# BAB 2
# TINJAUAN PUSTAKA

## 2.1 Lagu Nasional

Lagu Nasional adalah suatu lagu yang diakui menjadi suatu lagu resmi dan menjadi simbol suatu negara atau daerah. Lagu kebangsaan dapat membentuk identitas nasional suatu negara dan dapat digunakan sebagai ekspresi dalam menunjukkan nasionalisme dan patriotisme.

Lagu nasional biasanya di gunakan dalam kegiatan misal upacara bendera, Dalam acara atau kegiatan olahraga internasional, dan acara resmi yang di selenggarakan pemerintah. Lagu nasional sendiri memiliki tatacara dalam penggunaannya salah satunya yaitu Lagu Kebangsaan yang diiringi alat musik, dinyanyikan lengkap satu strofe, dengan satu kali ulangan pada refrein. Setiap orang dilarang mengubah Lagu Kebangsaan dengan nada, irama, katakata, dan gubahan lain dengan maksud untuk menghina atau merendahkan kehormatan Lagu Kebangsaan[5].

## *2.2 Game*

*Game* berasal dari kata bahasa inggris yang berarti dasar permainan. *Game* adalah permainan yang menggunakan interaksi dengan antarmuka pengguna melalui gambar yang dihasilkan oleh piranti video. Permainan video umumnya menyediakan sistem penghargaan misalnya skor yang dihitung berdasarkan tingkat keberhasilan yang dicapai dalam menyelesaikan tugas-tugas yang ada di dalam permainan. *Game* merupakan sesuatu hal yang dimainkan dengan suatu aturan tertentu yang biasa digunakan untuk tujuan kesenangan dan dapat juga digunakan untuk tujuan pendidikan. *Game* umumnya melibatkan stimulasi mental, fisik, atau keduanya. Banyak *game* yang dapat membantu mengembangkan keterampilan praktis yang berfungsi sebagai latihan, atau melakukan peran pendidikan, simulational, atau psikologis.

*Game* bertujuan untuk menghibur, *game* banyak disukai oleh anak-anak hingga orang dewasa. *Game* sebenarnya penting untuk perkembangan otak, untuk

meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam *game* terdapat berbagai konflik atau masalah yang menuntut kita untuk menyelesaikannya dengan cepat dan tepat. Tetapi *game* juga bisa merugikan karena apabila kita sudah kecanduan *game* kita akan lupa waktu dan akan menggangu kegiatan atau aktivitas yang sedang kita lakukan.[6]

### 2.2.1 Jenis Permainan (Genre *Game*)

*Game* dari tahun-tahun semakin banyak saja jenisnya, teknologi yang dipakaipun semakin canggih. Terdapat ratusan atau bahkan ribuan game yang telah bermunculan di dunia, game-game tersebut dikelompokkan menjadi beberapa *genre*. Pada dasarnya, *genre-genre game* yang beredar di pasaran dapat dikelompokkan sebagai berikut. [6]

1. *Action Game*

   Adalah tipe *game* dengan fitur utama berupa banyaknya aksi di mana pemain harus memiliki keterampilan reaksi yang cepat untuk menghindari musuh atau menghindari rintangan. Pengembang *game* tipe ini perlu memastikan *game* yang dibuat dioptimasi sehingga pemain memiliki pengalaman bermain yang baik, yang tidak terganggu oleh delay proses yang lama, contohnya: MetalGear Solid.

2. *Adventure Game*

   Adalah tipe *game* yang umumnya membuat pemain harus berjalan mengelilingi suatu tempat yang terkondisi, seperti sebuah istana, gua yang berkelok, dan planet yang jauh. Pemain melakukan navigasi suatu area, mencaripesan-pesan rahasia, memperoleh obyek yang memiliki kemampuan yang bervariasi, bertempur dengan musuh, dan lain-lain. Untuk membuat *game* ini,diperlukan perencanaan yang akurat sehingga memiliki alur cerita yang menarik bagi pemain, contohnya: Tomb Raider.

3. Edukasi

   Video *game* jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, baik itu untuk belajar mengenal warna, mengenal huruf dan angka, matematika, belajar bahasa asing dan lain-lain. *Developer* yang membuatnya, harus memperhitungkan berbagai hal agar game ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan keterampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan desain visual ataupun animasinya.

4. Simulasi, konstruksi dan manajemen

   Video *game* jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Dari mencari jodoh dan pekerjaan, membangun rumah, gedung hingga kota, mengatur pajak dan dana kota hingga keputusan memecat atau menambah karyawan. Dunia kehidupan rumah tangga sampai bisnis membangun konglomerasi, dari jualan minum pinggir jalan hingga membangun laboratorium cloning. *Video game* jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas.

5. Petualangan

   Refleks dan kelihaian pemain dalam bergerak, berlari, melompat hingga memecut atau menembak tidak diperlukan disini. *Video game* murni petualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat, termasuk didalamnya petualangan dengan teks atau sistem tunjuk dan klik.

6. *Platform Game*

   Adalah tipe *game* yang mengharuskan pemain mengarahkan suatu obyek dengan melalui berbagai tahap atau tingkatan area untuk menyerang musuh dan menghindar terhadap serangan. Tipe game ini sedikit serupa dengan *action game*, tetapi aksinya tidak secepat *action game*. Teknik *collision detection* sangat sering dimanfaatkan pada tipe game ini, contohnya: Sonic The Hedgehog & Mario Bros.

7. *Music Game*

   Adalah tipe *game* yang menuntut pemainnya untuk menekan tombol sesuai dengan tombol yang ada di layar dengan diiringi dengan musik, contohnya: Guitar Hero & Dance Dance Revolution

8. Olahraga

   Biasanya *game genre* ini permainannya diusahakan serelistik mungkin dibuat semirip mungkin dengan aslinya walau kadang ada yang menambah unsur fiksi dalam game olahraga tersebut.

### 2.2.2 Sudut Pandang Permainan *(Game View Point)*

Sebuah *game* biasanya mempunyai sudut pandang permainan tersendiri disesuaikan berdasarkan *genre game* yang diambil. Berikut beberapa macam sudut pandang permainan yang biasa digunakan:

1. *Side Scrolling*

   Adalah sudut pandang permainan yang terlihat dari samping dan memungkinkan karakter utama untuk bergerak dari kiri ke kanan serta memungkinkan background pada game seolah-olah bergeser mengikuti pergerakan karakter utama.

2. *Top Down*

   Adalah sudut pandang permainan yang memungkinkan karakter utama bermanuver ke empat arah namun cara permainannya sendiri bergeser dari bawah ke atas, dan biasanya game yang menggunakan sudut pandang permainan jenis ini adalah shooter game.

3. *Isometric*

   Adalah sudut pandang permainan yang memungkinkan permainan terlihat diantara sisi *Side Scrolling* dan juga *Top Down*, dan biasanya diterapkan pada *game* dengan *genre* RTS (*Real Time Strategy*).

### 2.2.3 Klasifikasi Permainan (Game Classification)

Berikut ini adalah klasifikasi permainan berdasarkan pembatasan usia dari pemainnya, menurut *ESRB (Entertainment Software Rating Board)* adalah sebagai berikut:

1. *Early Childhood* (3+) memiliki isi yang mungkin cocok untuk usia 3 keatas. Tidak mengandung materi yang tidak pantas seperti adegan kekerasan.
2. *Everyone* (6+) memiliki konten yang mungkin cocok untuk usia 6 keatas. Judul-judul dalam kategori ini mungkin berisi sedikit kartun, fantasi atau kekerasan ringan dan tidak jarang menggunakan bahasa yang ringan.
3. *Everyone* (10+) memiliki konten yang mungkin cocok untuk usia 10 keatas. Judul-judul dalam kategori ini mungkin berisi lebih banyak kartun, fantasi atau kekerasan ringan, bahasa ringan dan juga minimal tema sugestif.
4. *Teen* (13+) memiliki konten yang mungkin cocok untuk usia 13 keatas. Judul-judul dalam kategori ini mungkin mengandung kekerasan, tema sugestif, humor kasar, sedikit darah, simulasi perjudian, dan tidak jarang menggunakan bahasa yang kuat.
5. *Mature* (17+) memiliki konten yang mungkin cocok untuk orang usia 17 dan lebih tua. Judul-judul dalam kategori ini mungkin berisi intens kekerasan, darah, konten seksual dan bahasa yang kuat.
6. *Adults Only* (18+) memiliki konten yang hanya boleh dimainkan oleh orang-orang 18 tahun keatas. Judul-judul dalam kategori ini mungkin termasuk adegan yang intens, kekerasan dan juga mengandung konten seksual.

## *2.3 Game Edukasi*

Penerapan *education game* bermula dari perkembangan *video game* yang sangat pesat dan menjadikannya sebagai media efektif yang interaktif dan banyak dikembangkan di perindustrian. Melihat kepopuleran game tersebut, para pendidik berpikir bahwa mereka mempunyai kesempatan yang baik untuk menggunakan komponen rancangan game dan menerapkannya pada kurikulum dengan penggunaan industri berbasis *game. Game* harus memiliki desain antar muka yang interaktif dan mengandung unsur menyenangkan.

*Education game* adalah *game* yang khusus dirancang untuk mengajarkan user suatu pembelajaran tertentu, pengembangan konsep dan pemahaman dan membimbing mereka dalam melatih kemampuan user, serta memotivasi user untuk memainkannya.[6]

### 2.3.1 Kriteria *Game* Edukasi

Menurut Hurd dan Jenuings perancangan *education game* yang baik haruslah memenuhi kriteria dari *education game* itu sendiri. Berikut ini adalah beberapa kriteria dari sebuah education game, yaitu:[7]

1. Nilai Keseluruhan (*Overall Value*)

   Nilai keseluruhan dari suatu game terpusat pada desain dan panjang durasi *game*. Aplikasi ini dibangun dengan desain yang menarik dan interaktif. Untuk penentuan panjang durasi, aplikasi ini menggunakan fitur *timer*.

2. Dapat Digunakan (*Usability*)

   Mudah digunakan dan diakses adalah poin penting bagi pembuat *game*. Apliksi ini merancang sistem dengan *interface* yang *user friendly* sehingga user dengan mudah dapat mengakses aplikasi.

3. Keakuratan (*Accuracy*)

   Keakuratan diartikan sebagai bagaimana kesuksesan model/gambaran sebuah game dapat dituangkan ke dalam percobaan atau perancangannya. Perancangan aplikasi ini harus sesuai dengan model *game* pada tahap perencanaan.

4. Kesesuaian (*Appropriateness*)

   Kesesuaian dapat diartikan bagaimana isi dan desain *game* dapat diadaptasikan terhadap keperluan *user* dengan baik. Aplikasi ini menyediakan menu dan fitur yang diperlukan user untuk membantu pemahaman user dalam menggunakan aplikasi.

5. Relevan (*Relevance*)

   Relevan artinya dapat mengaplikasikan isi game ke target *user*. Agar dapat relevan terhadap user, *sistem* harus membimbing mereka dalam pencapaian tujuan pembelajaran. Karena aplikasi ini ditujukan untuk anak-anak maka desain antarmuka harus sesuai dengan nuansa anak-anak, yaitu menampilkan warna-warna yang ceria.

6. Objektifitas (*Objectives*)

   Objektifitas menentukan tujuan user dan kriteria dari kesuksesan atau kegagalan. Dalam aplikas ini objektivitas adalah usaha untuk mempelajari hasil dari permainan.

7. Umpan Balik (*Feedback*)

   Untuk membantu pemahaman user bahwa permainan *(performance*) mereka sesuai dengan objek game atau tidak, *feedback* harus disediakan. Aplikasi ini menyajikan animasi dan efek suara yang mengindikasikan kesuksesan atau kegagalan permainan.

## 2.4 Minat

### 2.4.1 Pengertian Minat

Minat dalam Kamus Besar Bahasa Indonesia berartikan cenderungan hati yang tinggi terhadap sesuatu. Minat merupakan sifat yang relatif menetap pada diri seseorang. Minat besar sekali pengaruhnya terhadap kegiatan seseorang, sebab dengan minat ia akan melakukan sesuatu yang diminatinya. Sebaliknya, tanpa minat seseorang tidak mungkin melakukan sesuatu.[8]

Menurut Joko Sudarsono "Minat merupakan bentuk sikap ketertarikan atau sepenuhnya terlibat dalam suatu kegiatan karena menyadari pentingnya atau bernilainya kegiatan tersebut".[9] Definisi secara sederhana lainnya diberikan oleh Muhibbin Syah yang mendefinisikan bahwa " Minat (*interest*) berarti kecenderungan atau kegairahan yang tinggi atau keinginan yang besar terhadap sesuatu".[10] Begitu pula dengan Slameto yang menyatakan bahwa "Minat adalah suatu rasa lebih suka dan rasa keterikatan pada suatu hal atau aktivitas, tanpa ada yang menyuruh.[11]

Hilgrad (Slameto,) menyatakan *"Interest is persisting tendency to pay attention to and enjoy some activity and content."* Minat adalah kecenderungan yang tetap untuk memperhatikan dan mengenang beberapa kegiatan. Kegiatan yang diminati diperhatikan terus menerus dengan disertai rasa senang dan diperoleh rasa kepuasan. Lebih lanjut dijelaskan minat adalah suatu rasa suka dan ketertarikan pada suatu hal atau aktivitas tanpa ada yang menyuruh[11]. Minat adalah kecenderungan dalam diri individu untuk tertarik pada suatu objek atau menyenangi suatu objek.

Dari beberapa pendapat yang dikemukakan oleh para ahli seperti yang dikutip di atas, dapat disimpulkan bahwa minat adalah perhatian, rasa suka dan rasa ketertarikan seseorang (siswa) terhadap suatu kegiatan yang ditunjukkan dengan adanya partisipasi, keinginan siswa untuk melakukan aktivitas dengan baik dan perhatian siswa dalam materi kegiatan secara aktif dan serius.

### 2.4.2 Indikator Minat

Dalam Kamus Besar Bahasa Indonesia, indikator adalah alat pemantau (sesuatu) yang dapat memberikan petunjuk/keterangan (Depdikbud).[8] Hubungannya dengan minat siswa, maka indikator adalah sebagai alat pemantau yang dapat memberikan petunjuk ke arah minat. Ada beberapa indikator siswa yang memiliki minat yang tinggi, hal ini dapat dikenali melalui proses kegiatan maupun di rumah.

Slameto menyatakan bahwa Minat adalah suatu rasa lebih suka dan rasa ketertarikan pada suatu hal atau aktivitas, tanpa ada yang menyuruh.[11] Minat pada dasarnya adalah penerimaan akan suatu hubungan antara diri sendiri dengan sesuatu di luar diri. Semakin kuat atau dekat hubungan tersebut semakin besar minat. Suatu minat dapat diekspresikan melalui suatu pernyataan yang menunjukkan bahwa siswa lebih menyukai suatu hal daripada hal lainnya, dapat pula dimanifestasikan melalui partisipasi dalam suatu aktivitas. Siswa yang memiliki minat terhadap subjek tertentu cenderung untuk memberikan perhatian yang lebih besar terhadap subjek tertentu.

Berdasarkan definisi tersebut, maka dapat disimpulkan indikator minat yaitu rasa suka/senang dalam aktivitas melakukan kegiatan, rasa ketertarikan untuk melakukan kegiatan, adanya kesadaran untuk melakukan kegiatan tanpa disuruh, berpartisipasi dalama aktivitas, memberikan perhatian yang besar dalam kegiatan.

## 2.5 Algoritma Fisher-Yates Shuffle.

*Shuffle Fisher-Yates* (bernama setelah *Ronald Fisher* dan *Frank Yates*), juga dikenal sebagai *shuffle Knuth* (setelah *Donald Knuth*), adalah sebuah algoritma untuk menghasilkan permutasi acak dari seperangkat terbatas dalam hal biasa, untuk secara acak mengocok set. Sebuah varian dari *shuffle Fisher-Yates*, yang dikenal sebagai algoritma Sattolo itu, dapat digunakan untuk menghasilkan acak siklus panjang n sebagai gantinya. Jika diimplementasikan dengan benar, maka hasil dari *algoritma* ini tidak akan berat sebelah, sehingga setiap permutasi memiliki kemungkinan yang sama. Versi modern dari algoritma ini adalah juga

agak efisien, hanya membutuhkan waktu sebanding dengan jumlah barang yang dikocok dan tidak ada ruang penyimpanan tambahan[12].

Proses dasar dari *Fisher-Yates* menyeret mirip dengan memilih secara acak tiket bernomor keluar dari cab, atau kartu dari setumpuk, satu demi satu sampai tidak ada yang lebih kiri. Algoritma *Fisher-Yates* menyediakan cara untuk melakukannya pengurutan secara numerik dengan cara yang efisien dan ketat, benar dilakukan, menjamin hasil yang bias. Gambar 2.1 algoritma *fisher – yates shuffle* yang ditulis dalam bahasa javascript.[13]

```
function shuffle(array) {
  var m = array.length, t, i;

  // While there remain elements to shuffle…
  while (m) {

    // Pick a remaining element…
    i = Math.floor(Math.random() * m--);

    // And swap it with the current element.
    t = array[m];
    array[m] = array[i];
    array[i] = t;
  }

  return array;
}
```

**Gambar 2. 1 algoritma *fisher – yates shuffle* dalam bahasa javascript.**

Tabel 2.1 berikut adalah tahapan dan langkah *algoritma Fisher-Yates Shuffle* dan dimasukkan ke dalam pseudocode.

**Tabel 2. 1 pseudocode *algoritma Fisher-Yates Shuffle***

| No | Algoritma | Pseudocode |
|---|---|---|
| 1 | Ketika masih ada elemen tersisa untuk diacak | While (m) { |
| 2 | Ambil elemen secara acak dari elemen yang tersisa | i = Math.floor(Math.random() *m--); |
| 3 | Kemudian tukar dengan elemen saat ini | t = array[m];<br>array[m] = array [i];<br>array[i] = t;<br>} |

Kelebihan algoritma fisher yattes dibanding random:

1. Algoritma random hanya menghasilkan sebuah bilangan yang dipilih secara acak untuk satu kali pengacakan sehingga kemungkinan terjadi perulangan kemunculan bilangan yang telah muncul sebelumnya sangat besar. Pada algoritma fisher yattes pengacakan dilakukan berdasarkan urutan sehingga kemunculan bilangan yang sama untuk kedua kalinya bias dihindari.
2. Fisher yattes cocok digunakan pada kasus yang membutuhkan pengacakan urutan angka atau kata dalam suatu array.

Kekurangan algoritma Fisher yattes dibanding algoritma random:

Algoritma *Fisher yattes* tidak cocok digunakan pada kasus yang hanya memerlukan 1 atau 2 array dari banyak array yang diacak karena algoritma *Fisher yattes* mengurutkan *array* terlebuh dahulu sehingga tidak *efisien*. Jika hanya membutuhkan 1 atau 2 *array* saja lebih mudah menggunakan random.

## 2.6 *Tools* yang digunakan

Pada sub bab ini akan dijelaskan *tools* yang digunakan dalam pembangunan *Game* edukasi lagu nasional.

### *2.6.1 Action Script 3.0*

*ActionScript 3.0* adalah adalah bahasa berorientasi objek untuk membuat aplikasi dan konten multimedia ber-script untuk pemutaran di runtime *Flash client* (seperti Flash Player dan Adobe AIR). Dengan sintaks yang mirip dengan Java dan C#, bahasa A*ctionScript* seharusnya akrab bagi programmer berpengalaman [14].

Beberapa fitur kunci dari bahasa ActionScript 3.0 adalah:

1. *First-class* support untuk constructs berorientasi objek umum, seperti kelas, objek, dan interface
2. *Single-threaded execution model*

3. *Runtime type-checking*
4. Compile-time type-checking opsional
5. Fitur dinamis seperti *runtime* pembuatan fungsi konstruktor baru dan variable
6. *Runtime exceptions*
7. *Direct support* untuk CML sebagai tipe data built-in
8. Package untuk pengorganisasian code libraries
9. Namespaces untuk kualifikasi identifier
10. *Regular expressions*

### *2.6.2 Adobe Flash*

*Flash* adalah aplikasi yang sangat kaya sehingga menarik bagi designer maupun developer multimedia. Para designer sangat menyukai aplikasi ini karena kemudahannya dalam menghasilkan animasi, sedangkan para *developer* menikmati perlengkapannya untuk membuat aplikasi-aplikasi hebat yang ada pada Flash. Flash merupakan standar professional untuk menghasilkan *web experience* yang bercita-rasa tinggi [15].

Dalam membuat sebuah aplikasi menggunakan Macromedia Flash, terdapat beberapa konsep dasar yang perlu untuk dipahami terlebih dahulu. Konsep dasar tersebut meliputi *animasi, movie, objek, teks, sound* dan simbol.

1. Objek

   *Flash* menyediakan *tool* untuk membuat objek sederhana seperti garis,lingkaran, persegi empat.

2. Teks

   Pada *toolbox* disediakan fasilitas untuk menulis teks. Flash teks dikategorikan dalam tiga jenis yaitu teks statis label, teks dinamis, dan teks input.

3. Simbol

Dalam Macromedia Flash ada beberapa simbol yang harus diketahui yaitu *movie clip, button,* dan *graphic*. Masing-masing simbol mempunyai fungsi tersendiri.

4. Sound

Format sound yang dapat pergunakan dalam Flash dapat bermacam macam seperti WAV, MP3. Sound dapat mengimpor dari luar tetapi untuk sound-sound tertentu telah disediakan di dalam program Flash.

5. Animasi

Animasi adalah proses penciptaan efek gerak atau efek perubahan bentuk yang terjadi selama beberapa waktu. Animasi bisa berupa gerak sebuah objek dari tempat yang satu ke tempat yang lain, perubahaan warna, atau perubahan bentuk.

6. Movie

Dalam membuat animasi, maka seseorang akan mengatur jalan cerita dari animasi tersebut, membuat beberapa objek dan merangkainya menjadi suatu bagian yang bermakna tertentu. Suatu movie terkadang terdiri dari beberapa animasi yang biasa disebut movie clip

### 2.6.3 XML

*XML* adalah sebuah meta-language (bahasa yang digunakan untuk mendeskripsikan bahasa lain) yang memungkinkan seorang desainer membuat sendiri tag yang menyediakan fungsi yang tidak tersedia pada *HTML (Connolly dan Begg, 2010, p.1073).* XML merupakan sebuah versi meta-language yang tidurunkan dari *Standard Generalized Markup Language (SGML)*, yang didesain khusus untuk dokumen website dan dapat mendukung perancang untuk menciptakan tag sendiri, yang memiliki kemampuan untuk mendefinisikan, mentransmisikan, memvalidasikan dan menginterprestasikan data antara aplikasi dan organisasi.[16]

XML dikembangkan oleh *World Wide Web Consortium (W3C)* yang didukung sekitar 150 orang anggotanya dan versi 1.0 pertama kali dirilis pada tahun 1998.

Keunggulan yang dimiliki XML antara lain:

1. *Simplicity, XML* menggunakan bahasa yang sederhana, mudah dimengerti oleh manusia dan mesin.
2. *Open standard and platform/vendor-independent, XML* menggunakan standar terbuka dan tidak bergantung pada *platform* tertentu.
3. *Extensibility, XML* memungkinkan *user* untuk mendefinisikan *tag* sendiri sehingga dapat berkembang sesuai dengan kebutuhan *user.*
4. *Reuse, XML* memungkinkan *libraries* dari XML *tags* untuk dibangun sekali dan dapat digunakan kembali oleh banyak aplikasi.
5. *Separation of content and presentation, XML* memisahkan antara isi dan tampilan dari suatu data sesuai dengan pengaturan yang diinginkan.
6. *Improved load balancing,* data dapat ditampilkan pada *browser* dengan baik.
7. *Support for the integration of data from multiple source.* Kemampuan mengintegrasikan data dari berbagai macam sumber yang berbeda adalah hal yang sulit dan memakan waktu. *XML* dapat menggabungkan data dari banyak sumber yang berbeda dengan cara yang mudah.
8. *Ability to describe data from a wide variety of applications, XML* dapat digunakan untuk mendeskripsikan data yang terdapat pada aplikasi yang berbeda.
9. *More advanced search engine,* dengan *XML* mesin pencarian akan mampu untuk menyederhanakan berbagai macam *tags.*
10. *New opportunities, XML* memiliki banyak kelebihan yang dapat dihadirkan dalam banyak aplikasi teknologi saat ini. Struktur hirarkisnya cocok untuk kebanyakan tipe dokumen.

Kekurangan-kekurangan XML

1. *Parser* harus didesain untuk memahami struktur data bersarang yang berubah-ubah dan harus melakukan pengecekan tambahan untuk mendeteksi sintaks atau data yang tidak terformat atau terurut dengan benar.
2. urutan penekanan tombol untuk mengetikkan ekspresi *XML* pada *keyboard* standar komputer seringkali tidak fleksible.

XML menggunakan teknologi Document Type Definitions (DTDs) yang dapat mendefinisikan sintaksis yang valid dari dokumen XML.

#### *2.6.3.1* Deklarasi *XML*

Sebuah file XML diawali dengan pilihan deklarasi *XML*, yang menunjukan versi *XML* yang digunakan oleh penulis dalam dokumen, encoding system yang digunakan dan menentukan apakah ada deklarasi external markup yang perlu dimasukkan, *XML* bersifat *case-sensiftive* (huruf kecil tidak sama dengan huruf besar). Artinya tag <*home*> dan <*Home*> merupakan hal yang berbeda.

#### *2.6.3.2 Element XML*

*Element* atau *tag* adalah bentuk umum dalam *markup*. Elemen pertama pasti adalah sebuah *root element*, yang terdiri atas banyak *sub element* lain. Sebuah dokumen *XML* harus mempunyai satu *root element*.

#### *2.6.3.3 Atribut XML*

*Atribut* adalah pasangan nama dan nilai yang mendeskripsikan informasi tentang suatu *element*. Atribut diletakkan dalam tag awal setelah nama elemen dan nilai *atribut* ada dalam tanda petik. Contoh:

<STAFF brachNo=”B001”>

Jika sudah diberikan atribut maka dapat ditunjukkan element itu adalah *element* yang kosong, contoh: <SEX gender=”M”/>

#### *2.6.3.4 Entity References*

Setiap *entity* harus memiliki nama yang unik dan userannya dalam sebuah dokumen *XML* disebut dengan entity references. Sebuah *entity references* diawali dengan tanda dan (&) dan diakhiri dengan titik koma (;). *Entity references* adalah karakter yang mengganti *illegal* character di *XML*.

#### *2.6.3.5 Komentar XML*

Komentar pada *XML* menggunakan sintaks sama seperti komentar pada *HTML*.

Contoh:

<!—komentar -->

Komentar ini dapat mengandung semua string kecuali '—'.

### *2.6.4 UML*

*UML* adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sebuah sistem. *UML* adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek *(*OOP*)* serta aplikasinya. *UML* adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. *UML* mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980- an. Sekarang *UML* sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari IBM.[17]

*UML* memiliki 13 jenis diagram. *Spesifikasi UML* tidak mengatakan di mana diagram harus digunakan dalam metodologi tertentu.

1. *Use Case Diagram* mengkategorikan cara di mana sistem yang digunakan.
2. *Class Diagram* menunjukkan kelas dan bagaimana kelas dapat dipasang bersama-sama (mereka juga dapat menunjukkan *object*).
3. *Object Diagram* memperlihatkan *obyek* saja dan bagaimana *obyek* dapat dipasang bersama-sama.

4. *Activity Diagram* menunjukkan aktivitas oleh manusia atau objek dalam cara yang mirip dengan *Flow Chart*.
5. *State Machine Diagram* menunjukkan berbagai state setiap obyek dengan siklus hidup yang menarik atau rumit.
6. *Communication Diagram* menunjukkan pesan yang dikirim antara *objek* dalam *skenario* tertentu.
7. *Sequence Diagram* menunjukkan informasi yang sama dengan *communication diagram*, tetapi menekankan urutan daripada koneksi.
8. *Package Diagram* menunjukkan bagaimana kelas terkait dikelompokkan bersama-sama, untuk kepentingan pengembang.
9. *Deployment Diagram* menunjukkan mesin, proses dan artefak dikerahkan untuk jadi seorang sistem.
10. *Component Diagram* menunjukkan komponen dapat digunakan kembali (benda atau subsistem) dan *interface* mereka.
11. *Interaction Overview Diagram* menunjukkan langkah-langkah individu dari suatu kegiatan menggunakan *sequence diagram*.
12. *Timing Diagram* menunjukkan kendala waktu yang tepat untuk pesan dan menyatakan objek.
13. *Composite Structure Diagram* menunjukkan bagaimana benda cocok bersama dalam suatu agregasi atau komposisi, menampilkan antarmuka dan berkolaborasi benda.

#### *2.6.4.1 Use Case Diagram*

*Use case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.[18]

*Use case diagram* dapat sangat bisa digunakan untuk menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test* case untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include fungsionalitas use case* lain sebagai bagian dari *proses* dalam dirinya. Secara umum diasumsikan bahwa use case yang di-*include* akan dipanggil setiap kali *use case* yang meng-include dieksekusi secara normal.

Sebuah *use case* dapat di-include oleh lebih dari satu use case lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar use case menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain.

#### *2.6.4.2 Activity Diagram*

*Activity diagrams* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana alir berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. [18]

*Activity diagram* merupakan state diagram khusus, di mana sebagian besar state adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur *aktivitas* dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu use case atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara use case menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti state, *standar UML* menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk

menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik *sinkronisasi* yang dapat berupa titik, garis *horizontal* atau *vertikal.*

*Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

#### *2.6.4.3 Sequence Diagram*

*Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi *horizontal* (objek-objek yang terkait). [18]

*Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger aktivitas* tersebut, *proses* dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline vertikal*. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada *fase desain* berikutnya, message akan dipetakan menjadi operasi/metoda dari class. Activation bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk *objek boundary,* controller dan *persistent entity*.

#### *2.6.4.4 Class Diagram*

*Class adalah* sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (*atribut/properti*) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). [18]

*Class diagram* menggambarkan struktur dan *deskripsi class, package* dan *objek* beserta hubungan satu sama lain seperti *containment,* pewarisan, *asosiasi,* dan lain-lain.

Class memiliki tiga area pokok:

1. Nama (dan *stereotype*)
2. *Atribut*
3. *Metoda*

*Atribut* dan *metoda* dapat memiliki salah satu sifat berikut:

1. *Private,* tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected,* hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
3. *Public,* dapat dipanggil oleh siapa saja

*Class* dapat merupakan implementasi dari sebuah *interface,* yaitu class abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah class. Dengan demikian *interface* mendukung resolusi metoda pada saat run-time.

Sesuai dengan perkembangan *class model, class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package.*

**Hubungan Antar Class:**

1. *Asosiasi*, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah query antar *class*.
2. *Agregasi*, yaitu hubungan yang menyatakan bagian ("terdiri atas..").

3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari class lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan *fungsionalitas* baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah *generalisasi*.
4. Hubungan dinamis, yaitu rangkaian pesan (message) yang di-passing dari satu *class* kepada class lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

## 2.7 Jaringan Semantik

Menurut Wijaya jaringan semantik merupakan jaringan data dan informasi yang menunjukan hubungan antar berbagai objek dimana informasi yang terhubung tersebut adalah informasi yang proporsional (suatu pernyataan yang dapat bernilai benar atau salah). Dalam matematika, istilah jaringan semantik merupakan suatu label atau *graph* berarah.[19]

Struktur jaringan semantik terdiri dari node atau simpul dan busur atau arc yang menghubungkannya. Simpul menyatakan objek sedangkan busur menyatakan links. Links dari jaringan semantik digunakan untuk menunjukkan hubungan (*relationship*) antar simpul-simpul tersebut.

## 2.8 Pretest dan posttest

Desain pretest-posttest tumbuh dari posttest sederhana hanya desain, dan mengatasi beberapa masalah yang timbul dengan tugas dan alokasi peserta untuk kelompok.

Salah satu contoh adalah pendidikan, di mana peneliti ingin memantau pengaruh metode pengajaran baru pada kelompok anak-anak. Daerah lain termasuk mengevaluasi dampak dari konseling, pengujian pengobatan medis, dan mengukur konstruk psikologis. Satu-satunya ketentuan adalah bahwa subjek harus acak ditugaskan untuk kelompok, dalam desain eksperimental yang benar, untuk benar mengisolasi dan meniadakan setiap gangguan atau variabel pengganggu.[20]

### 2.8.1 Posttest Hanya Desain Dengan Non-Equivalent Grup Kontrol

Desain pretest-posttest merupakan perluasan posttest hanya desain dengan kelompok nonequivalent, salah satu metode yang paling sederhana untuk menguji efektivitas intervensi.

Dalam desain ini, yang menggunakan dua kelompok, satu kelompok diberi pengobatan dan hasilnya dikumpulkan di akhir. Kelompok kontrol tidak menerima pengobatan, selama periode waktu yang sama, namun mengalami persis tes yang sama.

Kemudian dihitung nilai rata – rata hasil dengan menggunakan rumus:

$$M=\frac{\sum X}{\sum Y} \qquad (2.1)$$

Keterangan:

M : Nilai Rata-Rata

ΣX : Jumlah nilai

ΣN : Jumlah responden

Setelah mendapatkan rata – rata kemudian dicari gain ternormalisasi menggunakan rumus:[21]

$$\langle g \rangle = \frac{posttest\ score - pretest\ score}{maximum\ possible\ score - pretest\ score} \qquad (2.2)$$

1. Uji Normalitas

Uji Normalitas dilakukan untuk memeriksa data sample berasal dari populasi yang berdistribusi normal. Syaratnya adalah data sampel dikatakan berdistribusi normal apabila skor yang diperoleh berasal dari suatu populasi yang berdistribusi normal. [22]

Lankah langkah untuk mengetahui berdistribusi normal:

a. Merumuskan hipotesis pengujian normalitas data:
   H0: data sampel berasal dari populasi yang bedistribusi normal
   H1: data sampel berasal dari populasi yang tidak bedistribusi normal
b. Menguji data normalitas dengan menggunakan uji *Shapiro-Wilk*
c. Melihat nilai signifikansi pada kolom *Shapiro-Wilk,* dengan menggunakan taraf signifikasi 5% (α=0,05), Jika nilai signikasi < 0,05, maka $H_0$ ditolak.
d. Jika kedua pengujian normal maka dilakukan uji homogenitas. Jika salah satu kelas eksperimen dan kelas kontrol terdistribusi tidak normal, maka tidak diuji homogenitas, tetapi digunakan uji statistik NON-parametik dengan uji Mann-Whitney.

2. Uji Homogenitas

Uji homogenitas dilakukan untuk menguji homogen atau tidaknya data sampel yg diambil dari data yang sama. Menguji homogenitas menggunakan langkah-langah berikut [22]

Lankah-langkah untuk uji homogenitas:

e. Merumuskan hipotesis pengujian homogenitas data:
   H0: data sampel berasal dari varians yang sama atau homogen
   H1: data sampel berasal dari varians yang tidak sama atau tidak homogen
f. Menguji homogenitas dengan menggunakan rumusan *levene's test*
g. Melihat nilai signifikansi pada iji *levene's test,* dengan menggunakan taraf signifikasi 5% (α=0,05), Jika nilai signikasi < 0,05, maka $H_0$ ditolak.

3. Hipotesis

Uji hipotesis dilakukan untuk mengetahui peningkatan penguasaan kelas eksperimen dan kelas kontrol setelah masing-masing kelas diberi perlakuan yang berbeda, yang dilihat dari hasil nilai selisih pretest dan postest kedua kelas tersebut (gain).[22]

Dilakukan uji hopotesih dengan tahap sebagai berikut:

1. Merumuskan hipotesis pengujian
   H0: Tidak terjadi peningkatan
   H1: Terdapat peningkatan
2. Menggunakan uji statistik *non-parametik* dengan uji *Mann-whitney*
3. Malihat nilai t bada baris dengan taraf signifikansi. Dengan menggunakan taraf signifikansi 5% (α=0,05). kriteria pengambilan keputusan jika signikasi > 0,05 maka H0 ditolak.

## 2.9 Penghitungan hasil kuesioner

Dalam mengolah data kuesioner menggunakan Skala Likert dengan rumus sebagai berikut:

1. Bobot Jawaban

Bobot jawaban yang digunakan mempunyai gradasi dari sangat positif hingga sangat negatif berupa kata-kata seperti pada Tabel 2.2 berikut.

**Tabel 2. 2 Bobot nilai kuisoner**

| Kategori Jawaban | Bobot |
|---|---|
| Sangat Tidak Setuju (STS) | 1 |
| Tidak Setuju (TS) | 2 |
| Ragu-Ragu (RR) | 3 |
| Setuju (S) | 4 |
| Sangat Setuju (SS) | 5 |

2. Persentase Jawaban

   Persentase masing-masing jawaban dihitung dengan menggunakan rumus sebagaimana berikut ini:

$$P = \frac{S}{skorideal} \times 100\% \qquad (2.3)$$

   Keterangan:

   P = Nilai persentase yang dicari

   S = Jumlah frekuensi dikalikan dengan nilai yang di tetapkan jawaban

   Skor Ideal = Nilai tertinggi dikalikan dengan jumlah sampel [23]