BAB 2
TINJAUAN PUSTAKA

2.1 Bahasa

Bahasa merupakan alat komunikasi yang memegang peranan penting dalam kehidupan manusia. Dalam komunikasi manusia dapat menyampaikan semua pemikiran dan perasaannya kepada orang lain. Tanpa adanya bahasa tidak dapat menyampaikan apa yang terkandung dalam pikiran kepada orang lain, seseorang tidak akan mengetahui apa yang diinginkan orang lain terhadapnya, dan proses komunikasi akan mengalami hambatan.

Dalam pengunaan bahasa sebagai media informasi, ada dua cara yang lazim dilakukan yaitu bahasa lisan dan tulisan. Bahasa lisan lebih mudah dibandingkan dengan bahasa tulisan. Sebab, bahasa lisan dilakukan dengan langsung tatap muka disertai dengan mimik, gerak, lafal, titik, nada disela oleh jeda, diakhiri dengan intonasi dan diikuti oleh kesenyapan yang tidak mungkin adanya perpaduan asimilasi bunyi [6]. Hal itu berbeda dengan bahasa tulis hanya dapat dituangkan dalam bentuk tulisan salah satunya seperti surat, yang harus sesuai dengan aturan kaidah pengunaan ejaan.

2.1.1 Ejaan

Ejaan adalah seperangkat aturan tentang cara menuliskan bahasa dengan menggunakan huruf, kata, dan tanda baca sebagai sasarannya [7]. Apabila berbicara tentang ejaan maka tidak luput dari fungsi ejaan. Fungsi ejaan adalah sebagai berikut :

1. Landasan sebagai pembukaan tata bahasa.
2. Landasan pembakuan kosakata dan peristilahan.
3. Alat penyaring masuknya unsur-unsur bahasa lain.

Dalam sejarah panjang bahasa Indonesia telah mengalami beberapa kali pergantian sistem ejaan. Diawali tahun 1901 dengan *Van Ophuijsen* dan terakhir dengan EYD di tahun 1972 hingga sekarang ini.

Ejaan Bahasa Indonesia yang Disempurnakan atau lazim disebut EYD berlaku sejak diresmikan oleh Presiden Soeharto tanggal 16 Agustus 1972. Bersamaan dengan pedoman umum Ejaan Bahasa Indonesia yang Disempurnakan

ditetapkan oleh Menteri Pendidikan dan Kebudayaan tanggal 31 Agustus 1975 dinyatakan dengan resmi berlaku di seluruh Indonesia. Ejaan yang berlaku sekarang disebut Ejaan yang Disempurnakan karena ejaan itu hasil dari penyempurnaan beberapa ejaan yang pernah disusun sebelumnya.

2.2 Kamus Bahasa Dalam *Game*

Kata-kata yang dibutuhkan untuk *game see the word* diambil dari kata-kata yang terdapat dalam kamus besar bahasa Indonesia dan bahasa Inggris. Dari seluruh kata yang diekstraksi dari kamus tidak seluruhnya dapat digunakan untuk *game see the word*. Karena itu dilakukan seleksi kata-kata yang dapat dilakukan dalam *game see the word*. Seleksi kata dilakukan dengan cara eliminasi kata, baik secara manual maupun otomatisasi. jenis kata-kata yang perlu dieliminasi agar daftar kata bahasa indonesia maupun bahasa Inggris yang dihasilkan dapat digunakan dalam *game see the word* adalah :

1. Kata ulang

   Kata ulang perlu dieliminasi karena *game see the word* tidak memiliki representasi tanda baca '-', sehingga kata ulang tidak akan dapat digunakan dalam permainan.

2. Representasi huruf dalam kamus

   Representasi huruf perlu dieliminasi karena mengacu pada salah satu peraturan Scrabble seperti hanya memperbolehkan kata-kata yang memiliki panjang minimal dua huruf, sehingga representasi huruf yang hanya memiliki panjang satu huruf tidak dapat digunakan dalam permainan.

3. Frasa, peribahasa dan bentuk lain yang memiliki dua atau lebih kata

   Bentuk ini perlu dieliminasi karena pada *game see the word* tidak memiliki representasi spasi, sehingga bentuk yang memiliki dua atau lebih kata tidak dapat digunakan.

Ukuran papan dalam *game* ini merupakan salah satu konfigurasi yang dapat dihasilkan dari salah satu ciri kata-kata dalam suatu bahasa, yaitu panjang kata. Hal ini menjadi sangat penting karena kesempatan kata-kata yang cenderung

panjang dapat dipakai dalam permainan menjadi lebih besar apabila ukuran papan yang digunakan dapat memberikan ruang yang cukup bagi kata-kata tersebut.

Untuk menyesuaikan dengan karakteristik kata-kata dalam bahasa Indonesia, akan dilakukan perbandingan rata-rata panjang kata dari kata-kata dalam bahasa Inggris dan kata-kata dalam bahasa Indonesia.

Daftar kata yang digunakan untuk bahasa Indonesia adalah daftar kata yang terdapat dalam kamus bahasa Indonesia yang telah diseleksi yang telah dijelaskan sebelumnya . Untuk bahasa Inggris, daftar kata yang digunakan adalah daftar kata-kata yang dapat digunakan dalam kamus bahasa Inggris standar. Daftar-daftar kata ini kemudian akan diolah untuk mencari rata-rata panjang kata dalam daftar kata yang dipakai untuk *game* ini dalam bahasa Inggris. Rata-rata panjang kata dicari dengan membagi jumlah seluruh huruf yang ada dalam daftar kata dengan jumlah seluruh kata dalam daftar kata tersebut. Dari hasil perbandingan ini akan diputuskan ukuran dari papan game bahasa Indonesia. Rata-rata panjang kata dari daftar kata-kata bahasa Inggris , daftar lema bahasa Indonesia dan daftar sublema bahasa Indonesia ditampilkan dalam tabel 2.1.

Tabel 2.1 Rata-rata panjang kata pada bahasa Inggris dan bahasa Indonesia

Daftar Kata	Rata-rata Panjang Kata
Bahasa Inggris	8,867
Bahasa Indonesia	7,73

2.2.1 Pendistribusian Huruf dalam Daftar Kata Bahasa Inggris

Konfigurasi *tileset* bahasa Inggris adalah konfigurasi *tileset* yang mengikuti distribusi huruf pada *tileset* pada *scrabble* bahasa Inggris standar. Distribusi s*crabble* bahasa Inggris pertama kali disusun oleh Alfred Mosher Butts pada tahun 1931 dalam permainan *Lexico*. Alfred Mosher Butts melakukan analisis frekuensi pada halaman depan koran *New York Times*. Melalui analisis frekuensi huruf dan kata ini, Alfred Mosher Butts merumuskan distribusi huruf dan skema penilaian yang pada akhirnya digunakan oleh permainan *scrabble* hingga saat ini. Distribusi dan skema nilai huruf ini terdiri atas 100 buah *tile* yang

terdiri atas 98 buah *tile* huruf distribusi lengkap dari *Scrabble* bahasa Inggris standar ini dapat dilihat pada tabel 2.2.

Table 2.2 Frekuensi kemunculan Huruf dalam Bahasa Inggris

Huruf	Jumlah *Tiles*	Persentase kemunculan
A	9	9.18
B	2	2.04
C	2	2.04
D	4	4.08
E	12	12.24
F	2	2.04
G	3	3.06
H	4	2.04
I	9	9.18
J	1	1.02
K	1	1.02
L	4	4.08
M	2	2.04
N	6	6.12
O	8	8.16
P	2	2.04
Q	1	1.02
R	6	6.12
S	4	4.08
T	6	6.12
U	4	4.08
V	2	2.04
W	2	2.04
X	1	1.02
Y	2	2.04

Z	1	1.02

Dilihat dari tabel di atas dapat diurutkan huruf-huruf dengan dimulai dari huruf yang mempunyai nilai *tiles* yang besar, yaitu sebagai berikut :

E(12), A(9), I(9), O(8), T(6), R(6), N(6), S(4), L(4), U(4), D(4), G(3), B(2), C(2), M(2), P(2), F(2), H(2), V(2), W(2), Y(2), J(1), K(1), Q(1), X(1), Z(1).

2.2.2 Pendistribusian Huruf dalam Daftar Kata Bahasa Indonesia

Konfigurasi *tileset* ini dibuat berdasarkan distribusi dari huruf dalam kata-kata bahasa Indonesia yang dapat digunakan dalam *game see the word*. Distribusi dihitung dari himpunan kata-kata yang terdapat dalam kamus bahasa Indonesia. Deskripsi distribusi lengkap dari huruf-huruf dalam daftar kata bahasa Indonesia dijelaskan dalam tabel 2.3

Tabel 2.3 Frekuensi kemunculan Huruf dalam Bahasa Indonesia

Huruf	Jumlah Tiles	Persentase Kemunculan
A	13	13.68
B	3	3.16
C	1	1.05
D	2	2.11
E	10	10.53
F	1	1.05
G	5	5.26
H	1	1.05
I	7	7.37
J	1	1.05
K	5	5.26
L	4	4.21
M	6	6.32
N	9	9.47

O	3	3.16
P	3	3.16
Q	1	1.05
R	6	6.32
S	4	4.21
T	4	4.21
U	1	1.05
V	1	1.05
W	1	1.05
X	1	1.05
Y	1	1.05
Z	1	1.05
TOTAL	97	100

Dilihat dari tabel diatas dapat diurutkan huruf-huruf dengan dimulai dari huruf yang mempunyai nilai tiles yang besar, yaitu sebagai berikut :

A(13), E(10), N(9), I(7), M(6), R(6), G(5), K(5), L(4), S(4), T(4), B(3), O(3), P(3), D(2), C(1), F(1), H(1), J(1), Q(1), U(1), V(1), W(1), X(1), Y(1), Z(1).

2.3 Game

Game adalah sebuah sistem formal yang tertutup yang membawa pemain ke dalam konflik yang terstruktur dan menyelesaikan ketidakpastian dengan hasil yang tidak sama [8].

Game harus diikuti secara sukarela, memiliki tujuan, konflik, dan aturan, ada pihak yang menang dan kalah, interaktif, memiliki tantangan, dapat menciptakan suatu nilai internal sendiri, dapat membuat pemain terlibat, dan merupakan sebuah sistem formal yang tertutup. Dia menambahkan bahwa sebuah *game* memiliki masalah untuk diselesaikan, namun tidak semua masalah adalah sebuah *game*. Berdasarkan hal ini serta seluruh kualitas dari sebuah *game*, *game* adalah sebuah aktifitas penyelesaian masalah yang menggunakan sikap bermain sebagai pendekatannya [9].

Sebuah *game* juga tidak hanya diperuntukan untuk kesenangan semata tetapi juga dapat memberikan pengetahuan tambahan buat para pemainnya, sehingga sebuah *game* dapat menambah wawasan untuk pemainnya, misalnya jenis *word game* dimana *game* tersebut memberikan tantangan, kesenangan, dan secara tidak disadari dapat menambah wawasan dalam pengejaan huruf pada suatu kata serta membantu mengingat kosakata baru.

Sebuah *game* memiliki banyak elemen – elemen penting yang menyusunnya. Ada 4 elemen penting yang menyusun sebuah *game* [9]:

1. *Mechanic*

   Mechanic adalah aturan, batasan, dan prosedur yang berlaku pada sebuah *game*. *Mechanic* menentukan tujuan pemain, apa yang dapat dilakukan untuk mencapai tujuan itu dan apa yang tidak dapat dilakukan. Sebuah *game* dapat memiliki *mechanic* yang unik dan mungkin berbeda dengan keadaan di dunia nyata. Semua yang terjadi di dalam sebuah *game* diatur dan dibatasi oleh *mechanic*.

2. *Story*

   Story atau cerita adalah kejadian berurutan yang terjadi di dalam sebuah *game*. Cerita dapat berbentuk linear dan *pre-scripted* atau bercabang dan muncul tiba - tiba. Cerita dalam sebuah *game* akan menunjukkan tujuan pemain, apa yang terjadi selama permainan, dan bagaimana akhir dari *game* tersebut. Sebuah *game* dengan cerita yang baik dapat membuat pemain merasa terlibat dalam kejadian – kejadian di *game* tersebut.

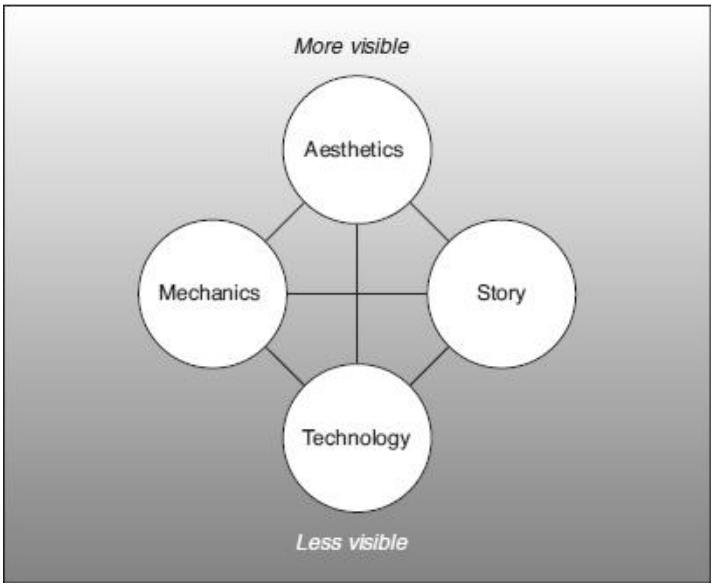
3. *esthetics*

   Aesthetics atau estetika adalah penampilan luar dari sebuah *game*. Estetika dari suatu *game* adalah bagian yang paling terlihat oleh pemain. Estetika suatu *game* meliputi suara, gambar, *video*, dan semua yang dapat langsung terlihat. Penampilan suatu *game* sangat tergantung pada cerita *game* tersebut. Estetika yang baik dapat memperkuat tema dan suasana yang disediakan oleh cerita.

4. *Technology*

   Teknologi adalah seluruh material dan media interaksi yang membentuk suatu *game*. Teknologi sangat beragam dan dapat berupa kertas, dadu, *controller*, CD, dsb. Teknologi yang dipilih untuk sebuah *game* menentukan apa yang dapat dilakukan oleh *game* tersebut dan apa yang tidak.

Keempat elemen *game* ini sama pentingya serta saling mendukung dan mempengaruhi satu sama lain. Cerita menentukan estetika yang akan digunakan serta *mechanic* yang akan diberlakukan. *Mechanic* mempengaruhi pemilihan teknologi yang akan digunakan serta estetika bagaimana *mechanic* itu sendiri akan ditampilkan. Teknologi dapat menentukan sejauh apa estetika sebuah *game* dapat dibuat dan tentunya akan membatasi *mechanic* dari *game*.



Gambar 2.1 Elemen *Game*[9]

Game dapat dikelompokkan menjadi beberapa genre yang berbeda sesuai dengan *game*play dari *game* tersebut. *Game* dibagi dalam beberapa genre [8]:

1. *Action*

   Game dengan genre *action* mengutamakan waktu bereaksi dan koordinasi mata dan tangan. *Action game*s adalah salah satu genre yang memiliki

banyak sub-genre. Contoh *game* yang berasal dari genre *action* antara lain *Grand Theft Auto, Call of Duty*, dan *Dynasty Warrior*.

2. *Strategy*

   Game strategi berfokus pada perencanaan dan taktik serta pengaturan pasukan dan sumber daya. Umumnya memiliki tema penjelajahan dan penguasaan.*Game* strategi dapat dibagi menjadi *Real-Time*, di mana waktu akan senantiasa terus berjalan, dan *Turn-Based*, di mana tiap pemain akan memiliki giliran untuk berpikir dan mengambil keputusan. Contoh *game* dari genre ini adalah *Command And Conquer, Civilization*, dan *Total War*.

4. *Role-Playing*

   Game bergenre *role-playing* berkisar pada membuat dan mengembangkan karakter. Pada umumnya memiliki struktur cerita yang kompleks dan dalam yang dibagi ke dalam berbagai *quest*. Genre ini berakar pada permainan *table top* dengan sistem *paper-based*, salah satunya adalah *Dungeons & Dragons*. Pada *game role-playing*, pemain akan melakukan banyak hal yang berujung pada perkembangan karakter melalui penjelajahan, jual beli barang, serta mengumpulkan *experience* untuk *level up*. Contohnya antara lain *Mass Effect, Deus Ex, Kingdom Hearts*, dan *Final Fantasy*.

5. *Sports*

   Game bergenre olahraga mensimulasikan olahraga yang ada di dunia nyata. Namun, ada pula beberapa judul *game* yang membuat olahraga baru yang tidak ada di dunia nyata. *Game* olahraga mengutamakan pada simulasi yang detail sehingga menyerupai olahraga yang sesungguhnya. Contoh *game* olahraga yang terkenal adalah *Tony Hawk's Pro Skater, Pro Evolution Soccer, FIFA Soccer*, dan *NBA Jam*.

6. *Racing/Driving*

   Game dengan genre *racing/driving* mensimulasikan pengendaraan mobil. Genre ini dapat dibagi menjadi dua jenis, yaitu *arcade style* dan *racing* simulation. *Arcade style* lebih menekankan pada aksi balap mobil itu sendiri. Sedangkan *racing* simulation lebih menekankan simulasi yang

lebih dalam dan nyata. Contoh *game racing arcade style* adalah *Burnout* dan *Mario Kart*. Contoh *racing simulation* antara lain *Gran Turismo* dan *F1 Career Challenge*.

7. *Simulation/Building*

   Game bergenre *simulation/building* berfokus pada pengaturan sumber daya digabungkan dengan pembangunan sesuatu. Meskipun sama – sama menggunakan sumber daya, genre ini berbeda dengan strategi karena yang genre ini berkisar pada pertumbuhan. Genre ini umumnya dibuat berdasarkan pada sistem yang diterapkan pada dunia nyata. Pemain harus mengatur pertumbuhan bisnis atau kotanya dengan baik sambil memperhatikan kondisi keuangan. Contoh *game* bergenre simulation/building adalah *SimCity, Zoo Tycoon,* dan *The Sims.*

8. *Flight and Other Simulation*

   Game dengan genre *simulation* menekankan pada simulasi kegiatan yang dibuat sedekat mungkin dengan kenyataan. Pemain diharapkan dapat menguasai seluruh komponen – komponen yang ada dan dapat mengoperasikan mesin dengan baik. Contoh *game* bergenre ini adalah *Microsoft Flight Simulatior* dan *X-Plane.*

9. *Adventures*

   Adventure adalah genre yang menekankan penjelajahan, pengumpulan, dan pemecahan teka–teki. Pemain umumnya mengendalikan karakter yang memiliki suatu petualangan. *Game adventure* pertama berupa *Text-based Adventure.* Genre *adventure* berpusat pada karakter dan kejadian di sekitarnya. Namun berbeda dengan *role-playing*, perkembangan karakter pada genre ini tidak dipengaruhi oleh pemain. Ciri khas dari genre ini adalah kegiatan *puzzle-solving* yang harus diselesaikan pemain sebelum melanjutkan perjalanan. Contoh *game* bergenre *adventure* adalah *Prince of Persia, Assassin Creed,* dan *Legend of Zelda.*

10. *Edutainment*

    Edutainment adalah sebuah genre yang menggabungkan *game* dengan pembelajaran. Target dari genre ini umumnya anak – anak. Genre ini

berfokus pada pemberian pembelajaran sambil tetap memberikan hiburan. Contoh dari genre ini adalah *Reader Rabbit's Kindergarten, Putt Putt Save the Zoo,* dan *Sierra's Driver's Education.*

11. *Children's game*s

    Genre ini didesain secara khusus untuk anak – anak. *Game play* yang diberikan umumnya sederhana. Terkadang memiliki unsur edukasi, namun tujuan utamanya adalah hiburan.

12. *Casual Game*

    Casual game dibuat dengan tujuan dapat dinikmati oleh siapa saja, pria dan wanita, anak – anak dan orang tua. *Game* dengan genre ini mengurangi kompleksitas, aksi, dan kekerasan agar dapat menarik peminat yang lebih luas. Pada umumnya, *game* dengan genre ini cukup sederhana dan seringkali memasukkan unsur *puzzle* ke dalamnya. Contoh *game* kasual adalah *Tetris*, *Diner Dash*, dan *Plant vs Zombie*.

2.3.1 Word Game

Word Game sering dirancang untuk menguji kemampuan bahasa atau untuk mengeksplorasi sifat-sifatnya. *Word Game* umumnya digunakan sebagai sumber hiburan yang termasuk dalam genre *puzzle*, tetapi telah dibuktikan untuk melayani suatu tujuan pendidikan juga. Misalnya, anak-anak muda bisa menemukan kesenangan bermain *game* sederhana seperti Hangman yang kompetitif, sementara secara alami mengembangkan keterampilan bahasa yang penting seperti *vocabulary* dan ejaan. Memecahkan teka-teki silang, yang membutuhkan pengenalan dengan kosakata yang lebih besar, dan merupakan hobi yang telah lama bahwa orang dewasa juga memainkannya untuk menjaga pikiran mereka tetap tajam. Salah satu contoh *Word Game* yang sangat popular adalah *mole word*. Permainan ini biasanya dilakukan secara sendiri.

2.3.2 Game Balancing

Game Balancing adalah seni menyesuaikan permainan sehingga semua elemen dalam permainan berada dalam hubungan baik dengan satu sama lain [10]. Ini adalah aspek penting karena pemain hanya akan terus bermain permainan jika

itu menantang, menarik, dan adil. *Balancing* merupakan isu utama di semua *game*, termasuk *game* komersial, bahkan kadang-kadang *game* menunjukkan contoh tipikal dari ketidakseimbangan dalam karakter atau unit, yang dapat menjadi tidak menarik dalam permainan. Contoh-contoh ketidakseimbangan terjadi dalam permainan di mana beberapa karakter yang lebih lemah dari yang lain, seperti tokoh-tokoh diplomatik atau medis.

Suatu *game* harus menciptakan keseimbangan secara keseluruhan dalam permainan untuk memastikan bahwa dua pemain memiliki keterampilan yang sama akan memiliki kemungkinan yang sama untuk memenangkan permainan. Hal ini sangat penting untuk membuat permainan yang adil. Setiap item, karakter, sumber daya, dan unit permainan harus berada dalam hubungan yang seimbang satu sama lain, dan biaya manfaat rasio keseluruhan harus konstan atau setidaknya mirip. Jika tidak, beberapa unit akan lebih disukai oleh orang lain, unit tertentu akan menjadi lebih efektif dan berguna. Hal ini menyebabkan pemain hanya memilih pilihan yang sangat spesifik, dan akibatnya, permainan menjadi membosankan.

Oleh karena itu untuk keseimbangan permainan yang tepat, terdapat hal yang perlu dihindari pada skenario berikut :

1. Jika seorang pemain memiliki pilihan antara beberapa alternatif dalam permainan, seperti membangun unit untuk strategi permainan, tapi salah satu dari mereka jauh lebih kuat daripada yang lain, pemain selalu akan memilih yang satu ini. Karena itu, *game* ini membosankan.
2. Jika selama permainan keuntungan kecil dari satu pemain secara besarbesaran mengalami peningkatan (umpan balik positif) dan mengarah ke kemenangan tertentu, permainan muncul tidak adil.
3. Jika selama permainan terdapat kelebihan dari sebuah pemain tidak berpengaruh, permainan akan menjadi tidak menarik.
4. Jika satu sisi lebih kuat dari sisi lain, terdapat kepastian menang atau kalah, *game* akan menjadi tidak menarik.

2.4 Multimedia

Multimedia adalah suatu kombinasi dari teks, grafik, suara, animasi dan *video* yang dimanipulasi secara *digital* dan ditampilkan di computer [11]. Jika pengguna akhir dapat mengontrol elemen multimedia apa dan kapan ditampilkan maka disebut sebagai multimedia interaktif. Multimedia interaktif menjadi *hypermedia* jika terdapat elemen multimedia terstruktur yang saling terhubung dan bisa digunakan oleh pengguna.

Suatu proyek multimedia tidak selalu harus interaktif. Seperti contoh saat pengguna hanya duduk dan menonton film di televisi, Multimedia yang seperti itu bersifat linear atau searah dimulai dari awal dan berjalan secara berurutan sampai terkahir. Sedangkan jika terdapat navigasi yang dapat digunakan dan dikontrol oleh pengguna maka multimedia tersebut bersifat *non-linear*.

Multimedia merupakan media yang isinya menggabungkan beberapa unsur. Multimedia mempunyai 5 unsur, yaitu :

1. *Text*

   Jika sebuah aplikasi multimedia tidak terdapat teks sama sekali, aplikasi mungkin menjadi tidak terlalu kompleks, simbol dan gambar dapat digunakan sebagai penunjuk, selain itu juga dapat menggunakan suara untuk memandu pengguna, namun hal ini tidak membuat menjadi lebih praktis [11]. Butuh usaha lebih dalam pemahaman, lebih mudah bagi pengguna untuk membaca teks secara langsung. Penggunaan teks pada suatu judul (untuk memudahkan pengguna mengetahui dari judul tesebut isi yang dibahas), untuk navigasi (membantu pengguna untuk mencapai sesuatumelalui navigasi tersebut), dan teks digunakan dalam isi (apa yang didapatkan oleh pengguna).

2. *Sound*

   Suara adalah sesuatu yang menghasilkan perbedaan tekanan yang sangat signifikan melalui medium udara hingga dapat menggetarkan indera pendengaran kita [11].

3. *Image*

   Gambar mungkin menjadi unsur paling penting dari proyek multimedia. Secara umum grafik terbagi menjadi 3 grup, yaitu:

   a. -*Colour Graphics*: termasuk di dalamnya grafik *4 bit*-16 warna, *8 bit*-256 warna, *16 bit*-32,768 warna, *24 bit* dan *32 bit* yang berisikan jutaan warna.
   b. *Gray Scale Graphics*: grafik dalam warna hitam dan putih, namun mampu menampilkan macam-macam kedalaman warna abu-abu *(4 bit, 8 bit,* dan *16 bit).*
   c. *Mono Graphics*: grafik dengan warna hitam dah putih saja.

4. *Animation*

   Animasi adalah sekumpulan objek visual yang terus mengalami perubahan per waktu, yang dapat menghasilkan presentasi lebih hidup [11]. Animasi yang paling simpel adalah animasi *2D,* yang sedikit lebih kompleks adalah animasi *21/2D,* dan animasi yang paling menyerupai asli adalah animasi *3D.* Sebuah kata yang berkedip, logo yang berubah-ubah warna, *cel animation*, sebuah tombol yang berubah-ubah adalah contoh sederhana dari animasi *2D*. Pada animasi *21/2D* terdapat *embossing, shadowing, beveling, dan highlighting*. Pada animasi *3D* membuat suatu objek/gambar dapat dilihat dari sisi depan, belakang, samping, atas, dan bawah.

5. *Video*

   Video adalah elemen multimedia yang dapat menarik perhatian dari keramaian di sebuah pameran dagang atau menarik minat siswa dalam proyek pembelajaran berbasis komputer. *Digital video* adalah yang paling menarik dari unsur multimedia, dan merupakan alat yang ampuh untuk membawa pengguna komputer lebih dekat ke dunia nyata. Dengan unsur *video,* aplikasi dapat secara efektif untuk memberikan pesan dan memperkuat cerita, dan pengguna cenderung untuk lebih mempercayai dari apa mereka lihat.

Multimedia sering digunakan dalam dunia hiburan. Selain dari dunia hiburan, Multimedia juga diadopsi oleh dunia *game*. Multimedia dimanfaatkan

juga dalam dunia pendidikan dan bisnis. Di dunia pendidikan, multimedia digunakan sebagai media pengajaran, baik dalam kelas maupun secara sendiri-sendiri. Di dunia bisnis, multimedia digunakan sebagai media profil perusahaan, profil produk, bahkan sebagai media kios informasi dan pelatihan dalam sistem *e-learning*.

2.5 Interaksi Manusia dan Komputer

Interaksi manusia dan komputer merupakan suatu ilmu yang mempelajari desain interaktif antara manusia dengan computer [12]. Pada awal mula pengembangan komputer, desain antarmuka hanya dimengerti oleh mereka yang mengembangkan sistem komputer tersebut. Seiring dengan waktu, penggunaan komputer semakin signifikan jadi diperlukan sebuah ilmu yang mempelajari sekaligus menjembatani perbedaan antara manusia dan komputer.

2.5.1 Antarmuka Pengguna dan Faktor Manusia Terukur

Antarmuka pengguna adalah bagian dari sistem komputer yang memungkinkan manusia berinteraksi dengan komputer. Agar dapat menghasilkan antarmuka dengan kualitas yang tinggi, disukai bahkan ingin ditiru oleh kompetitor, seorang perancang harus memiliki perencanaan yang matang, sensitif terhadap kebutuhan pengguna, taat dalam menganalisa keperluan, rajin testing sambil menjaga budget dan berjalan sesuai dengan jadwal [12]. Untuk memenuhi setiap kebutuhan dan kemauan pengguna, terdapat pedoman bagi perancang selama proses testing. Karena itu dilakukan praktik secara langsung untuk mengevaluasi. Terdapat 5 faktor manusia terukur sebagai pusat evaluasi, yaitu:

1. Waktu untuk belajar : mengukur berapa lama pengguna mempelajari penggunaan aksi yang sehubungan dengan suatu set tugas.
2. Kecepatan performa : mengukur berapa lama proses kinerja suatu tugas dilakukan.
3. Tingkat kesalahan : mengukur berapa banyak kesalahan dan apa jenis kesalahan yang dilakukan oleh pengguna.
4. Daya ingat : mengukur bagaimana pengguna dapat mengingat setelah sekian waktu lamanya.

5. Kepuasan subjektif : mengukur apakah pengguna menyukai berbagai macam aspek antarmuka.

Umumnya setiap perancang ingin sukses dalam setiap faktor pengukuran tersebut, namun untuk beberapa aplikasi tidak semua faktor manusia terukur tersebut dapat dipenuhi.

2.5.2 Prinsip Perancangan Antarmuka Pengguna

2.5.2.1 Mengenal Pengguna

Terdapat 3 jenis pengguna, yaitu:

1. *Novice*: *First-time User*, merupakan pengguna awam, tidak memiliki pengetahuan yang cukup terhadap sistem, hanya mengetahui dan mengoperasikan komputer secara umum dan mendasar.
2. *Knowledgeable*: *Intermittent User*, merupakan pengguna yang sudah lumayan cukup mengenal sistem, mengerti akan aplikasi yang digunakan namun belum dapat menelusuri sistem dan aplikasi secara mendalam.
3. *Expert: Frequent User*, merupakan pengguna yang sudah ahli, mengetahui mengenai sistem dan mampu untuk mengembangkan cara-cara kreatif dalam penggunaan aplikasi.

2.5.2.2 Delapan Aturan Emas Perancangan Antarmuka Pengguna

Dalam merancang antarmuka pengguna, terdapat aturan emas yang diterapkan di hampir semua sistem interaktif [12]. Berikut adalah kedelapan aturan emas tersebut:

1. Berusaha untuk tetap konsisten: terminologi yang identik harus digunakan pada prompts, menu, dan layar *help*. Penggunaan warna, *layout*, kapitalisasi, fonts, dan hal-hal lainnya harus konsisten kecuali pada tampilan untuk konfirmasi perintah menghapus.
2. Menyediakan penggunaan secara universal: menyediakan fitur untuk pemula, juga menyediakan seperti contoh shortcuts untuk pengguna *expert* agar mempermudah dan mempercepat aksi, selain itu juga dapat memperkaya rancangan antarmuka dan meningkatkan kualitas sistem.

3. Menawarkan umpan balik yang informatif: untuk setiap aksi yang dilakukan oleh pengguna, harus ada umpan balik sistem, untuk aksi yang sering dan minor umpan balik sederhana saja sudah cukup.
4. Merancang dialog pada keadaan akhir: urutan aksi dapat diatur menjadi bagian grup yaitu awal, tengah, dan akhir. Terdapat umpan balik pada aksi akhir, bisa juga sebagai indikasi untuk siap ke langkah berikutnya.
5. Mencegah kesalahan dan memberikan penanganan yang sederhana: setiap rancangan yang baik tidak memungkinkan untuk pengguna melakukan kesalahan yang fatal, untuk mencegah kesalahan sistem dibuat sesuai dengan aturannya, seperti contoh untuk kolom alfabet tidak boleh diisi dengan angka. Jika pengguna melakukan kesalahan, sistem harus dapat mendeteksi kesalahan pengguna, dan menawarkan solusi simpel dan intruksi yang spesifik untuk melakukan *recovery*.
6. Mengijinkan pembalikan aksi: sebisa mungkin suatu aksi harus dapat dilakukan pembalikan aksi. Hal ini dapat mengurangi kekhawatiran pengguna dalam melakukan aksi dan mendorong pengguna untuk mengeksplorasi pilihan yang kurang familiar karena pengguna tahu bahwa jika pengguna melakukan kesalahan, maka dapat dilakukan pembalikan aksi.
7. Mendukung pusat kendali *internal*: pengguna yang berpengalaman memiliki keinginan untuk mengendalikan antarmuka secara penuh dan mengharapkan antarmuka untuk merespon aksi yang mereka berikan/lakukan.
8. Mengurangin beban ingatan jangka pendek: kemampuan manusia untuk mengingat sangatlah terbatas. Karena itu harus dihindari keadaan dimana pengguna harus mengingat suatu informasi di suatu layar, dan menggunakan informasi tersebut kembali di layar lainnya.

Prinsip aturan emas ini berfokus pada produktivitas pengguna dengan menyediakan prosedur memasukkan data yang simpel, tampilan yang mudah dimengerti, dan umpan balik yang informatif secara berkala meningkatkan perasaan kompeten, penguasaan dan memegang kontrol penuh terhadap sistem.

2.6 Storyboard

Storyboard adalah sebuah metode yang digunakan dalam menvisualisasikan antar muka pengguna untuk mendapatkan umpan balik dari klien maupun pengguna sebelum mengimplementasi sebuah prototype.

Ada dua jenis pendekatan yang dapat dilakukan dalam membuat *storyboard* [11]. Pendekatan yang pertama adalah *storyboard* yang berupa *outline* grafis yang menjelaskan secara lengkap dan detil setiap layarlayar yang akan dirancang dari penggunaan kata-kata, pemilihan warna dan bayangan, pilihan navigasi, *font*, bentuk tombol, ada tidaknya suara, dan sebagainya. Pendekatan ini biasa cocok dilakukan oleh tim yang dapat dengan cepat merancang prototipenya dan kemudian secara cepat mengembangkannya menjadi *software* yang siap dipakai.

Pendekatan yang kedua yaitu menggunakan *storyboard* yang tidak begitu detil, hanya berupa sketsa-sketsa kasar dan mentah yang dikembangkan lebih lanjut ketika mengerjakan proyeknya secara langsung.

2.7 Bluetooth

Bluetooth adalah sebuah teknologi komunikasi *wireless* (tanpa kabel)yang beroperasi dalam pita frekuensi 2,4 GHz *unlicensed* ISM *(Industrial, Scientific and Medical)* dengan menggunakan sebuah *frequency hopping tranceiver* yang mampu menyediakan layanan komunikasi data dan suara secara *real-time* antara *host-host bluetooth* dengan jarak jangkauan layanan yang terbatas. *Bluetooth* sendiri dapat berupa *card* yang bentuk dan fungsinya hampir sama dengan *card* yang digunakan untuk *wireless local area network (WLAN)* dimana menggunakan frekuensi radio standar IEEE 802.11, hanya saja pada *bluetooth* mempunyai jangkauan jarak layanan yang lebih pendek dan kemampuan *transfer* data yang lebih rendah.

Pada dasarnya *bluetooth* diciptakan bukan hanya menggantikan atau menghilangka penggunaan kabel didalam melakukan pertukaran informasi, tetapi juga mampu menawarkan fitur yang baik untuk teknologi *mobile wireless* dengan biaya yang relat rendah, konsumsi daya yang rendah, *interoperability* yang

menjanjikan, mudah dalam pengoperasian dan mampu menyediakan layanan yang bermacam-macam.

Bluetooth merupakan *chip* radio yang dimasukkan ke dalam komputer, *printer*, ponsel dan sebagainya. *Chip bluetooth* ini dirancang untuk menggantikan kabel. Informasi yang biasanya dibawa oleh kabel dengan *bluetooth* ditrasmisikan pada frekuensi tertentu kemudian diterima oleh *chip bluetooth* kemudian informasi tersebut diterima oleh komputer, ponsel dan sebagainya. Jika kita bisa mentransmisikan data dari komputer ke *printer* mengapa tidak bisa mentransmisikan dari ponsel ke *printer* atau dari *printer* ke *printer* lainnya. Secara lebih rinci, *Bluetooth* merupakan nama yang diberikan untuk teknologi baru dengan menggunakan *short-range radio links* untuk menggantikan koneksi kabel *portable* atau alat elektronik yang sudah pasti. Tujuannya adalah mengurangi kompleksitas, *power* serta biaya. *Bluetooth* diimplementasikan pada tempat-tempat yang tidak mendukung sistem *wireless* seperti di rumah atau di jalan untuk membentuk *Personal Area Networking (PAN)*, yaitu peralatan yang digunakan secara bersama-sama. Ada tiga belas aplikasi spesifik dari *Bluetooth* yaitu :

1. *Generic Access:* prosedur untuk *link management* yang menyediakan jalan untuk membangun dan memelihara *secure link* antara *master* dan *slave.*
2. *Service Discovery: protocol* untuk mengetahui servis yang disediakan.
3. *Serial Port*: penggantian untuk kabel *serial port.*
4. *Generic object exchange*: menetapkan hubungan *client-server* untuk *object movement.*
5. *LAN access: protocol* antara *mobile computer* dan *fixed LAN.*
6. *Dial-up networking*: mengijinkan komputer atau *notebook* untuk *dial/call* via *mobile phone*
7. *Fax:* mengijinkan *mobile fax* untuk berbicara lewat *mobile phone.*
8. *Cordless telephony*: menghubungkan *handset* dengan *local base station*.
9. *Intercom: digital walkie-talkie.*
10. *Headset*: mengijinkan *hands-free voice communication.*
11. *Object push:* menyediakan jalan untuk pertukaran *simple object*s.
12. *File Transfer:* menyediakan fasilitas *transfer file* secara lebih general.

13. *Synchronization:* mengijinkan *PDA* untuk sinkronisasi dengan komputer lain.

2.7.1 Bluetooth on Android SDK

Platform Android mendukung data tanpa kabel pertukaran dengan perangkat *Bluetooth* lain. *Android SDK* juga menyediakan akses ke fungsi *Bluetooth* melalui *Android Bluetooth API* yang memungkinkan aplikasi nirkabel terhubung ke perangkat *Bluetooth* lainnya, memungkinkan *point-to-point* dan *multipoint* fitur nirkabel.

Beberapa kelas dan antarmuka yang diperlukan untuk menciptakan Koneksi *Bluetooth* :

1. *BluetoothAdapter*

   Kelas ini adalah titik masuk bagi semua *Bluetooth* untuk interaksi. Dengan menggunakan kelas ini, hal-hal yang bisa dilakukan adalah:

   a. Temukan perangkat *Bluetooth* lainnya.
   b. *Query* daftar berikut (dipasangkan) perangkat.
   c. Memberi penamaan Perangkat *Bluetooth* menggunakan *MAC dikenal (Media Access Control) address.*
   d. Buat *BluetoothServerSocket* untuk mendengarkan komunikasi dari perangkat audio lain.

2. *BluetoothDevice*

   Dengan menggunakan kelas ini, hal-hal yang bisa dilakukan adalah:

   a. Meminta koneksi dengan perangkat *remote* melalui *BluetoothSocket*
   b. Informasi mengenai perangkat, seperti nama, alamat, kelas, dan *pairing*.

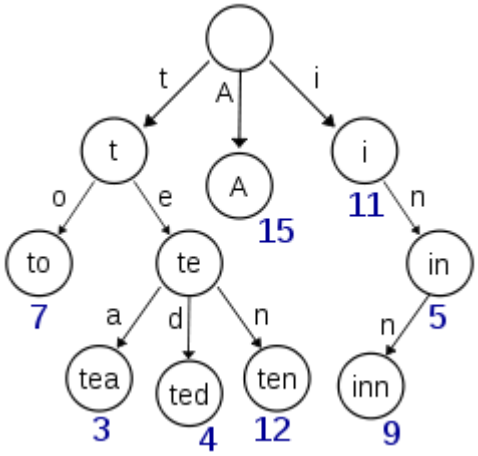
3. *BluetoothSocket*

   Ini merupakan antarmuka untuk *socket Bluetooth*. Dengan menggunakan kelas ini, hal-hal yang bisa dilakukan adalah:

   a. Pertukaran data dengan perangkat *Bluetooth* lain melalui *InputStream* dan *OutputStream.*

2.8 Word Tree

Word Tree adalah sebuah *tree* yang memiliki karakteristik khusus jika digunakan untuk menyimpan unsur-unsur dalam sebuah kamus. *Word Tree* juga memiliki simpul dan penghubung seperti *tree*. Setiap penghubung menjadi penentu arah dari informasi yang disimpan dalam *tree* tersebut. *Tree* dengan penyimpan penghubung terdiri dari blok informasi yang merepresentasikan sebuah penghubung pada sebuah simpul. Blok informasi itu sendiri berbentuk simpul.



Gambar 2.2 Pohon Huruf (*Word Tree*)

Keunggulan *Word Tree* dengan strategi penyimpanan penghubung ini adalah :

1. Penghematan tempat baik pada penyimpanan primer maupun sekunder.
   Penyimpanan dilakukan per penghubung maka model ini hanya menyimpan semua informasi yang berarti pada setiap simpul saja.
2. Kecepatan penghitungan indeks berikutnya.
   Model *Tree* dengan strategi penyimpanan penghubung tidak memerlukan *loop* untuk menghitung alamat berikut. Hal ini menyebabkan model ini akan lebih cepat mengkalkulasi alamat berikut yang harus ditinjau pada saat melakukan traversal.

Struktur data yang dipakai dalam pembangunan kamus *Tree* dengan strategi penyimpan penghubung adalah *linked-list*. Struktur ini berupa *node-node* yang berisi info mengenai penghubung. *Linked-list* ini berlaku bagi sibling dan bagi child pada *Tree* tersebut. Struktur linked-list ini menyebabkan tidak

diperlukannya delimiter pada suatu level sibling. Hal ini karena akhir sibling diketahui jika *next pointer* dari *node sibling* adalah *null.*

Tiap *node* pada *Tree* tersebut mengandung informasi yang menyatakan karakter yang dikandung. Hal ini cukup direpresentasikan dengan menggunakan 5 *bit*. Karena satuan terkecil untuk melakukan manipulasi data adalah 1 *byte* (8 *bit*), maka 3 *bit* sisa digunakan untuk menunjukkan ada tidaknya *child* atau *sibling*, serta untuk menunjukkan valid tidaknya suatu kata yang telah terbentuk dari *traversal* pada *Tree.*

2.8.1 Pencarian Kata Dan Kebutuhan Ruang Memori

Pencarian kata pada struktur t*ree* ini melibatkan beberapa perbandingan. Perbandingan terburuk adalah jika kata tersebut terdapat dalam kamus, merupakan kata yang terpanjang, dan tiap unsur hurufnya terletak pada ujung sibling. Kata yang demikian selain membutuhkan rata-rata perbandingan sesuai dengan panjang kata (mempengaruhi level *child*), juga membutuhkan perbandingan ketika melintasi suatu level sibling.

Perbandingan terbaik untuk kata-kata yang terdapat dalam kamus adalah jika kata tersebut pendek, dan tiap unsur hurufnya merupakan elemen pertama dari suatu level sibling. Perbandingan terbaik juga bisa didapatkan dari kata yang tidak terdapat dalam kamus, karena tidak perlu melakukan traversali pada *tree* tersebut. Secara rata-rata perbandingan yang dibutuhkan untuk pencarian suatu kata adalah sepanjang kata yang dicari tersebut. Kompleksitas pencarian kata adalah O (l), dimana l adalah panjang kata rata-rata.

Ruang memori yang dibutuhkan, [15] jumlah *node* yang terdapat dalam suatu kamus adalah

= [(n*w/2) + (w/2) + (n*w/4)] …….. (2.1)

Keterangan:

n adalah jumlah kata dalam kamus

w adalah panjang kata rata-rata

Sehingga jumlah memori yang dibutuhkan adalah

= X byte * [(n*w/2) + (w/2) + (n*w/4)] …….. (2.2)

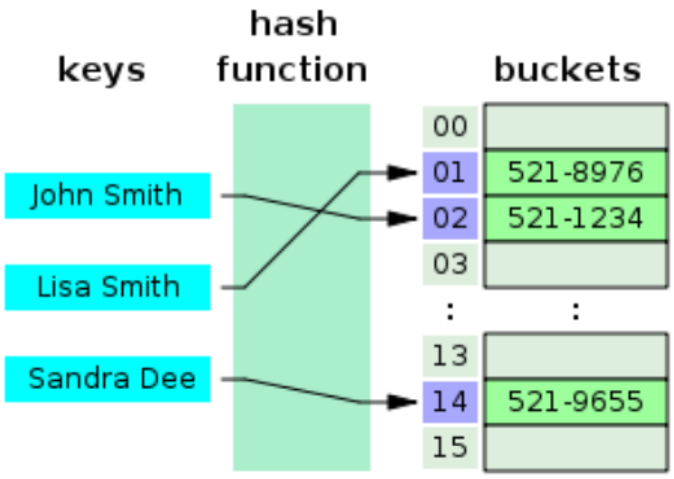
Keterangan:

X terdiri dari : 1 *byte* info, serta 4 *byte* untuk ukuran 2 *pointer* (*child* dan *sibling*).

2.9 Hash Table

Hash Table merupakan salah satu struktur data yang digunakan dalam penyimpanan data sementara. Tujuan dari *hash table* adalah untuk mempercepat pencarian kembali dari banyak data yang disimpan. Hash table menggunakan suatu teknik penyimpanan sehingga waktu yang dibutuhkan untuk penambahan data *(insertions)*, penghapusan data *(deletions)*, dan pencarian data *(searching)* relatif sama dibanding struktur data atau algoritma yang lain.

Hash table adalah data struktur yang memanfaatkan fungsi *hash* dalam mengasosiasikan sebuah *key* dengan sebuah nilai, sehingga *key* menjadi *index* dari data *array* yang disimpan, dalam hal ini *index* berbeda dengan kunci tetapi untuk mencari *index* dihitung dari kunci dengan menggunakan fungsi *hash*. Karena mirip dengan sebuah *array* sehingga *tabel hash* juga dikatakan *assosiative array*. Misalkan terdapan data nama dan nomor telepon, maka tabel *hash* akan menjadi seperti pada gambar 2.3

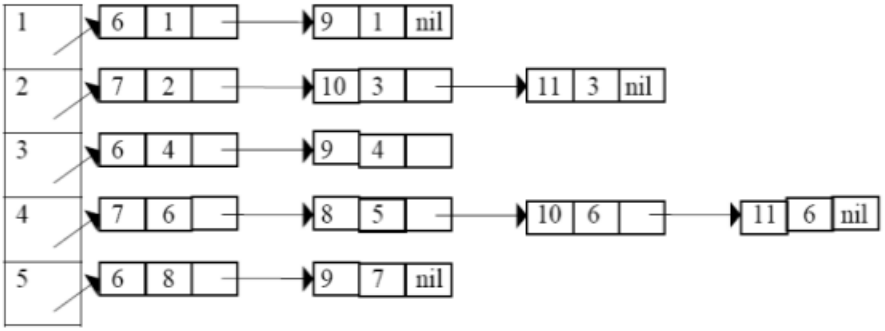


Gambar 2.3 Tabel *Hash*

Sedangakan dalam penyimpanan kamus data yang menjadi index adalah semua *node* mulai dari *root* sampai *leaf* simpul terakhir yang akan diurut mulai dari simpul paling kiri ke simpul palling kanan, Dan huruf-huruf yang terdapat

pada *edge* masing-masing *node* akan disimpan sebagai nilai yang ditunjuk oleh index yang berupa *node* itu sendiri. Setiap huruf disimpan dalam bentuk *integer* A direpresentasikan sebagai 1 dan Z sebagai 26 dengan nilai biner.

Nilai *array* tiap index akan ditempati oleh huruf yang terdapat pada *edge* yang meninggalkan *node* tersebut, *node* pertama yang meninggalkan *node* tersebut akan disimpan terlebih dahulu diikuti *edge* selanjutnya berurutan dari simpul yang paling kiri. Hal ini mungkin terjadi karena setiap lokasi *array* adalah suatu list berantai yang berisi pasangan kunci dan nilai, jika suatu nilai mempunyai kode *hash* yang sama maka niai tersebuat akan menpati list yang sama. hal ini dapat dilihat pada gambar 2.4.



Gambar 2.4 *Array* List Berantai

2.10Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak [27]. *UML* menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan *UML*, dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena *UML* juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa–bahasa berorientasi objek seperti *C++*, *Java*, *C#* atau *VB.NET*. Walaupun demikian, *UML* tetap dapat digunakan untuk modeling aplikasi prosedural dalam *VB* atau *C*. Seperti bahasa-bahasa lainnya, *UML* mendefinisikan notasi dan syntax/semantik.

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

1. *Use case Diagram* untuk memodelkan proses bisnis
2. *Conceptual Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi
3. *Sequence Diagram* untuk memodelkan pengiriman pesan *(message)* antar *object*s
4. *Collaboration Diagram* untuk memodelkan interaksi antar *object*s
5. *State Diagram* untuk memodelkan perilaku *object*s di dalam sistem
6. *Activity Diagram* untuk memodelkan perilaku *Use cases* dan *Objects* di dalam sistem
7. *Class Diagram* untuk memodelkan struktur kelas
8. *Object Diagram* untuk memodelkan struktur *object*
9. *Component Diagram* untuk memodelkan komponen *object*
10. *Deployment Diagram* untuk memodelkan distribusi aplikasi

UML menyediakan 8 macam diagram untuk memodelkan aplikasi berorientasi objek [17], yaitu:

1. *Use case Diagram* untuk menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”.
2. *Sequence Diagram* untuk menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu.
3. *Collaboration Diagram* untuk menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian pesan.
4. *Statechart Diagram* untuk menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima.

5. *Activity Diagrams* menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir.
6. *Class Diagram* untuk menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.
7. *Component Diagram* untuk menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan *(dependency)* diantaranya.
8. *Deployment Diagram* untuk menggambarkan detail bagaimana komponen di*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, *server* atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisikal.

Untuk pengembangan aplikasi permainan ini akan digunakan 4 macam *UML diagram*, yaitu :

1. *Use case Diagram*
2. *Activity Diagram*
3. *Sequence Diagram*
4. *Class Diagram*

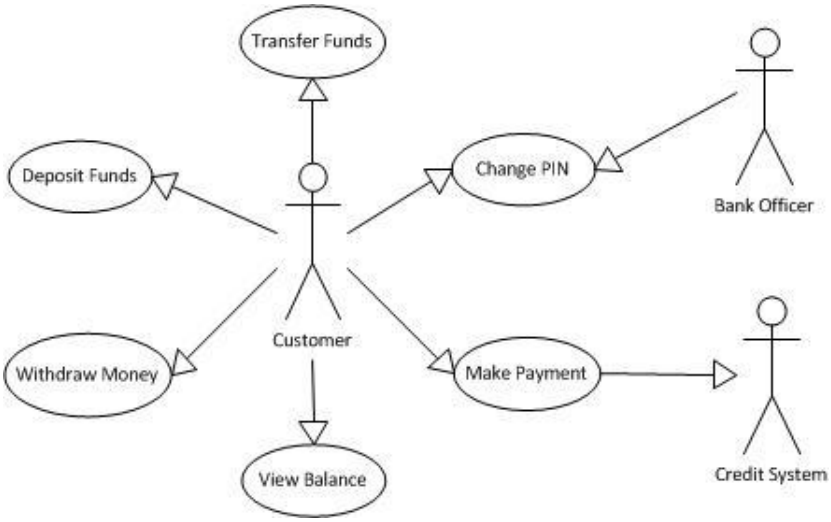
2.10.1 Use case Diagram

Use case mengidentifikasi dan menjelaskan fungsi sistem dari perspektif pengguna luar dan dengan cara dan terminologi yang mereka mengerti. Secara akurat dan menyelesaikan permintaan ini, pengguna tingkat tinggi terlibat yang merupakan ahli subjek materi yang memiliki pengetahuan tentang proses bisnis atau peristiwa.

Gambaran *use case* diwakili oleh sebuah elips horizontal dengan nama muncul kasus penggunaan di atas, bawah, atau di dalam elips. sebuah *use case* mewakili tujuan tunggal dari sistem yang menggambarkan urutan kegiatan dan interaksi pengguna dalam usaha mencapai tujuan. Pembuatan *use case* telah

terbukti menjadi teknik yang sangat baik untuk lebih memahami dan dokumen persyaratan sistem. *Use case* sendiri adalah tidak dianggap sebagai persyaratan fungsional, tetapi cerita (skenario). *Use case* menceritakan terdiri dari satu atau lebih persyaratan [18].

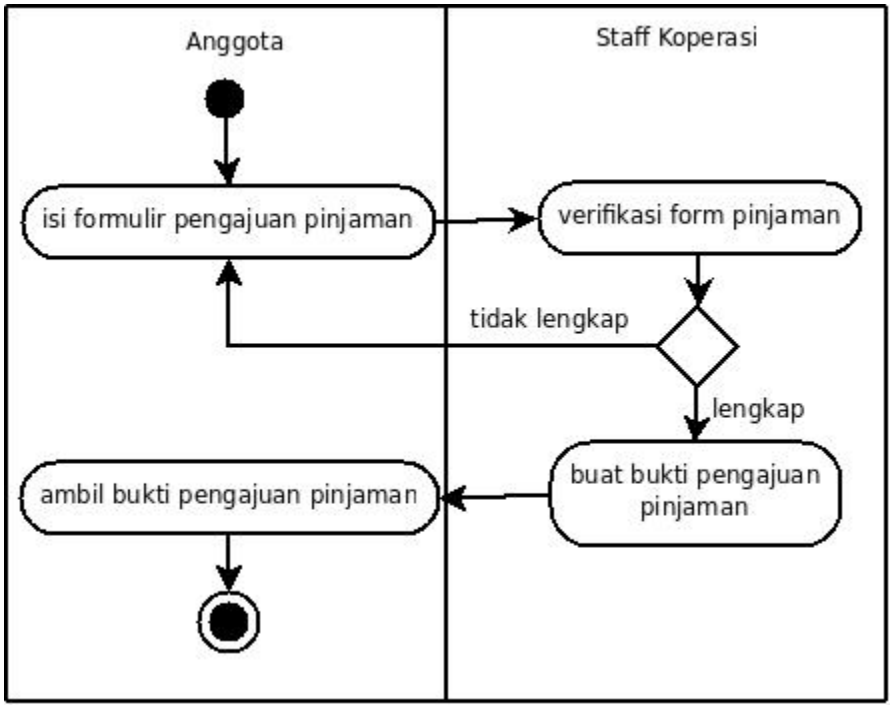
Use case diagram terdiri atas diagram untuk proses dan aktor. Aktor merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi. Proses merepresentasikan operasi-operasi yang dilakukan oleh aktor. Proses digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnya. Aktor yang melakukan operasi dihubungkan dengan garis lurus ke *use case.*



Gambar 2.5 Use Case Diagram

2.10.2 Activity Diagram

Activity diagram, secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas baik proses bisnis atau *use case*. Diagram ini juga dapat digunakan untuk memodelkan aksi yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari aksi tersebut.



Gambar 2.6 Activity Diagram

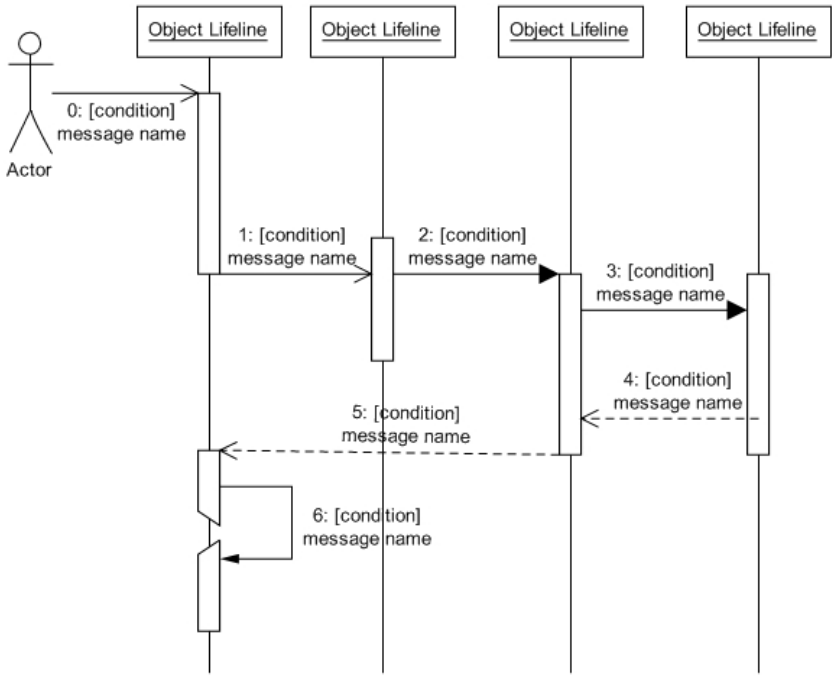
1. *Initial node*: menunjukan permulaan dari progres, dilambangkan dengan sebuah lingkaran solid.
2. *Actions*: dilambangkan dengan sebuah kotak dengan ujung membulat merepresentasikan setiap langkah individual. Urutan dari aksi membentuk keseluruhan dari aktivitas yang ditunjukkan oleh diagram.
3. *Flow*: dilambangkan dengan panah yang mengindikasikan progres melalui *actions*. Setiap panah tidak diperlukan kata untuk mengidentifikasikannya kecuali untuk menentukan pilhan.
4. *Decision*: dilambangkan dengan lambang wajik. Terdapat satu panah yang masuk dan terdapat 2 atau lebih panah yang keluar dari pilihan wajik. Setiap panah terdapat keterangan yang menerangkan kondisi.
5. *Merge*: dilambangkan dengan lambang wajik, terdapat 2 atau lebih panah yang masuk namun hanya satu panah yang keluar.
6. *Fork*: dilambangkan dengan kotak garis warna hitam, dimana hanya satu panah yang masuk, namun terdapat dua atau lebih panah yang keluar.

7. *Join*: dilambangkan dengan kotak garis warna hitam seperti *fork*, namun terdapat dua atau lebih panah yang masuk dan satu panah yang keluar.
8. *Activity final*: menunjukkan akhiran dari suat progress, dilambangkan dengan suatu lingkaran solid dilingkari oleh lingkaran kosong.

2.10.3 Sequence Diagram

Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.

Sequence diagram adalah diagram yang menggambarkan interaksi antara aktor dan sistem untuk sebuah skenario *use case* [18]. *Sequence diagram* menggambarkan bagaimana objek saling berinteraksi satu sama lain melalui *messages* dalam eksekusi dari sebuah *use case* atau operasi. Diagram ini menggambarkan langkah-langkah pesan dikirim dan diterima antara objek.



Gambar 2.7 Sequence Diagram

Berikut ini merupakan macam-macam notasi dari *sequence diagram*:

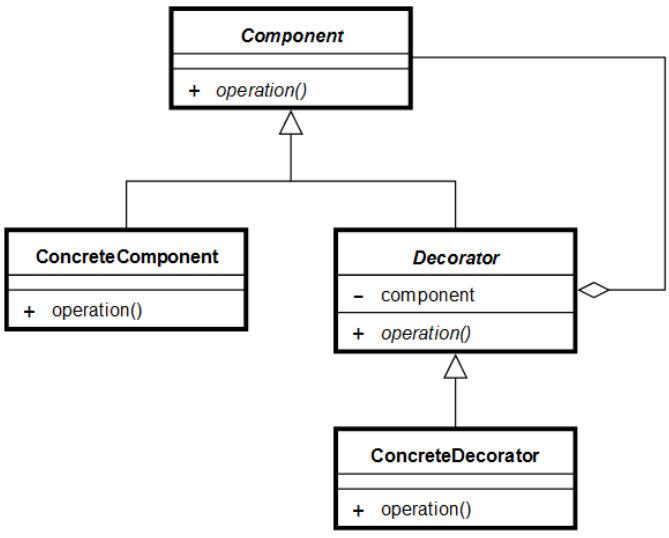
1. *Actor* : dilambangkan dengan gambar aktor dari *use case*.

2. *System* : dilambangkan dengan sebuah kotak.
3. *Lifelines* : garis putus-putus vertikal, yang merupakan perpanjangan dari aktor menuju dan simbol sistem, yang mengindikasikan rentetan.
4. *Activation bars* : sebuah bar yang mengindikasikan periode waktu kapan partisipan aktif di dalam interaksi.
5. *Input messages*: panah horizontal dari aktor menuju sistem yang mengindikasikan pesan masuk.
6. *Output messages* : panah horizontal dari sistem menuju aktor, ditunjukkan dengan panah bergaris putus-putus.

2.10.4 Class Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class diagram* menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan.

Class diagram digunakan untuk memodelkan kelas-kelas, termasuk atribut dan operasinya serta hubungan dan asosiasinya dengan kelas lainnya [5]. *Class diagram* menyediakan sebuah gambaran statis atau struktural dari sebuah sistem. *Class diagram* tidakmemperlihatkan komunikasi dinamis antar objek dari kelas yang ada dalam diagram.



Gambar 2.8 *Class Diagram*

2.11Android

Android adalah Sistem operasi pada ponsel yang dikembangkan oleh *Google* maka dibentuklah *Open Handset Alliance*. *Android* adalah *OS open source* pertama untuk ponsel .*Android* juga merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan *key-application* yang di-*release* oleh *Google*. Saat ini disediakan *Android SDK (Software Development Kit)* sebagai alat bantu dan *API (Application Programming Interface)* diperlukan untuk mulai mengembangkan aplikasi pada *platform android* menggunakan bahasa pemrograman *Java*.

2.11.1 Arsitektur Pada *Android*

Sistem operasi *android* memiliki 4 lapisan *(layer)* yang merupakan komponen sistem *android*. Gambar 2.7 merupakan lapisan arsitektur sistem operasi *android* [19].



Gambar 2.9 Overview Arsitektur Android

Google mengibaratkan *android* sebagai sebuah tumpukan *software*. Setiap lapisan dari tumpukan ini menghimpun beberapa program yang mendukung fungsi-fungsi spesifik dari sistem operasi. Berikut ini susunan dari lapisan – lapisan tersebut dilihat dari lapisan dasar hingga lapisan teratas.

2.11.1.1 Linux Kernel

Tumpukan paling bawah pada arsitektur *android* ini adalah *kernel. Google* menggunakan *kernel Linux* versi 2.6 untuk membangun *system android,* yang mencakup *memory management, security setting, power management* dan beberapa *driver hardware*. *Kernel* berperan sebagai *abstraction layer* antara *hardware* dan keseluruhan *software*. Sebagai contoh : *HTC GI* dilengkapi dengan kamera. *Kernel android* terdapat *driver* kamera yang memungkinkan pengguna mengirimkan perintah kepada *hardware* kamera.

Android dibangun di *atas kernel Linux* versi 2.6. Namun secara keseluruhan *android* bukanlah *linux,* karena dalam *android* tidak terdapat paket standar yang dimiliki oleh *linux* lainnya. *Linux* merupakan sistem operasi terbuka yang handal dalam manajemen *memori* dan proses. Oleh karenanya pada *android* hanya terdapat beberapa servis yang diperlukan seperti keamanan, manajemen *memori*, manajemen proses, jaringan dan *driver*. *Kernel linux* menyediakan *driver* layar, kamera, *WiFi, Flash Memory, audio* dan *IPC (Interprocess Communication)* untuk mengatur aplikasi dan lubang keamanan.

2.11.1.2 Libraries

Bertempat di level yang sama dengan *android runtime* adalah *libraries. Androi*d menyertakan satu *set library-library* dalam bahasan *C/C++* yang digunakan oleh berbagai komponen yang ada pada sistem *android*. Kemampuan ini dapat diakses oleh *programmer* melewati *android application framework*. Sebagai contoh *android* mendukung pemutaran format *audio, video* dan gambar. Berikut beberapa *core library* tersebut :

1. *System C Library* – variasi dari implemetasi standard *C* sistem *library (libc)* milik *BSD*, dioptimasi untuk piranti *embedded* berbasis *linux.*

2. *Media Libraries* – berdasarkan packet *video*‟s *openCORE*, *library-library* ini mendukung *playback* dan *recording* dari berbagai format audio dan *video* popular, *meliputi MPEG4, H.264, MP3, AAC, AMR, JPG,* dan *PNG.*
3. *Surface Manager* – Mengelola atau mengatur akses pada *display* dan lapisan *composites 2D* dan grafis *3D* dari berbagai aplikasi.
4. *LibWebCore – web browser engine* modern yang mendukung *android browser* maupun *embeddable web view.*
5. *SGL* – Mendasari grafis *2D*
6. *3D libraries* – implementasi berdasarkan *OpenGL ES* 1.0 *APIs, library* ini menggunakan perangkat keras akselerasi *3D* dan pengoptimalan *3D software rasterizer.*
7. *FreeType – bitmap* dan *vector font rendering*
8. *SQLite* – mesin *database* yang kuat dan ringan tersedia untuk semua aplikasi.

Library-library tersebut bukanlah aplikasi yang berjalan sendiri, namun hanya dapat digunakan oleh program yang berada di level atasnya. Sejak versi *android* 1.5 pengembang dapat membuat dan menggunakan pustaka sendiri menggunakan *NDK (Native Development Toolkit).*

2.11.1.3 Android Runtime

Lapisan berikutnya adalah *android runtime* yang berisi *core libraries* dan *dalvik virtual machine. Core libraries* mencakup serangkaian inti *library java*, artinya *android* menyertakan satu set *library-library* dasar yang menyediakan sebagian besar fungsi-fungsi yang ada pada *library-library* dasar bahasa pemprograman *java*. *Dalvik* adalah *java virtual machine* yang memberikan kekuatan pada sistem *android*. *Dalvik VM* ini di optimalkan untuk ponsel. Setiap aplikasi yang berjalan pada *android* berjalan pada prosesnya sendiri, dengan *instance* dari *dalvik virtual machine. Dalvik* telah dibuat sehingga sebuah piranti yang memakainya dapat menjalankan *multivirtual machin*e dengan efisien. *Dalvik VM* dapat mengeksekusi *file* dengan format *.dex (Dalvik Executable)* yang telah

dioptimasi untuk menggunakan minimal *memory footprint. Virtual machine* ini *register-based*, dan menjalankan *class-class* yang *decompile* menggunakan *"dx" tool* yang telah disertakan. *Dalvik virtual machine* menggunakan *kernel linux* untuk menjalankan fungsi – fungsi seperti *threading* dan *low level memory management*.

2.11.1.4 Application framework

Lapisan selanjutnya adalah *Application framework*, yang mencakup program untuk mengatur fungsi – fungsi dasar *smartphone*. *Application framework* merupakan serangkaian *tool* dasar seperti alokasi *resources smartphone*, aplikasi ponsel, pergantian antar proses atau program dan pelacakan lokasi fisik ponsel. Para pengembang aplikasi memiliki aplikasi penuh kepada *tool–tool* dasar tersebut dan memanfaatkannya untuk menciptakan aplikasi yang lebih kompleks.

Programmer mendapatkan akses penuh untuk memanfaatkan *API (Android Protocol Interface)* yang juga digunakan *core applications*. Arsitektur aplikasi didesain untuk menyederhanakan pemakaian kembali komponen–komponen, dimana setiap aplikasi dapat menunjukkan kemampuannya dan aplikasi lain dapat memakai kemampuan tersebut. Mekanisme yang sama memungkinkan pengguna mengganti komponen yang dikehendaki. Di dalam semua aplikasi terdapat servis dan sistem yang meliputi :

2. Satu *set Views* yang dapat digunakan untuk membangun aplikasi meliputi *lists, grids, text boxes, buttons,* dan *embeddable web browser*.
3. *Content Providers* yang memungkinkan aplikasi untuk mengakses data dari aplikasi lain misalnya *contact* atau membagi data yang dimiliki.
4. *Resources Manager*, menyediakan akses ke *non-code resources* misalnya *localized strings, graphics* dan *layout files.*
5. *Notification Manager* yang memungkinkan semua aplikasi untuk menampilkan *custom alerts* pada status bar.
6. *Activity Manager* yang mengatur *life cycle* dari aplikasi dan menyediakan *common navigation backstack*.

2.11.1.5 Applications

Di lapisan teratas terdapat aplikasi itu sendiri. Di lapisan inilah dapat ditemukan fungsi-fungsi dasar *smartphone* seperti menelepon dan mengirim pesan singkat, menjalankan *web browser*, mengakses daftar kontak, dan lain-lain. Sebagian penggunan pada lapisan inilah yang paling sering diakses, dimana pengaksesan fungsi-fungsi dasar tersebut melalui *user interface*.

2.11.2 *XML* Pada *Android*

XML (*Extensible Markup Language*) adalah bahasa *markup* yang mendifinisikan seperangkat aturan untuk *encoding* dalam dokumen yang bersifat *human-readable* dan *machine-readable.* Pada *android* sendiri berkas berjenis *XML* memiliki banyak fungsi, diantaranya :

1. Berkas *Android Manifest*

   Setiap aplikasi android harus memiliki berkas ini (dengan nama *AndroidManifest.xml*) pada direktori *root*. Pada berkas ini berisi informasi penting tentang aplikasi untuk sistem *android*.

2. *Layout*

   Android menyediakan kosakata *XML* sederhana yang sesuai dengan kelas *View* dan *subclass*-nya, seperti untuk *widget* dan *layout*.

3. *Database*

   Database XML adalah persistensi data perangkat lunak sistem yang memungkinkan data yang akan disimpan dalam format *XML*. Data ini kemudian dapat dilihat, di-*eksport* dan serialisasi ke dalam format yang diinginkan. *XML database* biasanya berhubungan dengan dokumen berorientasi *database*.

2.12*Tools* Pendukung Pembuatan Aplikasi *Android*

2.12.1 Java Development Kit (JDK)

JDK adalah *software development kit* merupakan seperangkat kode, *library*, *form* dan paket yang berguna untuk membuat aplikasi atau *applet java*. Di dalam *JDK* terdapat *Java Runtime Environment (JRE)* yang akan memproses *byte code* yang dihasilkan dari *interpenter java*. Tugas dari *interpenter java* ini adalah

untuk *mengcompile source.java* menjadi *source.class* yang selanjutnya *source.class* inilah yang akan dijalankan oleh *JRE*.

2.12.2 Software Development Kit (SDK)

SDK adalah suatu kumpulan dari *library* dan *toolkit* seperti *emulator* untuk mengembangkan atau menciptakan aplikasi untuk suatu perangkat lunak, perangkat komputer, sistem operasi atau *platform.* Di dalam *SDK* terdapat *tools* yang dibutuhkan dalam pengembangan *android*, diantaranya yaitu *adb shell* yang merupakan singkatan dari *android development bridge* yang dapat menjalankan terminal *android* seperti terminal pada sistem operasi *linux*, dan *command* yang terdapat dalam *adb shell* sendiri seperti *command linux* pada umumnya.

2.12.3 Eclips

Eclipse adalah sebuah *Integrated Development Environment (IDE)* untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform (platform-independent).* Berikut ini adalah sifat dari *Eclipse* :

1. *Multi-platform* : Target sistem operasi *eclipse* adalah *Microsoft Windows, Linux, Solaris, AIX, HP-UX* dan *Mac OS X.*
2. *Multi-language*: *Eclipse* dikembangkan dengan bahasa pemrograman *Java*, akan tetapi *Eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti *C/C++, Cobol, Python, Perl, PHP*, dan lain sebagainya.
3. *Multi-role:* Selain sebagai *IDE* untuk pengembangan aplikasi, *Eclipse* pun bisa digunakan untuk aktivitas seperti dokumentasi, test perangkat lunak, pengembangan *web*, dan lain sebagainya.