atau skema pengulangan sering digunakan dalam pembuatan program. Terdapat beberapa macam *loop*, yaitu:

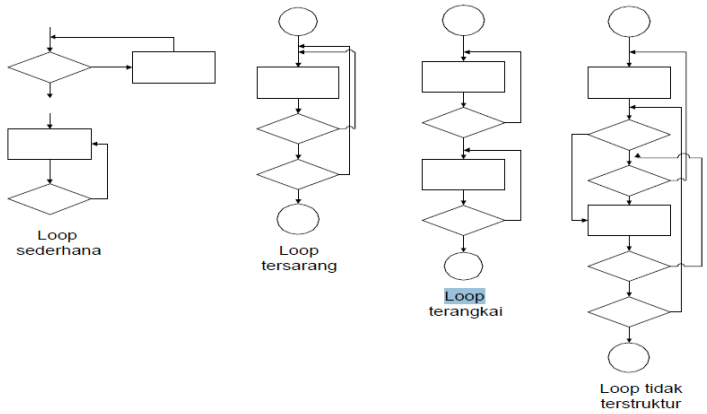
   a. *Loop* Sederhana. Pengujian yang dilakukan harus memperhatikan hal-hal yaitu mengabaikan keseluruhan *loop*, hanya terdapat satu jalur yang melewati *loop*, suatu variable akan melewati loop jika bernilai lebih besar dari nilai yang ditentukan, harus memperhatikan batasan variable pada loop. (kondisi: n-1, n, n+1)

   b. *Loop* Tersarang. Jumlah pengujian akan bertambah secara geometris sesuai jumlah persarangan *loop* yang ada. Untuk menyederhanakan pengujian, maka harus diperhatikan hal-hal berikut yaitu memulai pengujian dari loop terdalam, dengan memulai pengujian dari nilai minimum, melakukan pengujian loop sederhana dari loop terdalam hingga loop terluar dengan memperhatikan parameter yang digunakan.

   c. Loop Terangkai. Pengujian untuk loop terangkai harus disesuaikan dengan independensi variable antara loop tersebut. Jika variable yang digunakan dalam loop kedua tidak

bergantung dengan loop pertama, maka digunakan pengujian loop sederhana, sedangkan bila loop kedua bergantung secara nilai dengan loop kedua, maka lakukan pengujian tersarang.

d. Loop Tidak Terstruktur. Sama sekali tidak dianjurkan untuk digunakan dalam membuat program.



Gambar 2.8. pengujian Loop (Janner Simarmata, 2010)

b. Pengujian Kotak Hitam (*Black-Box Testing*)

Menurut Rogers S.Pressman (2002:551) Pengujian *black box* berfokus pada penyerahan fungsional perangkat lunak dengan demikian pengujian *black box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk satu program. Pengujian *Black Box* merupakan alternatif dari teknik *white box*, tetapi merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalah daripada metode *white box*.

Pengujian *Black Box* berusaha menemukan kesalahan dalam ketegori sebagai berikut:

a. Fungsi-fungsi yang tidak benar atau hilang.
b. Kesalahan Interface.
c. Kesalahan dalam struktur data ekternal.
d. Kesalahan lahan kinerja.
e. Inisialisasi dan kesalahan terminasi.

Pengujian *Black Box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika *internal* perangka lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian *Black Box* merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan.

Adapun klasifikasi Black box testing mencakup beberapa pengujian, yaitu (Janner Simarmata, 2010):

1. Pengujian fungsional (*functional testing*). Pada jenis pengujian ini, perangkat lunak diuji untuk persyaratan fungsional.
2. Pengujian tegangan (*stress testing*). Pengujian ini berkaitan dengan kualitas aplikasi di dalam lingkungan. Ide nya adalah untuk menciptakan sebuah lingkungan yang lebih menuntut aplikasi, tidak seperti saat aplikasi dijalankan pada beban kerja normal.
3. Pengujian beban (*load testing*). Pada pengujian ini, aplikasi akan diuji dengan beban berat atau masukan, seperti yang terjadi pada pengujian

situs web, untuk mengetahui apakah aplikasi/situs gagal atau kinerjanya menurun.

4. Pengujian khusus (*ad hoc testing*). Jenis pengujian ini dilakukan tanpa penciptaan rencana pengujian (test plan) atau kasus pengujian (test case). Pengujian khusus membantu dalam menentukan lingkup dan durasi dari berbaga pengujian lainnya dan juga membantu para penguji dalam mempelajari aplikasi sebelum memulai pengujian dengan pengujian lainnya.
5. Pengujian penyelidikan (*exploratory testing*). Pengujian penyelidikan mirip dengan pengujian khusus dan dilakukan untuk mempelajari atau mencari aplikasi. Pengujian ini merupakan pendekatan yang menyenangkan untuk pengujian.
6. Pengujian usabilitas (*usability testing*). Pengujian ini dilakukan jika antarmuka pengguna dari aplikasinya penting dan harus spesifik untuk jenis pengguna tertentu.
7. Pengujian asap (*smoke testing*). Pengujian ini dilakukan untuk memeriksa apakah aplikasi tersebut sudah siap untuk pengujian yang lebih besar dan bekerja dengan baik tanpa cela sampai tingkay ang paling diharapkan.
8. Pengujian pemulihan (*recovery testing*). Pengujian ini pada dasarnya dilakukan untuk memeriksa seberapa cepat dan baiknya aplikasi bias pulih terhadap semua jenis crash atau kegagalan hardware, masalah bencana, dan lain-lain.

9. Pengujian volume (*volume testing*). Pengujian ini dilakukan terhadap efisiensi dari aplikasi. Jumlah data yang besar diproses melalui aplikasi (yang sedang diuji) untuk memeriksa keterbatasan ekstrem dari sistem.
10. Pengujian domain (*domain testing*). Pengujian ini merupakan penjelasan yang paling sering menjelaskan teknik pengujian.
11. Pengujian scenario (*scenario testing*). Pengujian ini adalah pengujian yang realistis, kredibel dan memotivasi stakeholder, tantangan untuk program dan mempermudah penguji untuk melakukan evaluasi.
12. Pengujian regresi (*regression testing*). Pengujian ini adalah gaya pengujian yang berfokus pada pengujian ulang (*retesting*) setelah ada perubahan.
13. Penerimaan pengguna (*user acceptance*). Pada pengujian ini, perangkat lunak akan diserahkan kepada pengguna untuk mengetahui apakah perangkat lunak memenuhi harapan pengguna dan bekerja seperti yang diharapkan.
14. Pengujian alfa (*alpha testing*). Pada pengujian ini pengguna akan diundang ke pusat pengembangan. Pengguna akan menggunakan aplikasi dan pengembangan mencatat setiap masukan atau tindakan yang dilakukan oleh pengguna.
15. Pengujian beta (*beta testing*). Pada pengujian ini, perangkat lunak didistribusikan sebagi sebuah versi beta dengan pengguna yang menguji aplikasi di situs mereka.